

CONFERENCES IN RESEARCH AND PRACTICE IN
INFORMATION TECHNOLOGY

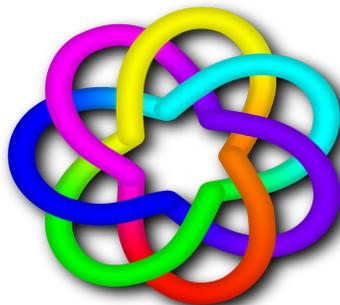
VOLUME 113

COMPUTER SCIENCE 2011

AUSTRALIAN COMPUTER SCIENCE COMMUNICATIONS, VOLUME 33, NUMBER 1



AUSTRALIAN
COMPUTER
SOCIETY



 **CORE**
Computing Research & Education

COMPUTER SCIENCE 2011

Proceedings of the
Thirty-Fourth Australasian Computer Science Conference
(ACSC 2011), Perth, Australia,
17–20 January 2011

Mark Reynolds, Ed.

Volume 113 in the Conferences in Research and Practice in Information Technology Series.
Published by the Australian Computer Society Inc.



Published in association with the ACM Digital Library.

Computer Science 2011. Proceedings of the Thirty-Fourth Australasian Computer Science Conference (ACSC 2011), Perth, Australia, 17–20 January 2011

Conferences in Research and Practice in Information Technology, Volume 113.

Copyright ©2011, Australian Computer Society. Reproduction for academic, not-for-profit purposes permitted provided the copyright text at the foot of the first page of each paper is included.

Editor:

Mark Reynolds

School of Computer Science and Software Engineering
Faculty of Engineering, Computing and Mathematics
The University of Western Australia
Crawley, WA 6009
Australia
Email: mark@csse.uwa.edu.au

Series Editors:

Vladimir Estivill-Castro, Griffith University, Queensland
Simeon J. Simoff, University of Western Sydney, NSW
Email: crpit@scm.uws.edu.au

Publisher: Australian Computer Society Inc.
PO Box Q534, QVB Post Office
Sydney 1230
New South Wales
Australia.

Conferences in Research and Practice in Information Technology, Volume 113.
ISSN 1445-1336.
ISBN 978-1-920682-93-4.

Printed, January 2011 by University of Western Sydney, on-line proceedings
Document engineering by CRPIT
CD Cover Design by Dr Patrick Peursum, Curtin University of Technology
CD Production by Snap St Georges Terrace, 181 St Georges Terrace, Perth WA 6000,
<http://www.stgeorges.snap.com.au/>

The *Conferences in Research and Practice in Information Technology* series disseminates the results of peer-reviewed research in all areas of Information Technology. Further details can be found at <http://crpit.com/>.

Table of Contents

Proceedings of the Thirty-Fourth Australasian Computer Science Conference (ACSC 2011), Perth, Australia, 17–20 January 2011

Preface	vii
Programme Committee	viii
Organising Committee	ix
Welcome from the Organising Committee	x
CORE - Computing Research & Education	xi
ACSW Conferences and the Australian Computer Science Communications	xii
ACSW and ACSC 2011 Sponsors	xiv

Contributed Papers

Building Instance Knowledge Network for Word Sense Disambiguation	3
<i>Shangfeng Hu, Chengfei Liu, Xiaohui Zhao and Marek Kowalkiewicz</i>	
Human Action Recognition Using Silhouette Histogram	11
<i>Chaur-Heh Hsieh, Pingsheng Huang and Ming-Da Tang</i>	
Performance Improvement of Vertical Handoff Algorithms for QoS Support over Heterogeneous Wire- less Networks	17
<i>Shusmita Anwar Sharna and Manzur M. Murshed</i>	
Resource Provisioning based on Leases Preemption in InterGrid	25
<i>Mohsen Amini Salehi, Bahman Javadi and Rajkumar Buyya</i>	
Jointly Compatible Pair Linking for Visual Tracking with Probabilistic Priors	35
<i>Robert Reid</i>	
Hybrid Wrapper-filter Approaches for Input Feature Selection using Maximum relevance-Minimum redundancy and Artificial Neural Network Input Gain Measurement Approximation (ANNIGMA) ...	43
<i>Shamsul Huda, John Yearwood and Andrew Straneieri</i>	
Web-Based Visualisations Supporting Rehabilitation of Heart Failure Patients by Promoting Be- havioural Change	53
<i>Susanne Fischer, Burkhard Wüensche, Linda Cameron, Eva Rose Morunga, Umang Parikh, Lana Jago and Stefan Mueller</i>	
Visualizing the Refactoring of Classes via Clustering	63
<i>Keith Cassell, Craig Anslow, Lindsay Groves, Peter Andreae and Stuart Marshall</i>	
Optimistic and Efficient Concurrency Control for Asynchronous Collaborative Systems	73
<i>Hai Feng Shen and Yongyao Yan</i>	
A Dynamic Archive Based Niching Particle Swarm Optimizer Using a Small Population Size	83
<i>Zhaolin Zhai and Xiaodong Li</i>	
Optimized Relative Lempel-Ziv Compression of Genomes	91
<i>Shanika Kuruppu, Simon Puglisi and Justin Zobel</i>	

A Novel Fuzzy Logic Based Bit Freezing Technique to Improve Gene Specific Co-regulation Discovery from Gene Expression Databases	99
<i>Ji Zhang</i>	
Enhancing the Believability of Embodied Conversational Agents through Environment-, Self- and Interaction-Awareness	107
<i>Kiran Ijaz, Anton Bogdanovych and Simeon Simoff</i>	
EvoJava: A Tool for Measuring Evolving Software	117
<i>Joshua Oosterman, Warwick Irwin and Neville Churcher</i>	
Dynamic Visualisation of Software State	127
<i>James Ashford, Neville Churcher and Warwick Irwin</i>	
Analysis of Key Installation Protection using Computerized Red Teaming	137
<i>Ranjeet Tirtha, Philip Hingston, Chiou Lam and Martin Masek</i>	
A New Term Ranking Method based on Relation Extraction and Graph Model for Text Classification	145
<i>Dat Huynh, Dat Tran and Wanli Ma</i>	
Comparison of Binary and Multi-Variate Hybrid Decision Diagram Algorithms for K-Terminal Reliability	153
<i>Johannes U. Herrmann and Sieteng Soh</i>	
Author Index	163

Preface

The Australasian Computer Science Conference (ACSC) series is an annual forum, bringing together research sub-disciplines in Computer Science. The meeting allows academics and researchers to discuss research topics as well as progress in the field, and policies to stimulate its growth. This volume contains papers presented at the Thirty-Fourth ACSC in Perth, Australia. ACSC 2011 is part of the Australasian Computer Science Week which ran from January 17 to January 20, 2011.

The ACSC 2011 call for papers solicited contributions in all areas of research in Computer Science. This year's conference received 53 submissions from Australia, New Zealand, Pakistan, Taiwan, Bangladesh, South Africa, Germany and the Netherlands. The topics addressed by the submitted papers illustrate the broadness of the discipline.

The programme committee consisted of 32 highly regarded academics from Australia, New Zealand, China, Japan, Canada, Italy, Singapore the UK and USA. All papers were reviewed by at least three programme committee members, and, in some cases, external reviewers. Of the 53 papers submitted, 18 were selected for presentation at the conference.

The "Best Student Paper Award" went to Kiran Irjaz for her contribution reflected in the paper "Enhancing the Believability of Embodied Conversational Agents through Environment-, Self- and Interaction-Awareness", co-authored with her supervisor Dr Anton Bogdanovych and Professor Simeon Simoff, as a result of a major component from progress towards higher degree research. This was also selected for the "Best Paper Award".

I thank all authors who submitted papers and all conference participants for helping to make the conference a success. I also thank the members of the programme committee and the external referees for their expertise in carefully reviewing the papers. I am grateful to Professor Simeon Simoff from University of Western Sydney representing CRPIT for his assistance in the production of the proceedings. I thank Professor Tom Gedeon (President) and Dr Alex Potanin (Conference Coordinator) for their support representing CORE (the Computing Research and Education Association of Australasia).

Thanks to the School of Computer Science and Software Engineering at The University of Western Australia for web support for organising the conference.

Last, but not least, I express gratitude to our hosts in Perth (in particular Mihai Lazarescu and Ling Li).

Mark Reynolds
The University of Western Australia
ACSC 2011 Programme Chair
January 2011

Programme Committee

Chair

Mark Reynolds, University of Western Australia

Members

Carole Adam, RMIT University (Australia)
Michael Albert, University of Otago (New Zealand)
Steve Blackburn, Australian National University (Australia)
Stephane Bressan, National University of Singapore (Singapore)
Fred Brown, The University of Adelaide (Australia)
Rajkumar Buyya, University of Melbourne (Australia)
Andy Cockburn, University of Canterbury (New Zealand)
Curtis Dyreson, Utah State University (USA)
Vladimir Estivill-Castro, Griffith University (Australia)
Ansgar Fehnker, NICTA, Sydney (Australia)
Ken Hawick, Massey University - Albany (New Zealand)
Xiangjian He, University of Technology Sydney (Australia)
Michael Houle, National Institute for Informatics (Japan)
Paddy Krishnan, Bond University (Australia)
Chiou-Peng Lam, Edith Cowan University (Australia)
Bruce Litow, James Cook University (Australia)
Chris McDonald, University of Western Australia (Australia)
Tanja Mitrovic, Canterbury University (New Zealand)
Linda Pagli, University of Pisa (Italy)
Maurice Pagnucco, University of New South Wales (Australia)
Alex Potanin, Victoria University of Wellington (New Zealand)
Rajeev Raman, University of Leicester (UK)
Yuping Shen, Sun Yat-Sen University, Guangzhou, (China)
Markus Stumptner, University of South Australia (Australia)
Xiaoming Sun, Tsinghua University (China)
David Toman, University of Waterloo (Canada)
Hua Wang, University of Southern Queensland (Australia)
Thomas Wolle, NICTA, Sydney (Australia)
Burkhard Wuenche, University of Auckland (New Zealand)
Masafumi Yamashita, Kyushu University (Japan)
Xiaofang Zhou, University of Queensland (Australia)
Albert Zomaya, The University of Sydney (Australia)

Additional Reviewers

Craig Anslow	Shuji Kijima	John Skaller
Masahiro Baba	Pinyan Lu	Lili Sun
Maurizio Bonuccelli	Fabrizio Luccio	Christophe Tartary
Will Browne	Xiaokai Luo	Andreas Vogelsang
Liping Chongjian	Stuart Marshall	Ian Welch
Marcus Freen	David Martinez	Mengjie Zhang
Saurabh Garg	Wolfgang Mayer	Min Zhang
Benoit Gaudou	Nadia Pisanti	Qin Zhang
Georg Grossmann	Giuseppe Prencipe	
Minlie Huang	Nishant Sinha	

Organising Committee

Chair

Assoc. Prof. Mihai Lazarescu

Co-Chair

Assoc. Prof. Ling Li

Finance

Mary Simpson
Mary Mulligan

Catering and Booklet

Mary Mulligan
Dr. Patrick Puersum
Assoc. Prof. Mihai Lazarescu

Sponsorship and Web

Dr. Patrick Peursum
Dr. Aneesh Krishna

Registration

Mary Mulligan
Dr. Patrick Puersum

DVD and Signage

Dr. Patrick Peursum
Mary Mulligan

Venue

Dr. Mike Robey

Conference Bag

Dr. Sieteng Soh

Welcome from the Organising Committee

On behalf of the Australasian Computer Science Week 2011 (ACSW2011) Organising Committee, we welcome you to this year's event hosted by Curtin University. Curtin University's vision is to be an international leader shaping the future through its graduates and world class research. As Western Australia's largest university, Curtin is leading the state in producing high quality ICT graduates. At Curtin Computing, we offer both world class courses and research. Our Computing courses cover three key areas in IT (Computer Science, Software Engineering and Information Technology), are based on the curricula recommendations of IEEE Computer Society and ACM, the largest IT professional associations in the world, and are accredited by the Australian Computer Society. Curtin Computing hosts a top level research institute (IMPCA) and offers world class facilities for large scale surveillance and pattern recognition.

We welcome delegates from over 18 countries, including Australia, New Zealand, USA, U.K., Italy, Japan, China, Canada, Germany, Spain, Pakistan, Austria, Ireland, South Africa, Taiwan and Thailand. We hope you will enjoy the experience of the ACSW 2011 event and get a chance to explore our wonderful city of Perth. Perth City Centre is located on the north bank of the Swan River and offers many fun activities and a wealth of shopping opportunities. For panoramic views of Perth and the river, one can visit Kings Park or enjoy a relaxing picnic in one of the many recreational areas of the park.

The Curtin University campus, the venue for ACSW2011, is located just under 10km from the Perth City Centre and is serviced by several Transperth bus routes that travel directly between Perth and Curtin University Bus Station, as well as several other routes connecting to nearby train services.

ACSW2011 consists of the following conferences:

- Australasian Computer Science Conference (ACSC) (Chaired by Mark Reynolds)
- Australasian Computing Education Conference (ACE) (Chaired by John Hamer and Michael de Raadt)
- Australasian Database Conference (ADC) (Chaired by Heng Tao Shen and Athman Bouguettaya)
- Australasian Information Security Conference (AISC) (Chaired by Colin Boyd and Josef Pieprzyk)
- Australasian User Interface Conference (AUIC) (Chaired by Christof Lutteroth)
- Australasian Symposium on Parallel and Distributed Computing (AusPDC) (Chaired by Jinjun Chen and Rajiv Ranjan)
- Australasian Workshop on Health Informatics and Knowledge Management (HIKM) (Chaired by Kerryn Butler-Henderson and Tony Sahama)
- Computing: The Australasian Theory Symposium (CATS) (Chaired by Taso Viglas and Alex Potanin)
- Australasian Computing Doctoral Consortium (ACDC) (Chaired by Rachel Cardell-Oliver and Falk Scholer).

The nature of ACSW requires the co-operation of numerous people. We would like to thank all those who have worked to ensure the success of ACSW2011 including the Organising Committee, the Conference Chairs and Programme Committees, our sponsors, the keynote speakers and the delegates. Many thanks go to Alex Potanin for his extensive advice and assistance and Wayne Kelly (ACSW2010 chair) who provided us with a wealth of information on the running of the conference. ACSW2010 was a wonderful event and we hope we will live up to the expectations this year.

Assoc. Prof. Mihai Lazarescu and Assoc. Prof. Ling Li
Department of Computing, Curtin University

ACSW2011 Co-Chairs
January, 2011

CORE - Computing Research & Education

CORE welcomes all delegates to ACSW2011 in Perth. CORE, the peak body representing academic computer science in Australia and New Zealand, is responsible for the annual ACSW series of meetings, which are a unique opportunity for our community to network and to discuss research and topics of mutual interest. The original component conferences ACSC, ADC, and CATS, which formed the basis of ACSWin the mid 1990s now share this week with six other events - ACE, AISC, AUIC, AusPDC, HIKM, ACDC, which build on the diversity of the Australasian computing community.

In 2011, we have again chosen to feature a small number of plenary speakers from across the discipline: Heng To Shen, Gene Tsudik, and Dexter Kozen. I thank them for their contributions to ACSW2011. I also thank the keynote speakers invited to some of the individual conferences. The efforts of the conference chairs and their program committees have led to strong programs in all the conferences again, thanks. And thanks are particularly due to Mihai Lazarescu and his colleagues for organising what promises to be a strong event.

In Australia, 2009 saw, for the first time in some years, an increase in the number of students choosing to study IT, and a welcome if small number of new academic appointments. Also welcome is the news that university and research funding is set to rise from 2011-12. However, it continues to be the case that per-capita funding for computer science students has fallen relative to that of other physical and mathematical sciences, and, while bodies such as the Australian Council of Deans of ICT seek ways to increase student interest in the area, more is needed to ensure the growth of our discipline.

During 2010, CORE continued to negotiate with the ARC on journal and conference rankings. A key aim is now to maintain the rankings, which are widely used overseas as well as in Australia. Management of the rankings is a challenging process that needs to balance competing special interests as well as addressing the interests of the community as a whole.

CORE's existence is due to the support of the member departments in Australia and New Zealand, and I thank them for their ongoing contributions, in commitment and in financial support. Finally, I am grateful to all those who gave their time to CORE in 2010; in particular, I thank Alex Potanin, Jenny Edwards, Alan Fekete, Aditya Ghose, Leon Sterling, and the members of the executive and of the curriculum and ranking committees.

Tom Gedeon

President, CORE
January, 2011

ACSW Conferences and the Australian Computer Science Communications

The Australasian Computer Science Week of conferences has been running in some form continuously since 1978. This makes it one of the longest running conferences in computer science. The proceedings of the week have been published as the *Australian Computer Science Communications* since 1979 (with the 1978 proceedings often referred to as *Volume 0*). Thus the sequence number of the Australasian Computer Science Conference is always one greater than the volume of the Communications. Below is a list of the conferences, their locations and hosts.

2012. Volume 34. Host and Venue - RMIT University, Melbourne, VIC.

2011. Volume 33. Host and Venue - Curtin University of Technology, Perth, WA.

2010. Volume 32. Host and Venue - Queensland University of Technology, Brisbane, QLD.

2009. Volume 31. Host and Venue - Victoria University, Wellington, New Zealand.

2008. Volume 30. Host and Venue - University of Wollongong, NSW.

2007. Volume 29. Host and Venue - University of Ballarat, VIC. First running of HDKM.

2006. Volume 28. Host and Venue - University of Tasmania, TAS.

2005. Volume 27. Host - University of Newcastle, NSW. APBC held separately from 2005.

2004. Volume 26. Host and Venue - University of Otago, Dunedin, New Zealand. First running of APCCM.

2003. Volume 25. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Adelaide Convention Centre, Adelaide, SA. First running of APBC. Incorporation of ACE. ACSAC held separately from 2003.

2002. Volume 24. Host and Venue - Monash University, Melbourne, VIC.

2001. Volume 23. Hosts - Bond University and Griffith University (Gold Coast). Venue - Gold Coast, QLD.

2000. Volume 22. Hosts - Australian National University and University of Canberra. Venue - ANU, Canberra, ACT. First running of AUIC.

1999. Volume 21. Host and Venue - University of Auckland, New Zealand.

1998. Volume 20. Hosts - University of Western Australia, Murdoch University, Edith Cowan University and Curtin University. Venue - Perth, WA.

1997. Volume 19. Hosts - Macquarie University and University of Technology, Sydney. Venue - Sydney, NSW. ADC held with DASFAA (rather than ACSW) in 1997.

1996. Volume 18. Host - University of Melbourne and RMIT University. Venue - Melbourne, Australia. CATS joins ACSW.

1995. Volume 17. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Glenelg, SA.

1994. Volume 16. Host and Venue - University of Canterbury, Christchurch, New Zealand. CATS run for the first time separately in Sydney.

1993. Volume 15. Hosts - Griffith University and Queensland University of Technology. Venue - Nathan, QLD.

1992. Volume 14. Host and Venue - University of Tasmania, TAS. (ADC held separately at La Trobe University).

1991. Volume 13. Host and Venue - University of New South Wales, NSW.

1990. Volume 12. Host and Venue - Monash University, Melbourne, VIC. Joined by Database and Information Systems Conference which in 1992 became ADC (which stayed with ACSW) and ACIS (which now operates independently).

1989. Volume 11. Host and Venue - University of Wollongong, NSW.

1988. Volume 10. Host and Venue - University of Queensland, QLD.

1987. Volume 9. Host and Venue - Deakin University, VIC.

1986. Volume 8. Host and Venue - Australian National University, Canberra, ACT.

1985. Volume 7. Hosts - University of Melbourne and Monash University. Venue - Melbourne, VIC.

1984. Volume 6. Host and Venue - University of Adelaide, SA.

1983. Volume 5. Host and Venue - University of Sydney, NSW.

1982. Volume 4. Host and Venue - University of Western Australia, WA.

1981. Volume 3. Host and Venue - University of Queensland, QLD.

1980. Volume 2. Host and Venue - Australian National University, Canberra, ACT.

1979. Volume 1. Host and Venue - University of Tasmania, TAS.

1978. Volume 0. Host and Venue - University of New South Wales, NSW.

Conference Acronyms

ACDC	Australasian Computing Doctoral Consortium
ACE	Australasian Computer Education Conference
ACSC	Australasian Computer Science Conference
ACSW	Australasian Computer Science Week
ADC	Australasian Database Conference
AISC	Australasian Information Security Conference
AUIC	Australasian User Interface Conference
APCCM	Asia-Pacific Conference on Conceptual Modelling
AusPDC	Australasian Symposium on Parallel and Distributed Computing (replaces AusGrid)
CATS	Computing: Australasian Theory Symposium
HIKM	Australasian Workshop on Health Informatics and Knowledge Management

Note that various name changes have occurred, which have been indicated in the Conference Acronyms sections in respective CRPIT volumes.

ACSW and ACSC 2011 Sponsors

We wish to thank the following sponsors for their contribution towards this conference.



CORE - Computing Research and Education,
www.core.edu.au



Perth Convention Bureau,
www.pcb.com.au



**AUSTRALIAN
COMPUTER
SOCIETY**

Australian Computer Society,
www.acs.org.au



Curtin University of Technology,
www.curtin.edu.au



THE UNIVERSITY OF
WESTERN AUSTRALIA
University of Western Australia,
www.uwa.edu.au

CONTRIBUTED PAPERS

Building Instance Knowledge Network for Word Sense Disambiguation

Shangfeng Hu, Chengfei Liu

Faculty of Information and Communication Technologies
Swinburne University of Technology
Hawthorn 3122, Victoria, Australia
{shu, cliu}@groupwise.swin.edu.au

Xiaohui Zhao

Information Systems Group
Department of Industrial Engineering & Innovation Sciences
Eindhoven University of Technology
Eindhoven, The Netherlands
x.zhao@tue.nl

Marek Kowalkiewicz

SAP Research
Brisbane, Australia
marek.kowalkiewicz@sap.com

Abstract

In this paper, a new high precision focused word sense disambiguation (WSD) approach is proposed, which not only attempts to identify the proper sense for a word but also provides the probabilistic evaluation for the identification confidence at the same time. A novel Instance Knowledge Network (IKN) is built to generate and maintain semantic knowledge at the word, type synonym set and instance levels. Related algorithms based on graph matching are developed to train IKN with probabilistic knowledge and to use IKN for probabilistic word sense disambiguation. Based on the Senseval-3 all-words task, we run extensive experiments to show the performance enhancements in different precision ranges and the rationality of probabilistic based automatic confidence evaluation of disambiguation. We combine our WSD algorithm with five best WSD algorithms in senseval-3 all words tasks. The results show that the combined algorithms all outperform the corresponding algorithms.

Keywords: natural language processing, word sense disambiguation

1 Introduction

Word sense disambiguation (WSD) is to identify the proper sense of words in the context. As a typical topic of natural language processing, WSD is widely used in machine translation, knowledge acquisition, information retrieval, etc. (Navigli 2009)

As a knowledge system in nature, WSD heavily relies on knowledge resources. Supervised WSD approaches

mostly require manual sense tagged corpus. They provide the best performances in public evaluation (Palmer et al. 2001; Snyder and Palmer 2004). There are some knowledge based WSD systems which are built on a lexical knowledge base. They exploit the semantic relationships between concepts in semantic networks and computational lexicons (Yarowsky and Florian 2002; Cuadros and Rigau 2006).

Recently, a few graph based approaches for knowledge based WSD were proposed (Navigli and Velardi 2005; Sinha and Mihalcea 2007; Navigli and Lapata, 2007; Agirre and Soroa 2009). All these approaches are built at type level and the semantic relations between types.

Besides semantic relations, syntactic structures and relations are also valuable to WSD. Martinez et al. proposed a syntactic feature based WSD approach (Martinez et al. 2002). Fernandez-Amoros also presented a syntactic pattern WSD algorithm (Fernandez-Amoros 2004).

However there is no knowledge base for WSD systems which properly keeps both semantic relations and syntactic features in the context. Actually, relationships between two synsets may be different within different syntactic structures of the contexts. To reflect this difference, we consider keeping context related relationships between synsets in patterns at instance level.

Instance based learning (Ng and Lee 1996; Daelemans et al. 1999) is a promising approach for WSD. Instance based WSD algorithms do not neglect exceptions and accumulate further aid for disambiguation through new examples. However existing instance based WSD approaches do not consider the syntactic features. We believe that keeping syntactic structures as instance patterns will benefit WSD and it is a key point to combine semantic relationships and syntactic features.

Besides the above considerations about a knowledge base, we also concern about the accuracy of WSD results. The poor accuracy of WSD results is a bottleneck for the

Copyright © 2011, Australian Computer Society, Inc. This paper appeared at the 34th Australasian Computer Science Conference (ACSC 2011), Perth, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 113. M. Reynolds, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

application of WSD (Navigli 2009). Inspired by the human learning process, we reckon that the confidence evaluation of WSD is important. Supposing a school girl is reading a text, even though she cannot understand the whole text, she knows which part she can understand and which part she cannot. Therefore, she can learn knowledge from the understood part with high confidence. The confidence is based on the evaluation on accuracy of understanding.

Martinez's approach (Martinez et al. 2002) provides high precision WSD results. This work emphasizes the importance of high precision WSD. However, it employs fixed rule-related thresholds without quantitative evaluation of disambiguation results. Preiss (Preiss 2004) proposed a probabilistic WSD approach. However, they convert the probabilistic results into qualitative ones without quantitative analysis and do not show how their probabilistic results relate to the accuracy of disambiguation.

In this paper, we propose a novel multilayer instance knowledge network (IKN) together with related probabilistic training and WSD algorithms. The IKN WSD algorithm combines the semantic relations and syntactic features together to provide WSD results with quantitative confidence evaluation.

The rest of the paper is structured as follows. In Section 2, we introduce IKN, its graph matching algorithm and the probabilistic training algorithm of IKN based on the graph matching algorithm. The IKN WSD algorithm is presented in Section 3. Section 4 presents the experiment results of the IKN WSD algorithm and its combinations with existing WSD algorithms. Related works are discussed in Section 5. Concluding remarks are given in Section 6.

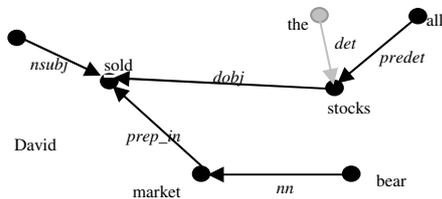


Figure 1 Semantic Dependency Graph for the sentence “David sold all the stocks in bear market.”

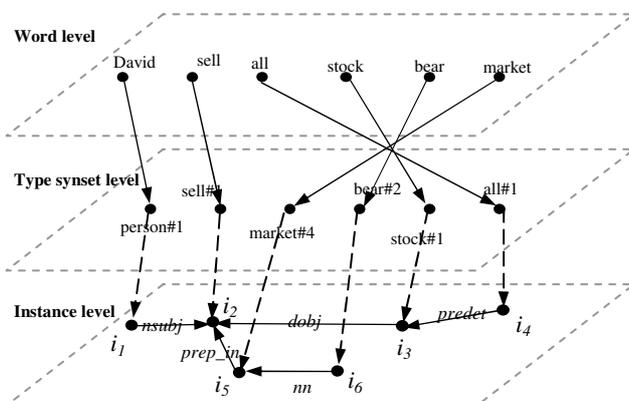


Figure 2 Connecting instance graph pattern to WordNet

2 Instance Knowledge Network (IKN)

Text understanding often requires contextual knowledge beyond the literal information of the text itself. Such contextual knowledge can be searched from the previously understood contents that are similar to the current text. Technically, such knowledge can be maintained in a semantic knowledge network which stores instances of word senses and relationships between these instances.

We propose a novel IKN - instance knowledge network to keep the contextual knowledge between word senses. IKN keeps the relationships between word senses not only at the type level (i.e., relations between type synsets) but also at the instance level (i.e., through a series of instance dependency graphs that are connected to the type synsets). It differs from most knowledge networks (Cuadros and Rigau 2008; Agirre and Soroa 2009) which keep the knowledge in the type synsets and their relations only.

2.1 Constructing IKN

We create IKN by extending WordNet (Fellbaum 1998) with a sense tagged corpus, such as, SemCor (Miller et al. 1993).

Figure 2 shows the simplified structure of IKN. The word level and the type synset level are inherited from WordNet. The instance level consists of a collection of *instance graph patterns (IGPs)*, which are built from the texts in an existing corpus. We used SemCor in our system. The relationships between the instance level and the type synset level also need to be created. The brief procedure is as follows:

First, all the texts in the corpus are parsed into semantic dependency graphs using Stanford Dependency Parser (Stanford_Parser, Klein et al. 2002). Figure 1 shows a dependency graph for the sentence “David sold all the stocks in bear market.” We suppose the sentence is sense tagged by WordNet3.0 synonym set (synset), where David, sold, all, stocks, bear and market are assigned with synsets person#1, sell#1, all#1, stock#1, bear#2 and market#4, respectively. Each dependency graph is then inserted into IKN as an IGP by setting a unique identifier to each word node of the graph and the word node becomes an instance node of the IGP. Obviously, an IGP inherits the dependency relations between instance nodes from the dependency graph.

Then we connect instance nodes in each IGP at the instance level to type synsets at the synset level. Because each word in dependency graphs is sense tagged, each instance node has a sense tag inherits from the word. We connect each instance node to the type synset labeled by the sense tag. It is worthwhile noting that the relation between type synsets and instance nodes is one-to-many, i.e., a type synset may connect to multiple instance nodes.

In Figure 2, the IGP ($\{i_1, i_2, i_3, i_4, i_5, i_6\}, \{(i_1, i_2), (i_2, i_3), (i_2, i_5), (i_3, i_4), (i_5, i_6)\}$) at the instance level is obtained from the dependency graph in Figure 1. Instance node i_1 coming from word node “David” in Figure 1 is connected to its tagged sense synset person#1 at the synset level. To simplify the representation, the relations between type synsets are not given. IKN will be trained to obtain probabilistic knowledge (see Section 2.3).

2.2 Graph Matching Algorithm

Given a candidate sentence represented as a dependency graph G shown at the candidate level of Figure 3, we propose a graph matching algorithm to find all matching sub-graphs of IGPs in IKN for dependency graph G . The training algorithm for IKN and WSD algorithm will be based on this algorithm.

The algorithm can be described as 2 main steps.

- (1). For each candidate word of dependency graph G at the candidate level, we find all instance nodes at the instance level that are *semantically related* to the word through IKN. We call these instance nodes as *semantic related instance nodes (SRINs)* of the candidate word.
- (2). Among all SRINs of those candidate words in G , we discover all sub-graphs of IGPs that match G maximally.

Figure 3 shows a general picture on how the graph matching works. We now describe in detail these two main steps in the following two sub sections.

2.2.1 Finding Semantic Related Instance Nodes

Given candidate word w of candidate dependency graph G at the candidate level, we need to find all SRINs of w at the instance level through the word and type synset levels of

IKN. This can be done in the following three sub steps.

Firstly, given a candidate word w , we find all sense synsets of w . We first find a unique symbol word w' at the word level for w . Then, we find the sense synsets of w' at the type synset level. Note that multiple sense synsets may exist for w' . We also consider these synsets as the sense synsets of w . For example in Figure 3, candidate word *market* in dependency graph G_1 has a symbol word *market* at the word level. Two synsets *market#3* and *market#4* at the type synset level are sense synsets of symbol word *market* as well as the sense synsets of candidate word *market* in G_1 . Similarly, *clear#16* is a sense synset of candidate word *cleared* in G_1 , and *assets#1* is a sense synset of candidate word *assets* in G_2 .

Secondly, given a set of sense synsets of w , we find all synsets that are semantically related to these sense synsets within the type synset level. For each sense synset s of w , there may exist other synsets at the type synset level which are *semantically related* to s . We call these synsets as *semantically related synsets (SRSs)* of s in IKN. An SRS s_i of s is defined as a synset which holds one of the following three relationships with s .

- (1). A single semantic relation exists from s to s_i within the type synset level;
- (2). A semantic path exists from s to s_i within the type synset level, each step of the path has the same semantic relation and direction, and the semantic

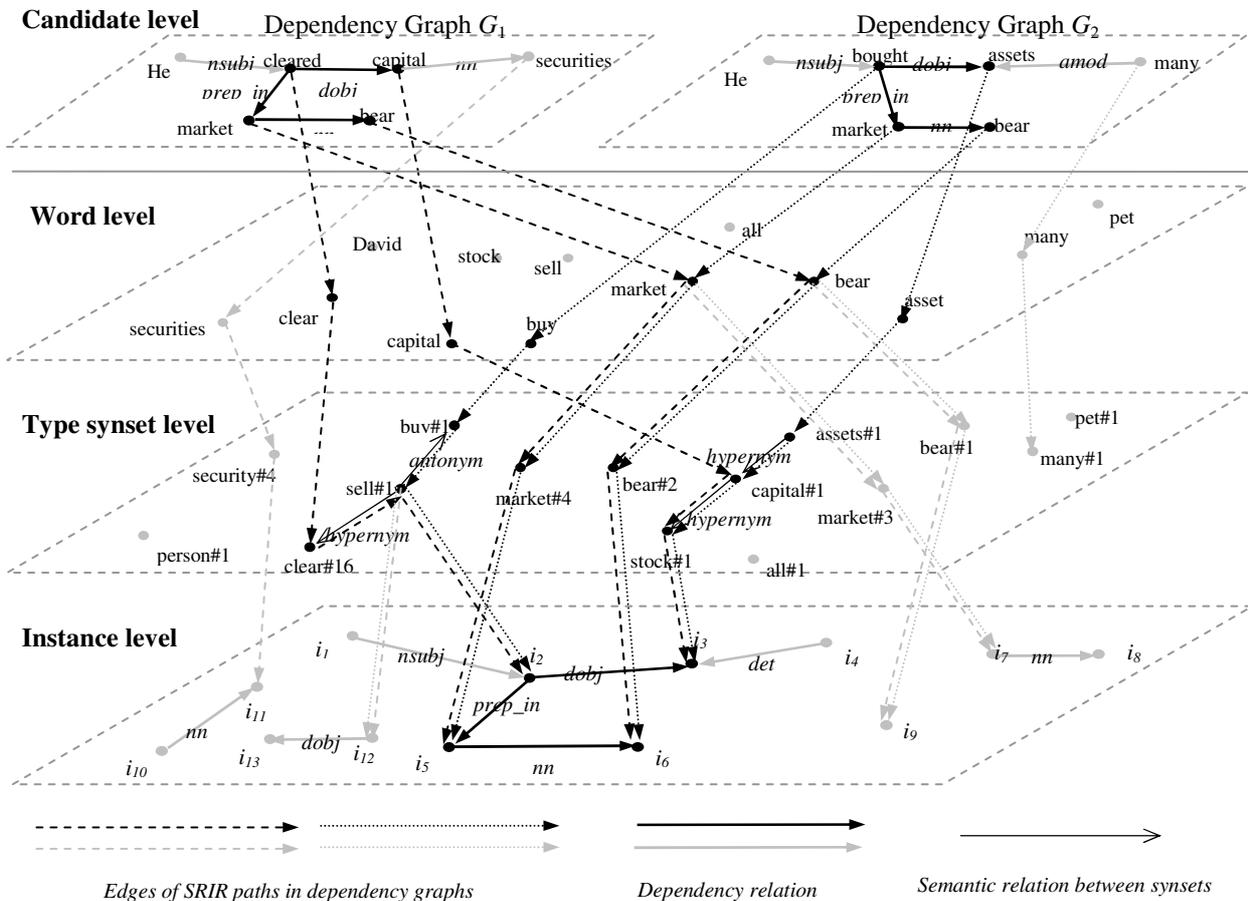


Figure 3 Graph Matching Algorithm on Instance Knowledge Network

relation is transitive;

(3). Sense synset s is an SRS of itself, denoted as *self*.

An SRS of a candidate word is an SRS of one of the sense synsets of the candidate word. For example, in Figure 3, the synset sell#1 is an SRS of candidate word *cleared* with the semantic relation *hypernym* to its sense synset clear#16; it is also an SRS of candidate word *bought* with the semantic relation *antonym* to its sense synset buy#1. The synset stock#1 is an SRS of *capital* with the direct *hypernym* relation to capital#1; it is also an SRS of *assets* to assets#1 with *hypernym* as an indirect transitive relation. We also consider market#3 and market#4 as SRSs of candidate word *market* because both market#3 and market#4 are sense synsets of *market* and they are deemed as SRSs with *self* semantic relation to the sense synsets.

Thirdly, given a set of SRSs of w , we find all SRINs of these SRSs. In IKN, a synset can have multiple instance nodes, so there can be multiple SRINs which are instances of each SRS of w . For example in Figure 3, sell#1 is an SRS of *cleared*, and it has two SRINs i_2 and i_{12} .

So when given a candidate word w , multiple SRINs may be returned. For each returned SRIN n , we define $SRIR(w, n)$ as the *semantically related instance relationship (SRIR)* between w and n . From the above three sub steps, we know that such an SRIR stands for a path from w to n , including relations between a candidate word at the candidate level and its word symbol at the word level, between a word symbol and a sense synset at the type synset level, between a sense synset and an SRS both at the type synset level, and between an SRS and an instance node at the instance level. $SRIR(w, n)$ can be denoted as (w, s, t, n) where w is the candidate word; s is the sense synset of w on the path; t is the semantic relation between s and the SRS s' on the path which directly connects to the instance node n . Given a particular SRIR r , we define the function $SS(r)$ to return sense synset s and define the function $SRC(r)$ to return the semantic relation t .

For example, in Figure 3, we denote $SRIR(cleared, i_2)$ between the candidate word *cleared* and its SRIN i_2 as $(cleared, clear\#16, hyponym, i_2)$ and we can get $SS(SRIR(cleared, i_2)) = clear\#16$ and $SRC(SRIR(cleared, i_2)) = hyponym$. Similarly we have $SRIR(assets, i_3)$ as $(assets, assets\#1, hypernym, i_3)$, $SS(SRIR(assets, i_3)) = assets\#1$ and $SRC(SRIR(assets, i_3)) = hyponym$; $SRIR(capital, i_3)$ as $(capital, capital\#1, hypernym, i_3)$, $SS(SRIR(capital, i_3)) = capital\#1$ and $SRC(SRIR(capital, i_3)) = hyponym$; $SRIR(market, i_5)$ as $(market, market\#4, self, i_5)$, $SS(SRIR(market, i_5)) = market\#4$ and $SRC(SRIR(market, i_5)) = self$; $SRIR(bear, i_6)$ as $(bear, bear\#2, self, i_6)$, $SS(SRIR(bear, i_6)) = bear\#2$ and $SRC(SRIR(bear, i_6)) = self$.

It is worth mentioning that an SRS can also correspond to multiple sense synsets, and hence multiple candidate words. Consequently an instance node or SRIN can correspond to multiple sense synsets as well as multiple candidate words. For example in Figure 3, i_2 is an instance of sell#1, sell#1 is an SRS of candidate word *cleared* with the semantic relation *hypernym*, so we consider i_2 as an SRIN of candidate word *cleared* with semantic relation

hypernym to its sense synset clear#16. Similarly, i_2 is also an SRIN of *bought* with semantic relation *antonym*.

2.2.2 Discovering Instance Matching Sub-graphs

When all SRINs of words in candidate dependency graph G are found, we now discover all sub-graphs of IGP at the instance level that match G maximally. This can be achieved by a breadth-first traversal of G . To give a clear explanation, we divide it in following two steps.

Firstly, for each edge $e(w_1, w_2)$ being traversed in G , we find its matching edges in all IGP at the instance level. An edge $e'(iw_1, iw_2)$ in an IGP G' is called a matching edge of $e(w_1, w_2)$ if it satisfies the following conditions:

- (1). iw_1 and iw_2 are an SRIN of w_1 and w_2 , respectively;
- (2). The dependency relation d between w_1 and w_2 in G is the same as the dependency relation d' between iw_1 and iw_2 in the IGP.

For example in Figure 3, dependency relation between i_2 and i_3 is *dobj* which is the same as the dependency relation between the candidate words *cleared* and *capital*. Furthermore, there is an SRIR between *cleared* and i_2 , and another SRIR between *capital* and i_3 . Thus, we consider (i_2, i_3) as a matching edge of $(cleared, capital)$ of G . Similarly, (i_2, i_5) is a matching edge of $(cleared, market)$ with common dependency relation *prep_in*, and (i_5, i_6) is a matching edge of $(market, bear)$ with common dependency relation *nn*.

Secondly, we try to connect the found matching edge $e'(iw_1, iw_2)$ to those previously found set of matching edges or sub-graphs in the IGP G' . We denote the set of previously found set of matching sub-graphs as S and $S = \{ \}$ at the beginning. $e'(iw_1, iw_2)$ can be connected to one sub-graph G_s in S if G_s includes a node iw that matches either iw_1 or iw_2 , and corresponds to same candidate word w_1 or w_2 . If none of such G_s exists in S , we simply add $e'(iw_1, iw_2)$ as a sub-graph to S .

When the traversal of G is done, we select the maximum sub-graph from S as the matching sub-graph of candidate dependency graph G from IGP G' . We call this sub-graph as an *instance matching sub-graph (IMSG)* of G .

For example in Figure 3, we start breadth-first traversal of G_1 from word *cleared* and $S = \{ \}$ at the beginning for one of the IGP G' at the instance level. After $(cleared, capital)$ is traversed, $S = \{ \{(i_2, i_3)\} \}$ (for simplicity, only edges are recorded). After $(cleared, market)$ is traversed, $S = \{ \{(i_2, i_3), (i_2, i_5)\} \}$. After $(market, bear)$ is traversed, $S = \{ \{(i_2, i_3), (i_2, i_5), (i_5, i_6)\} \}$. After the traversal of G_1 is done, suppose there is no change to S , then the only sub-graph $\{(i_2, i_3), (i_2, i_5), (i_5, i_6)\}$ of S is the IMSG of G from IGP G' . There may be IMSGs from other IGP for G .

2.3 Probabilistic Knowledge Training

Based on the graph matching algorithm, we train IKN by a sense tagged corpus. In our work, we use the SemCor to train IKN which was initially built from SemCor. From the training, probabilistic knowledge are obtained and attached to the IGP.

In the training process, at first we parse the sense tagged corpus into candidate dependency graphs. Then we employ the graph matching algorithm to find IMSGs at the instance level of IKN. Each candidate dependency graph may match with many IMSGs. Each instance node pair in an IGP may be matched by many candidate dependency graphs in different IMSGs. Finally, we generate the conditional probabilities for each instance node pair in the IMSGs. Each time a pair of instance nodes is matched in an IMSG of a candidate dependency graph, some of the conditional probabilities related to them are generated or updated. In the following, we focus on the discussion on how we generate the conditional probabilities for a pair of instance nodes in an IGP of IKN.

For a pair of instance nodes i_1 and i_2 (denoted as $\langle i_1, i_2 \rangle$) in an IGP of IKN, we define two sets of conditional probabilities: $PI(i_1, i_2)$ from i_1 to i_2 , and $PI(i_2, i_1)$ from i_2 to i_1 . i_1 and i_2 may be directly or indirectly connected in the IGP. Each conditional probability in a set is created for its real WSD use and represents the *category* of sense synset pairs based on a *particular SRC pair* (refer to Section 2.2.1 for SRC definition) between $\langle i_1, i_2 \rangle$ and its matching word pairs. Through the training process, $PI(i_1, i_2)$ and $PI(i_2, i_1)$ are obtained and attached to $\langle i_1, i_2 \rangle$ as part of IKN. Due to space limitation, we only explain how we define a set of conditional probabilities in $PI(i_1, i_2)$ as those in $PI(i_2, i_1)$ can be defined similarly.

At first, for instance node pair $\langle i_1, i_2 \rangle$, we define condition C1 for that both i_1 and i_2 are in an IMSG of a candidate dependency graph from the training set. We count the number of times in the training process that C1 is satisfied as $\text{Count}_{\text{all}}(i_1, i_2)$.

Each time $\langle i_1, i_2 \rangle$ satisfies C1 in a matching, a candidate word pair $\langle w_1, w_2 \rangle$ can be found in an candidate dependency graph and then the pair of SRIRs $\text{SRIR}(w_1, i_1)$ and $\text{SRIR}(w_2, i_2)$ are determined. Then we can define condition C2 for satisfying $\text{SRC}(\text{SRIR}(w_1, i_1)) = t_1$ and $\text{SRC}(\text{SRIR}(w_2, i_2)) = t_2$. We count the number of times in training process that both conditions C1 and C2 are satisfied as $\text{Count}_c(i_1, t_1, i_2, t_2)$. Here, for the instance node pair $\langle i_1, i_2 \rangle$, we attempt to categorize the matching candidate word pairs based on different SRCs of the corresponding pairs of SRIRs. For each particular pair of SRCs $\langle t_1, t_2 \rangle$, we count the number of all matching candidate word pairs in the training set which satisfy the SRCs of matching. It is easy to see $\text{Count}_{\text{all}}(i_1, i_2) = \sum \text{Count}_c(i_1, t_i, i_2, t_j)$ for all $\langle t_i, t_j \rangle$ pairs.

After that, for each candidate word w , we define the proper sense synset as $\text{PSS}(w)$ which is the tagged sense synset for w . We define condition $\text{C3}(i_1)$ satisfying $\text{SS}(\text{SRIR}(w_1, i_1)) = \text{PSS}(w_1)$, i.e., the sense synset of the SRIR between w_1 and i_1 is the proper tagged sense of w_1 . Similarly we define condition $\text{C3}(i_2)$ satisfying $\text{SS}(\text{SRIR}(w_2, i_2)) = \text{PSS}(w_2)$. We count the number of candidate word pairs in the training process that satisfy conditions C1, C2 and $\text{C3}(i_1)$ as $\text{Count}_c(i_1, t_1, \text{true}, i_2, t_2, \text{null})$. It stands for the size of the set of candidate word pairs, each pair $\langle w_1, w_2 \rangle$ in the set matches $\langle i_1, i_2 \rangle$, and the pair of SRCs is $\langle t_1, t_2 \rangle$, and the sense synset in $\text{SRIR}(w_1, i_1)$ is the proper sense of w_1 . Similarly, we count the number of candidate word pairs in the training process that satisfy conditions C1, C2 and *not* $\text{C3}(i_1)$ as $\text{Count}_c(i_1, t_1, \text{false}, i_2,$

$t_2, \text{null})$. We also count the number of candidate word pairs in training process that satisfy conditions C1, C2, $\text{C3}(i_1)$ and $\text{C3}(i_2)$ as $\text{Count}_c(i_1, t_1, \text{true}, i_2, t_2, \text{true})$. It stands for the size of the set of candidate word pairs, each pair $\langle w_1, w_2 \rangle$ in the set matches $\langle i_1, i_2 \rangle$, and the pair of SRCs is $\langle t_1, t_2 \rangle$, and the sense synsets in $\text{SRIR}(w_1, i_1)$ and $\text{SRIR}(w_2, i_2)$ are the proper senses of w_1 and w_2 , respectively. Similarly, we count the number of candidate word pairs in training process that satisfy conditions C1, C2, $\text{C3}(i_1)$ and *not* $\text{C3}(i_2)$ as $\text{Count}_c(i_1, t_1, \text{true}, i_2, t_2, \text{false})$.

It is not difficult to understand that $\text{Count}_c(i_1, t_1, \text{true}, i_2, t_2, \text{false}) + \text{Count}_c(i_1, t_1, \text{true}, i_2, t_2, \text{true}) = \text{Count}_c(i_1, t_1, \text{true}, i_2, t_2, \text{null})$, and $\text{Count}_c(i_1, t_1, \text{true}, i_2, t_2, \text{null}) + \text{Count}_c(i_1, t_1, \text{false}, i_2, t_2, \text{null}) = \text{Count}_c(i_1, t_1, \text{null}, i_2, t_2, \text{null}) = \text{Count}_c(i_1, t_1, i_2, t_2)$.

Now, for each *particular SRC pair* $\langle t_1, t_2 \rangle$, we can define a conditional probability from i_1 to i_2 as $P(i_2, t_2 | i_1, t_1) = \text{Count}_c(i_1, t_1, \text{true}, i_2, t_2, \text{true}) / \text{Count}_c(i_1, t_1, \text{true}, i_2, t_2, \text{null})$. We give the formal interpretation of $P(i_2, t_2 | i_1, t_1)$ as follows.

For all matched pairs $\{\langle w_{1i}, w_{2j} \rangle\}$ of $\langle i_1, i_2 \rangle$ with $\text{SRC}(\text{SRIR}(w_{1i}, i_1)) = t_1$ and $\text{SS}(\text{SRIR}(w_{2j}, i_2)) = t_2$, let $s_{1i} = \text{SS}(\text{SRIR}(w_{1i}, i_1))$ and $s_{2j} = \text{SS}(\text{SRIR}(w_{2j}, i_2))$, $P(i_2, t_2 | i_1, t_1)$ is the probability of s_{2j} being the proper sense of w_{2j} when s_{1i} is the proper sense of w_{1i} . $P(i_2, t_2 | i_1, t_1)$ is added in set $PI(i_1, i_2)$ to represent the category of those sense synset pairs based on $\langle t_1, t_2 \rangle$.

The conditional probabilities obtained will be attached to instance node pairs within IGPs. These conditional probabilities in different IGPs materialize the relationships of synset pairs in different contexts. The syntactic structures are also kept in IGPs. So the semantic relations and the syntactic structures can work together in our IKN WSD.

3 IKN WSD Algorithm

Based on IKN and the probabilistic knowledge obtained and attached to the instance level of IKN during the above training process, we propose a quantitative WSD approach, which comprises a word sense based iterative process and a probabilistic reasoning algorithm.

3.1 Word Sense Based Iterative Reasoning

We believe that the understanding of a word may depend on the understanding of other words in the context. As such, we propose an iterative reasoning process which is described as follows:

- (a) We parse the text to be disambiguated into candidate dependency graphs and set an initial probability for every sense of the words in text. All the probabilities of senses for a particular word add up to 1.0.
- (b) Apply our probabilistic reasoning algorithm to update the probability of each word sense by the related sense probabilities of other words. The probabilistic reasoning algorithm will ensure that the sense probabilities of each particular word add up to 1.0.
- (c) Repeat (b) until the probabilities of word senses get stabilized.

Agirre and Soroa (Agirre and Soroa, 2009) proposed an Personalized PageRank WSD approach which is also an iteration algorithm. Their algorithm is based on a ranking system between type synsets. Different from their ranking approach, the input and output of (b) in our algorithm are probability based, so we consider it as a reasoning algorithm.

In this paper, the initial probabilities are derived from WordNet. For each candidate word which has tag of part of speech (PoS) by Stanford parser, we find all the senses for this word in WordNet with the same PoS. WordNet provides *tagcount* between sense and word which shows the frequency of a sense for a word in a large scale corpus. The probability of a sense for a word is its *tagcount* (in WordNet) divided by the summation of the *tagcounts* of all senses of the word with the same PoS.

The probabilistic reasoning algorithm in (b) will be described in next 3 sub-sections. For each sense s of candidate word w in a candidate dependency graph, we find the set of maximum conditional probabilities $\{P(s | w, s_{ij}, w_i)\}$, where w_i is any surrounding word of w in the dependency graph and s_{ij} is any sense synset of w_i (to be discussed in Section 3.2). After that, we calculate the un-normalized sense probability $P_b^k(s|w)$ for the current step by combining the set of sense probabilities at the previous step $\{P^{k-1}(s_{ij}|w_i)\}$ and the set of corresponding maximum conditional probabilities $\{P(s | w, s_{ij}, w_i)\}$ (to be discussed in Section 3.3). Finally, we normalize $P_b^k(s|w)$ to $P^k(s|w)$ to ensure that the sense probabilities of each particular word add up to 1.0 and get the WSD results (to be discussed in Section 3.4).

3.2 Finding Maximum Conditional Probability

To a sense synset s of a candidate word w in a candidate dependency graph G and a sense synset s' of a surrounding candidate word w' of w , We attempt to find the maximum conditional probability $P(s | w, s', w')$ using a set of conditional probability sets. Each probability set $PI(i', i)$ in this set is selected because of a matched SRIN pair $\langle i', i \rangle$ of the candidate word pair $\langle w', w \rangle$. In the following, we explain how we get $P(s | w, s', w')$.

We first find all IMSGs of G in IKN by the graph matching algorithm. For each found IMSG G' which contains the matched SRIN pair $\langle i', i \rangle$ of $\langle w', w \rangle$, if $SS(SRIR(w', i')) = s'$ and $SS(SRIR(w, i)) = s$, then we search a conditional probability $P(i, t | i', t') \in PI(i', i)$ such that $t = SRC(SRIR(w, i))$ and $t' = SRC(SRIR(w', i'))$, here $\langle t', t \rangle$ represents a particular category of sense synset pairs to which $\langle s', s \rangle$ belongs. If such a probability exists, we add it into the probability set $PS(w', s', w, s)$ which is a set of conditional probabilities from s' to s . When we process all found IMSGs of G , $PS(w', s', w, s)$ records all relevant probabilities of matched SRIN pairs $\{\langle i', i \rangle\}$ of the candidate word pair $\langle w', w \rangle$. Now we set $P(s | w, s', w')$ as the maximum conditional probability among those probabilities in $PS(w', s', w, s)$ to represent the probability of s being the proper sense of w when s' is the proper sense of w' .

3.3 Combining Probabilities

To a sense synset s_p of candidate word w in a candidate dependency graph G , based on the conditional probabilities from the sense synsets of different surrounding candidate words in the context and the probabilities of these sense synsets from the previous step, we can calculate the probability of s_p for the current step.

At first, we calculate the weighted average of the conditional probabilities of sense s_p of w from a surrounding word w_i in the context as

$$P^k(s_p | w_i, w) = \sum_{j=1 \text{ to } x} P(s_p | w_i, s_{ij}, w) P^{k-1}(s_{ij} | w_i)$$

Where $P^{k-1}(s_{ij} | w_i)$ is the probability of sense s_{ij} of w_i at iteration step $k-1$ and x is the number of senses of w_i . $P^k(s_p | w_i, w)$ is the conditional probability of sense s_p of word w from word w_i in the context at iteration step k .

Then, we combine the conditional probabilities of sense s_p of w from multiple surrounding words according to Naive Bayes Approach. We define the probability of sense s_p of w as

$$P_b^k(s_p | w) = P_0(s_p | w) \prod_{w_i \in G} \frac{P^k(s_p | w_i, w)}{P_0(s_p | w)}$$

$P_0(s_p | w)$ is the start-up probability of the sense s_p . It is also the general probability of s_p for word w which is calculated from the *tagcount* of the senses of w in WordNet.

Finally, we normalize the probabilities to ensure that the probabilities of all senses of a word add up to 1. We calculate $P_w^k(w) = \sum P_b^k(s_p | w)$, $p = 1, 2, \dots, y$ and y is the number of senses of candidate word w . Then we define the normalized probability

$$P^k(s_p | w) = \begin{cases} P_b^k(s_p | w) / P_w^k(w) & \text{if } P_w^k(w) > 0 \\ P_0(s_p | w) & \text{if } P_w^k(w) = 0 \end{cases}$$

3.4 Returning WSD Results with Confidences

At each iteration step, we choose the sense with the maximum normalized probability of a word as the result sense for the word and employ the probability of this sense as the *confidence* evaluation value for the word disambiguation. The confidence value is between 0 and 1. If there are two senses with the same probability, we select the one with smaller synset *rank* number of WordNet as the result sense of disambiguation. When disambiguation results remain unchanged for all the test words at an iteration step, we get the WSD result.

4 Experiments and Evaluation

In our experiments, IKN is created and then trained by the sense tagged corpus SemCor. All the experiment results are for senseval-3 (Mihalcea et al. 2004) all words task. The PoS tagging for our IKN WSD algorithm is based on Stanford dependency parser, so the results inherit its mistakes.

To get high precision WSD results, we define a threshold θ between 0 and 1 as the confidence value of our WSD algorithm. We choose the results with the confidence value equal to or higher than θ as the high precision WSD results.

In Section 3.1, for each sense synset, we presented the method for getting the start-up probabilities which are based on *tagcount* values in WordNet. To evaluate the experiment results, we present the baseline algorithm which takes the start-up probabilities for each sense synset of words, identifies the sense synset with the maximum probability as the result sense, and employs the probability of the result sense as the confidence value. If there are two sense synsets with the same probability, we also choose the one with lower *rank* number in WordNet.

Table 1 presents the precision and attempt coverage of our IKN WSD algorithm with different confidence thresholds θ , and *comparable* results of the baseline WSD algorithm on senseval-3 all words tasks. As our IKN WSD algorithm provides the result of those words with the *confidence* equal to or higher than θ , we define the number of attempted results as $\text{Attempt}(\theta)$ and the number of the correct results in these attempted results as $\text{Correct}(\theta)$. So the *precision* can be obtained by $\text{Correct}(\theta)/\text{Attempt}(\theta)$. We define the total number of test words as n , then the *attempt coverage* is $\text{Attempt}(\theta)/n$. In Senseval-3 all word tasks, $n = 2041$. The precision and attempt coverage are a pair of contradicting factors.

Normally, higher precision is associated with lower attempt coverage. We sort the WSD results of the baseline algorithm by confidence values in descending order. To compare fairly the precision of the baseline WSD results with our IKN WSD results for each particular θ , we select the baseline WSD results with biggest confidences. The number of the selected results is equal to $\text{Attempt}(\theta)$ (this implies that the same attempt coverage is selected for baseline as that of the IKN WSD). We define the correct results in these results as $\text{CorrectB}(\theta)$ to represent the comparable baseline WSD results. Then we can also get the precision for baseline results as $\text{CorrectB}(\theta)/\text{Attempt}(\theta)$. Now we can compare the precisions between the IKN and the baseline WSD algorithms. As shown in Table 1, the precision of our IKN WSD results are higher than baseline WSD results in different attempt coverage ranges.

Table 1 High precision result comparison of IKN WSD algorithm and baseline WSD algorithm

Threshold θ	Precision			Attempt Coverage
	IKN	Baseline	Improvement	
0.9	89.6%	86.3%	3.3%	31.06%
0.8	86.9%	84.1%	2.8%	38.22%
0.7	84.1%	79.2%	4.9%	46.35%
0.6	75.9%	73.7%	2.2%	60.46%
0.5	71.2%	69.2%	2.0%	73.05%

The IKN high-precision WSD results can be combined with any existing WSD algorithm through the following method. For each test word w_i , we define $s_{\text{IKN}}(w_i)$ as the result sense by IKN, $c_{\text{IKN}}(w_i)$ as the confidence of

disambiguation by IKN, $s_{\text{E}}(w_i)$ as the result sense by the existing WSD algorithm, and θ as the confidence threshold for IKN WSD. Then we can get the result sense $s_{\text{C}}(w_i)$ of w_i by the combined algorithm as

$$s_{\text{C}}(w_i) = \begin{cases} s_{\text{IKN}}(w_i) & \text{if } c_{\text{IKN}}(w_i) \geq \theta \\ s_{\text{E}}(w_i) & \text{if } c_{\text{IKN}}(w_i) < \theta \end{cases}$$

Table 2 Recall of combined algorithms of IKN and existing algorithms in Senseval-3 all words tasks

	Single	Combine with IKN		
		0.7	0.8	0.9
IKN Threshold θ	N/A	0.7	0.8	0.9
GAMBL	65.2%	65.2%	65.4%	65.4%
SenseLearner	64.6%	65.2%	65.1%	65.0%
Koc	64.1%	64.7%	64.7%	64.5%
R2D2	62.6%	63.8%	63.4%	63.2%
Meaning-allwords	62.4%	63.6%	63.6%	63.5%

Because the confidence value $c_{\text{IKN}}(w_i)$ of each word w_i is generated in our IKN WSD algorithm, the decision of result selection based on $c_{\text{IKN}}(w_i)$ is also made by the algorithm. So it is reasonable to consider the combined results as the results of the new combined algorithm. If our IKN WSD algorithm selects the results based on the imprecise confidence values, there is no guarantee that the combined result is always better than that generated by the existing algorithm.

Table 2 shows the performance of combined algorithms of IKN and the top five algorithms (Snyder and Palmer, 2004) in Senseval-3 all words tasks. Since the combined algorithms work for the full attempt coverage, we compare them with the existing algorithms by *recall*. When the confidence threshold reaches 0.8, each of the combined results is better than that of the corresponding single algorithm. This shows that IKN has better performance than existing algorithms in the set of test words with high disambiguation confidence. These results also show that high precision WSD methods could be used to improve the performance of existing algorithms.

5 Related Work and Discussion

Personalizing PageRank algorithm (Agirre and Soroa, 2009) is an iterative ranking process between word senses in the context. The basic principle is similar to IKN. This ranking algorithm is based on the semantic distance between concepts in a lexical knowledge base (LKB). However, their LKB is a type level network. The shortest semantic distance between two concepts has been fixed when the LKB was built. Although they extract sub-graphs for given input context, the shortest semantic distance between concepts is not changed. In other words, the shortest semantic distance is context free. The algorithms proposed by Mihalcea and Sinha (Mihalcea 2005; Sinha and Mihalcea 2007) are built on similarities between senses, which are also context free. SSI WSD algorithm (Navigli and Velardi 2005) is proposed on the basis of graph pattern matching. However, their graph patterns do

not contain the context sensitive syntactic feature either. Essentially, the graph matching between structural specifications of concepts is a measure of similarity between concepts too. Because the structural specifications are fixed for the concepts, the best matching structural specification is context free as well.

In IKN, we employ the maximum conditional probability of a sense synset from other sense synsets. Its effect is similar to the shortest semantic distance and the similarity in the above works. Different from LKB based algorithms and SSI, the conditional probabilities of IKN are kept at instance level. The maximum probability is determined by the graph matching between candidate graph and IGPs. Even containing the same two words, candidate graphs of different contexts may match different IGPs due to different syntactic structures. So the maximum conditional probability between sense synsets in IKN is syntactic context sensitive. This additional context relevance provides higher accuracy for WSD.

6 Conclusion and Future Works

In this paper, we have proposed a new instance knowledge network – IKN, and its graph matching algorithm. We have also developed the training algorithm for IKN and the probabilistic WSD algorithm using IKN. Our experimental study reveals reasonable performance of the IKN WSD algorithm. The high performance of combined algorithms of IKN with existing WSD algorithms shows that IKN based WSD can provide better results than the existing algorithms for the test words with high disambiguation confidence.

So far, we have conducted preliminary work on IKN based probabilistic WSD and there is plenty of room for improvement in the future. The dependency graphs we used at current stage are only for individual sentences due to the direct use of the output of the dependency parser. The IKN WSD algorithm will be beneficial from joining the sentence dependency graphs together by a high precision co-reference resolution algorithm. In addition, fuzzy graph matching may improve the attempt coverage. We will also study the semi-supervised learning algorithms on IKN.

7 References

- E. Agirre and A. Soroa. 2009 *Personalizing PageRank for Word Sense Disambiguation*. In EACL 2009
- M. Cuadros and G. Rigau. 2006. *Quality assessment of large scale knowledge resources*. In EMNLP 2006.
- M. Cuadros and G. Rigau. 2008. *KnowNet: Building a Large Net of Knowledge from the Web*. In COLING 2008.
- W. Daelemans, A. Van Den Bosch and J. Zavrel. 1999. *Forgetting exceptions is harmful in language learning*. Mach. Learn. 34(1) 1999.
- C. Fellbaum. 1998. *WordNet – an electronic lexical database*. MIT Press, 1998.
- D. Fernandez-Amoros. 2004. *WSD Based on Mutual Information and Syntactic Patterns* In ACL Senseval-3 Workshop 2004.
- D. Klein and C. Manning. 2002. *Fast Exact Inference with a Factored Model for Natural Language Parsing*. In NIPS 2002.
- D. Martinez, E. Agirre and L. Marquez. 2002. *Syntactic features for high precision word sense disambiguation*. In COLING 2002,.
- R. Mihalcea. 2005. *Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling*. In HLT2005
- R. Mihalcea and P. Edmonds, Eds. 2004. In ACL Senseval-3 Workshop 2004.
- H. Miller, C. Leacock, R. Teng and R. Bunker. 1993. *A semantic concordance*. In ARPA Workshop on HLT, 1993.
- R. Navigli. 2009 *Word sense disambiguation: A survey*. ACM Computing Surveys, 41(2), 2009.
- R. Navigli and M. Lapata. 2007. *Graph connectivity measures for unsupervised word sense disambiguation*. In IJCAI 2007.
- R. Navigli and P. Velardi. 2005. *Structural Semantic Interconnections: A Knowledge-Based Approach to Word Sense Disambiguation*. IEEE Trans. Pattern Anal. Mach. Intell, 27(7) 2005.
- H. Ng and H. Lee, 1996. *Integrating multiple knowledge sources to disambiguate word senses: An exemplar-based approach*. In ACL 1996.
- M. Palmer, C. Fellbaum, S. Cotton, L. Delfs, and H.T. Dang. 2001. *English tasks: All-words and verb lexical sample*. In SENSEVAL-2 Workshop 2001.
- J. Preiss. 2004. *Probabilistic word sense disambiguation*. Journal of Computer Speech and Language, 18(3) 2004.
- R. Sinha and R. Mihalcea. 2007. *Unsupervised graphbased word sense disambiguation using measures of word semantic similarity*. In ICSC 2007.
- B. Snyder and M. Palmer. 2004. *The English all-words task*. In ACL Senseval-3 Workshop 2004.
- Stanford_Parser.
<http://nlp.stanford.edu/software/lex-parser.shtml>
- D. Yarowsky and R. Florian, 2002. *Evaluating sense disambiguation across diverse parameter spaces*. J. Nat. Lang. Eng. 9(4), 2002.

Human Action Recognition Using Silhouette Histogram

Chaur-Heh Hsieh, *Ping S. Huang, and Ming-Da Tang

Department of Computer and Communication Engineering

Ming Chuan University

Taoyuan 333, Taiwan, ROC

*Department of Electronic Engineering

Ming Chuan University

Taoyuan 333, Taiwan, ROC

{hsiehch, pshuang}@mail.mcu.edu.tw, s7166076@ss24.mcu.edu.tw

Abstract

This paper presents a method for human action recognition using silhouette histogram. The human silhouette is obtained by using the background subtraction method and then mapped into three different polar coordinate systems that characterize three parts of a human figure respectively. Each polar coordinate system is quantized by partitioning it into several cells with different radii and angles. The evaluation applied at the Weizmann dataset shows that our method can achieve better performance than the existing silhouette-based system.

Keywords: silhouette, polar coordinate system, classifier.

1 Introduction

Human action recognition from videos is an important research area of computer vision. It can be applied to a variety of application domains, such as video surveillance, video retrieval and human-computer interaction systems. In the past, many human action recognition methods have been proposed and they are mainly divided into two kinds of approaches: shape-based and flow-based. The first method utilizes human silhouettes as features which are commonly extracted by background subtraction [1-5]. The second method adopts motion information from human movement such as optical flow [6-8]. Classification techniques that are usually used for feature matching consist of Nearest-Neighbor (N-N) [1, 6], Support Vector Machines (SVM) [9] and Hidden Markov Models (HMM) [2, 9, 10].

The silhouette-based method aims to recognize actions by characterizing the shapes of the actor's silhouette through space-time. The method is very popular since it is computationally efficient and robust to variations in clothing and lighting [11]. Nevertheless, it often suffers from the artifacts such as shadows, which will reduce the recognition accuracy. The flow-based method computes the optical field between adjacent frames and uses that as features for action recognition. This is suitable for recognizing small objects [6]. However, it is computational expensive, and the optical field is a very

coarse feature and thus different actions may exhibit similar flows over short periods of time [11].

Bobick and Davis [2] stack all silhouettes along the temporal axis from the sequence to form the Motion-History Image (MHI). MHI has a self-occlusion problem for complex movement and higher variation movement. Chen et al. [3] propose a method using star skeleton, which is obtained by connecting the centroid to gross extremities of a human contour. Chuang *et al.* [4] and Hsieh *et al.* [5] present a triangulation-base scheme for recognizing various human actions.

Hsiao et al. [1] propose the temporal-state shape context (TSSC) that can capture local space-time shape characteristics effectively. This can eliminate time warping effects in the local feature methods by segmenting the video sequence into several temporal states defined by soft fuzzy intervals. The distance between two action clips is defined as the summation of shape context distances in all frames. Besides estimating the distance measure between two actions, TSSC method can also provide spatial consistency information.

Motivated by the TSSC approach, this paper presents a human action recognition method by using three silhouette histograms. The human silhouette extracted by background subtraction is mapped into three different polar coordinate systems that characterize three parts of a human figure respectively. Each polar coordinate system is quantized by partitioning it into several cells with different radii and angles.

This paper is organized as follows. Section 2 introduces the proposed method. Experimental results and discussion are explained in Section 3. Finally conclusions are given in Section 4.

2 Proposed Method

2.1 Overview of Proposed System

The proposed system includes four main processes as shown in Figure 1. First, the human silhouette is extracted from the input video by background subtraction method [11]. Then, the extracted silhouette is mapped into three polar coordinate systems that characterize three parts of a human figure respectively. The largest circle covers the motion of the human body and the other two circles are to include the effect arms and legs have on the human action/silhouette. That is why two of the centres are between the shoulders and between the hips, respectively. Each polar coordinate system is quantized by partitioning

it into several cells with different radii and angles. By counting the number of pixels fallen into each cell from the silhouette at a particular frame, the silhouette histogram of the frame can be obtained. By collecting a sequence of silhouette histograms, a video clip is thus generated and used to describe the human action. Based on the silhouette histogram descriptor, an action classifier is trained and then used to recognize the action of an input video clip.

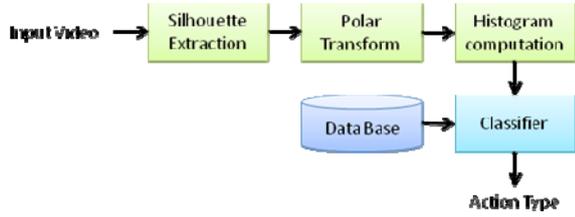


Figure 1: System Flow chart

2.2 Silhouette Extraction

We assume video sequences are captured from a still camera. Thus, background subtraction is applied to obtain the human silhouette from each frame. The well-known Gaussian Mixture Model (GMM) [11] is used to update the background. The method of background subtraction is given by

$$O_i(x, y) = \begin{cases} 1, & |F_i(x, y) - B_i(x, y)| > T \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where $B_i(x, y)$ is the background image, $F_i(x, y)$ is the input frame and T is the threshold value. Background refresh is achieved by

$$B_{i+1}(x, y) = (1 - \alpha) \cdot B_i(x, y) + \alpha \cdot F_i(x, y) \quad (2)$$

where $1 \geq \alpha \geq 0$.

2.3 Polar Transform

To be able to effectively describe the human shape, the Cartesian coordinate system is transformed into the polar coordinate system through the following equations:

$$r_i = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} \quad (3)$$

$$\theta_i = \tan^{-1} \left(\frac{y_i - y_c}{x_i - x_c} \right) \quad (4)$$

where (x_i, y_i) is the coordinate of silhouette pixels in the Cartesian coordinate system. (r_i, θ_i) is the radius and the angle in the polar coordinate system. (x_c, y_c) is the centre of the silhouette. The centre of the silhouette can be calculated by

$$x_c = \frac{1}{N} \sum_{i=1}^N x_i \quad (5)$$

$$y_c = \frac{1}{N} \sum_{i=1}^N y_i$$

where N is the total number of pixels.

The existing approaches [1, 4, 5] often use a single polar coordinate system to describe the human posture. However, our investigation indicates that the single coordinate is not enough to discriminate different postures with small difference. In this work, we design a method which contains three polar coordinate systems (three circles) defined as:

C1: Circle that encloses the whole human body.

C2: Circle that encloses the upper part of a body.

C3: circle that encloses the lower part of a body.

To verify the effectiveness of the proposed three-circle method, it is applied to describe two different actions in the Weizmann dataset: wave1 (one hand waving) and wave2 (two hands waving), as shown in Figure 2. Weizmann dataset is widely used by related research papers. The silhouette histograms obtained by C1 and C2 are shown in Figure 3 and Figure 4, respectively. We can see that two histograms from C1 are very similar such that the discriminability of action types is poor. On the contrary, superior discriminability is demonstrated by two histograms from C2.

2.4 Histogram Computation

To generate the histogram of each polar coordinate system, we partition each polar coordinate space into K cells by uniformly dividing each radius into n parts, and angles into m orientations such that $K = m \cdot n$. Therefore, K -bins are obtained for each histogram by

$$H(k) = \# \{ P_{\theta_i, r_j} \mid P_{\theta_i, r_j} \in \text{cell}_k \} \quad (6)$$

where the ranges of θ_i and r_j are defined by

$$\frac{2\pi i}{m} \leq \theta_i \leq \frac{2\pi(i+1)}{m}, \quad i = 0, 1, \dots, m-1$$

$$\text{and } \frac{j}{n} r_{\max} \leq r_j \leq \frac{(j+1)}{n} r_{\max}, \quad j = 0, 1, \dots, n-1.$$

(r_{\max} is the radius of circle C1, C2, or C3.)



Figure 2: the action of waving one hand (wave1) and waving two hand2(wave2)

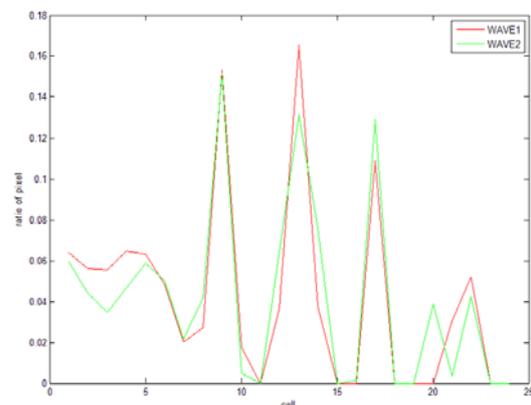
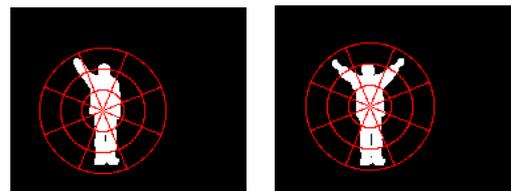


Figure 3: Silhouette histograms obtained by C1 for two waving actions.

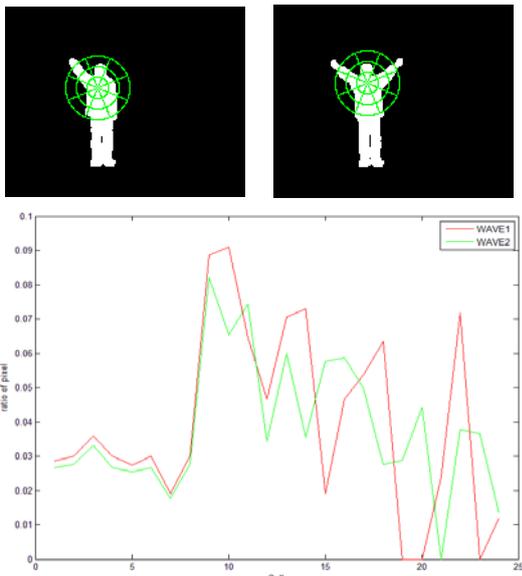


Figure 4: Silhouette histograms obtained by C2 for two waving actions.

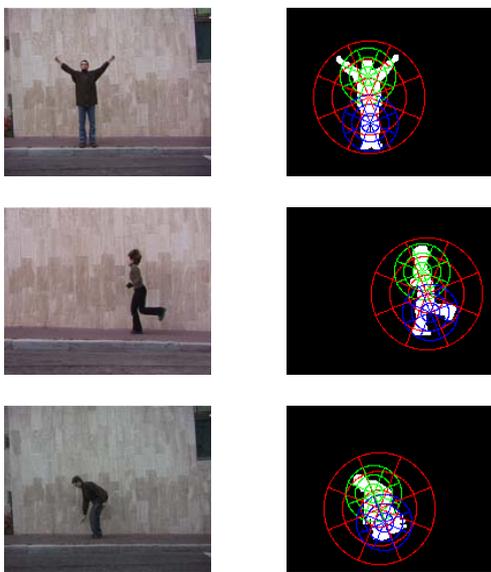


Figure 5. Human silhouettes of different actions and each is mapped onto three polar coordinate systems

Generally, the recognition accuracy is increased with the adding of the number of cells. However, it would also raise the time complexity. In our experiments, we chose $m=8$ and $n=3$, and a histogram with 24-bins is obtained for each polar coordinate system. The procedures for calculating the silhouette histogram can be organized into the following steps. First, we compute the centre of human silhouette and divide the silhouette into an upper part and a lower part according to the centre position. Then the centres of upper silhouette and lower silhouette are individually computed. Those three centre positions are taken as origins for the respective polar coordinate systems. Second, we compute the heights of all human silhouettes, which are used to calculate the radius of C1. The radius of C2 or C3 is half of C1 radius. The third step is to apply Equation (6) to compute three histograms separately for each human silhouette.

Figure 5 shows 3 frames of different actions and each is mapped onto three polar coordinate systems. Each histogram count is divided by the total number of pixels fallen into that coordinate space. The normalization is used to avoid the bias of different silhouettes. Finally, the normalized histograms are concatenated into a feature vector, which is used to describe the human posture at a particular time instant.

3 Experimental Results

Our algorithm is evaluated with Weizmann dataset that includes nine persons performing ten actions. Figure 6 shows 8 sample frames from this dataset with different actions. The result is compared to that of the TSSC method [1]. The video format is 180×144 with the frame rate of 25 fps. We adopt the nearest neighbor classifier to identify different action types. The leave-one-out [13] evaluation method is used, which means one clip of a video is used for testing, and the remaining clips of all videos are included into the training set. This process is repeated until all video clips are tested.

The video clip is a fixed-length video data, say ten frames. The video clips are generated by dividing a complete video into a designate number of clips. Neighbouring clips can be overlapped with a fixed length, called the overlapped length. In our experiments, we set the clip length as 10 frames, and the overlapped length as 5 frames. Totally, 961 clips are generated from 90 videos. Therefore, the feature dimension for each video clip which contains 10 frames is 720 ($24 \times 3 \times 10$). To improve the computation efficiency of classification, the principal component analysis (PCA) is adopted to reduce the dimension of a feature vector.

The recognition performance is evaluated by calculating confusion matrices shown in Table 1 and Table 2, which are generated from the TSSC method and our proposed method respectively. The results are obtained with a clip length of 10 and an overlapped length of 5.



Figure 6. Weizmann dataset

	Jump	Run	Skip	Walk	Side	Bend	Jack	Pjump	Wave1	Wave2
Jump	73	0	13	0	0	4	0	0	0	0
Run	0	62	3	0	1	0	0	0	0	0
Skip	1	0	54	0	0	0	0	0	0	0
Walk	0	0	0	113	1	0	1	0	0	0
Side	0	0	0	0	71	0	0	0	0	0
Bend	3	0	1	0	0	102	0	0	0	0
Jack	0	0	0	0	0	0	130	0	0	3
Pjump	0	0	1	0	1	0	0	93	0	5
Wave1	0	0	0	0	0	7	0	0	115	0
Wave2	0	0	0	0	0	0	0	0	0	103
Total: 961 clips							Error: 45		Accuracy: 95.31%	

Table 1: Confusion matrix of TSSC Method (N-N)

	Jump	Run	Skip	Walk	Side	Bend	Jack	Pjump	Wave1	Wave2
Jump	75	0	1	0	0	2	0	0	0	0
Run	0	60	1	0	0	0	0	0	0	0
Skip	0	0	69	0	0	0	0	0	0	0
Walk	0	2	0	113	0	0	0	0	0	0
Side	0	0	0	0	73	0	0	0	0	0
Bend	1	0	0	0	0	106	0	0	0	0
Jack	0	0	0	0	0	0	130	0	0	0
Pjump	1	0	0	0	1	1	1	93	0	0
Wave1	0	0	1	0	0	4	0	0	115	0
Wave2	0	0	0	0	0	0	0	0	0	111
Total:961 Error:16 Accuracy:98.33%										

Table 2: Confusion matrix of Our Method (N-N)

It can be seen that, for TSSC method, 45 clips are identified incorrectly, which achieves a recognition rate of 95.31%. However, our method can accomplish a higher recognition accuracy of 98.33%. Notice that the blend and skip actions have caused more errors for both methods.

Also, we have investigated the effects of overlapped length and clip length and the results are shown in Table 3 and Table 4, respectively. Table 3 indicates that the more the frames are overlapped, the better the recognition accuracy for both methods can achieve. The same conclusion is obtained for the clip length. The results listed in two tables show that our proposed method can achieve better recognition accuracy under various conditions of clip length and overlapped length.

Overlapped Frames	TSSC	Proposed method
7	97.96%	98.91%
5	95.31%	98.33%
3	92.90%	96.59%

Table 3: Recognition rates under different number of overlapped frames

Clip length	TSSC	Proposed method
10	95.31%	98.33%
15	97.35%	99.08%
20	98.59%	99.48%

Table 4: Recognition rates under different clip lengths

	Jump	Run	Skip	Walk	Side	Bend	Jack	Pjump	Wave1	Wave2
Jump	7	0	1	0	0	0	0	1	0	0
Run	0	5	2	1	0	0	0	1	0	0
Skip	0	0	4	0	0	0	0	0	0	0
Walk	0	3	0	8	0	0	0	0	0	0
Side	0	1	0	0	9	0	0	0	0	1
Bend	2	0	1	0	0	8	0	0	1	0
Jack	0	0	0	0	0	0	9	0	0	1
Pjump	0	0	1	0	0	0	0	6	1	0
Wave1	0	0	0	0	0	1	0	0	7	0
Wave2	0	0	0	0	0	0	0	1	0	7
Total:90 videos Error:20 Accuracy:77.77										

Table 5: Confusion matrix of TSSC Method (SVM)

	Jump	Run	Skip	Walk	Side	Bend	Jack	Pjump	Wave1	Wave2
Jump	9	0	0	0	0	0	0	0	0	0
Run	0	7	1	0	0	0	0	0	0	0
Skip	0	0	8	0	0	0	0	0	0	0
Walk	0	2	0	9	0	0	0	0	0	0
Side	0	0	0	0	9	0	0	1	0	0
Bend	0	0	0	0	0	8	0	0	0	0
Jack	0	0	0	0	0	0	9	0	0	0
Pjump	0	0	0	0	0	0	0	8	1	0
Wave1	0	0	0	0	0	1	0	0	8	1
Wave2	0	0	0	0	0	0	0	0	0	8
Total:90 videos Error:7 Accuracy:92.22										

Table 6: Confusion matrix of Our Method (SVM)

We argue that the leave-one-out test on clips is not very objective, because the parts of a testing video clip are included in the training set. Here, we design a test method called leave-n-out by taking one video for test and the remaining videos are used for training. Then n clips are extracted from the test video. Each video clip is evaluated separately, and the majority voting scheme is then applied to get the final decision. Table 5 is the TSSC result by using the leave-n-out method and Table 6 shows the result of our method. The results indicate that for TSSC the recognition accuracy is degraded significantly, while our method can still keep superior performance.

4 Conclusions

In this paper, we have developed an action recognition method which employs three polar coordinate systems to characterize different parts of the human posture. The human silhouette is mapped into a feature vector which contains three histograms. Experimental results show that our method can obtain superior performance than the TSSC method. Using the Weizmann dataset, the proposed method achieves a recognition rate of 98.33%. However, like the other silhouette-based approaches, the proposed method requires the extraction of a complete silhouette. Although the robustness to incomplete silhouettes has been raised by our method, the problem has not been completely solved. Furthermore, since the experiments are currently done within a well-controlled environment, more realistic outdoor scenes need to be tested. They will all be investigated in the future.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This research is supported in part by the National Science Council, Taiwan under the grant NSC 99-2632-E-130-001-MY3.

5 References

[1] Hsiao, P.C., Chen, C.S. and Chang L.W. (2008): Human action recognition using temporal-state shape context. *Proc. International Conference on Pattern Recognition*, 1-4.

[2] Bobick, F. and Davis, J.W. (2001): The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **23**: 257-267.

[3] Chen, H.S., Chen, H.T., Chen, Y.W. and Lee, S.Y. (2006): Human action recognition using star skeleton. *Proc. the 4th ACM international workshop on Video surveillance and sensor networks*, 171-178.

- [4] Chuang, C.C., Hsieh, J.W., Tsai, L.W. and Fan, K.C. (2008), "Human action recognition using star templates and delaunay triangulation. *Proc. International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 179-182.
- [5] Hsieh, J.W., Hsu, Y.T., Liao, H.Y. Mark, and Chen, C.C. (2008): Video-based human movement analysis and its application to surveillance systems," *IEEE Transactions on Multimedia*, **10**: 372-384.
- [6] Efros, A.A., Berg, A.C., Mori, G. and Malik J. (2003): Recognizing action at a distance. *Proc. Ninth IEEE International Conference on Computer Vision*, 726.
- [7] Liu, C., Yang, Y. and Chen Y. (2009): Human action recognition using sparse representation. *Proc. IEEE International Conference on Intelligent Computing and Intelligent Systems*, 184-188.
- [8] Liu, J., Ail, S. and Shah, M. (2008): Recognizing human action using multiple feature. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1-8.
- [9] Chang, C.C. and Lin, C.J. (2001): Software available at a library for support vector machines, <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [10] Petkovic, M., Jonker, W. and Zivkovic, Z. (2001): Recognizing strokes in tennis videos using Hidden Markov Models. *Proc. International Conference on Visualization, Imaging and Image Processing*, Marbella, Spain.
- [11] Ke, Y., Sukthankar, R. and Hebert, M. (2007): Spatio-temporal shape and flow correlation for action recognition. *Proc. Visual Surveillance Workshop*.
- [12] Benezeth, Y., Jodoin, P., Emile, B., Laurent, H. and Rosenberger, C. (2008): Review and evaluation of commonly-implemented background subtraction algorithms. *Proc. International Conference on Pattern Recognition*, 1-4.
- [13] Sun, X., Chen, M. and Hauptmann, A. (2009): Action recognition via local descriptors and holistic features. *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 58 - 65.

Performance Improvement of Vertical Handoff Algorithms for QoS Support over Heterogeneous Wireless Networks

Shusmita A. Sharna

Manzur Murshed

Gippsland School of Information Technology, Monash University
Churchill, Vic 3842, Australia

{Shusmita.Sharna, Manzur.Murshed}@monash.edu

Abstract

During the vertical handoff procedure, handoff decision is the most important step that affects the normal working of communication. An incorrect handoff decision or selection of a non-optimal network can result in undesirable effects such as higher costs, poor service experience, degrade the quality of service and even break off current communication. The objective of this paper is to determine the conditions under which vertical handoff should be performed in heterogeneous wireless networks. In this paper, we present a comprehensive analysis of different vertical handoff decision algorithms. To evaluate tradeoffs between their performance and efficiency, we propose two improved vertical handoff decision algorithm based on Markov Decision Process which are referred to as MDP_SAW and MDP_TOPSIS. The proposed mechanism assists the terminal in selecting the top candidate network and offer better available bandwidth so that user satisfaction is effectively maximized. In addition, our proposed method avoids unbeneficial handoffs in the wireless overlay networks.

Keywords: Heterogeneous Wireless Networks; Markov Decision Processes; Vertical Handoff.

1 Introduction

Nowadays many different types of networks communicate among themselves to form heterogeneous - networks. Due to different network access technologies, topologies, and implementations, one network might not be able to provide continuous coverage and required QoS parameters to a mobile user during an entire session. That is why handing over between different wireless networks appears as one of the fundamental solutions in today's heterogeneous wireless systems. Traditionally in a homogeneous network, handoff decision strategy is relatively simple based on received signal strength and network coverage. In heterogeneous networks, the problem is far complicated since handoff decision depends on various network quality-of-service (QoS) parameters (McNair and Zhu 2004). These new kind of handoff processes, used to rank and select among networks using different access technologies, are

categorically termed as *vertical handoff* (Stemm and Katz 1998).

Several interworking mechanisms have been proposed in recent literature to combine WLANs and cellular data networks into integrated wireless data environments. As mentioned by Zhang, the vertical handoff decision is formulated as a fuzzy MADM (Multiple Attribute Decision Making) problem. Two classical MADM methods are proposed: SAW (Simple Additive Weighting) and TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) (Zhang 2004). Tawil, Pujolle, and Salazar have introduced a novel Distributed Vertical Handoff Decision scheme (Tawil, Pujolle, and Salazar 2008). It places the calculation of the network quality at the candidate networks side instead of the mobile terminal side. Xia, Ling-ge, Chen, and Hong-wei mainly deal with a novel vertical handoff decision algorithm based on fuzzy logic with the aid of grey theory and dynamic weights adaptation (Xia, Ling-ge, Chen, and Hong-wei 2008). As mentioned by Stevens-Navarro, Lin, and Wong, the algorithm is based on the Markov Decision Process formulation with the objective of maximizing the expected total reward of a connection (Stevens-Navarro, Lin, and Wong 2008). Sun, Stevens-Navarro, and Wong also have proposed a vertical handoff decision algorithm for 4G wireless networks. The problem is formulated as a Constrained MDP (Sun, Stevens-Navarro, and Wong, 2008). As mentioned by Song and Jamalipour, the network selected is based on the Analytic Hierarchy Process (AHP) and Grey Relational Analysis (GRA) (Song and Jamalipour 2005). The AHP and GRA are also used for network selection in another work, where a mobile controlled three-step vertical handoff prediction algorithm is proposed (Kibria, Jamalipour, and Mirchandani 2005). Yang, Gondal, and Qiu have proposed a Multi-dimensional Adaptive SINR based Vertical Handoff algorithm, which uses the combined effects of SINR, user required bandwidth, user traffic cost and utilization from participating access networks to make handoff decisions for multi-attribute QoS consideration (Yang, Gondal, and Qiu 2008). Ormond, Murphy, and Muntean have proposed a utility-based strategy for network selection (Ormond, Murphy, and Muntean 2006). Liu, Li, Guo, and Dutkiewicz have discussed a hysteresis based and a dwelling-timer based algorithm (Liu, Li, Guo, and Dutkiewicz 2008). Shen and Zeng have proposed a cost-function-based network selection strategy in an integrated wireless and mobile network (Shen and Zeng 2008).

Although there have been various vertical handoff algorithms proposed in the literature, our goal is to introduce an efficient vertical handoff decision algorithm

Copyright (c) 2011, Australian Computer Society, Inc. This paper appeared at the Thirty-Fourth Australasian Computer Science Conference (ACSC2011), Perth, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 113. M. Reynolds, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.

by considering appropriate weights of each of the QoS parameters of candidate networks so that user satisfaction is effectively maximized. In order to assure required QoS for various applications and meanwhile avoid frequent handoffs in the heterogeneous systems, we integrate the analytic hierarchy process (AHP) (Alexander and Saaty 1989) and the Markov decision process (MDP) (Puterman 1994) on the network selection algorithms based on SAW (Simple Additive Weighting), and TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) to decide the best network for mobile users. We refer to these algorithms as MDP_SAW and MDP_TOPSIS. We have used both analytical and simulation tools ns-2.29 (*The Network Simulator ns-2*) to evaluate and compare expected total QoS offerings and the expected number of vertical handoff in the mean duration of a service under different states, discount factor and network switching costs. Numerical results show good performance improvement of our proposed scheme over SAW, TOPSIS, and MDP based algorithms.

The remainder of this paper is structured as follows. Section 2 provides the proposed model of vertical handoff decision algorithm. Section 3 shows simulation setup. The performance evaluation between different vertical handoff algorithms are presented in section 4. This paper is concluded in section 5.

2 Model Formulation

Assume that a heterogeneous wireless network comprises a single 3G network and a WLAN. We consider two QoS parameters, such as available bandwidth and delay for vertical handoff decision. Traditional SAW and TOPSIS use only the link reward function for network selection. Chosen network by SAW (Zhang 2004) and TOPSIS (Zhang 2004) for vertical handoff is the one which has the largest link reward function. In this paper we integrate the Analytic Hierarchy Process (AHP) (Alexander and Saaty 1989) and the Markov Decision Process (MDP) (Puterman 1994) on the traditional SAW, and TOPSIS to decide the best network for mobile users. We refer to these algorithms as MDP_SAW and MDP_TOPSIS.

In MDP_SAW and MDP_TOPSIS, the optimal network is selected by using *Value Iteration Algorithm* (VIA) (Puterman 1994).

Value Iteration Algorithm

- 1) Set $U^0(s) = 0$ for each state s . Specify $\epsilon > 0$, and set $k = 0$.
- 2) For each state s , compute $U^{k+1}(s)$ by $U^{k+1}(s) =$

$$\max_{a \in A} \{R(s, a) + \sum_{s' \in S} \lambda Pr[s'|s, a] U^k(s')\} \quad (1)$$

- 3) If $\|V^{k+1} - V^k\| < \epsilon(1 - \lambda)/2\lambda$, go to step 4. Otherwise, increase k by 1 and return to step 2.
- 4) For each $s \in S$, compute the stationary optimal policy $\pi(s)$.

$$\pi(s) = \arg \max_{a \in A} \left\{ R(s, a) + \sum_{s' \in S} \lambda Pr[s'|s, a] U^{k+1}(s') \right\}$$

and stop.

Chosen network by MDP_SAW and MDP_TOPSIS for vertical handoff is the one which has the optimal policy. The definitions of notations used in VIA followed by (Stevens-Navarro, Lin, and Wong 2008) are summarized below.

System states S are represented as a multi-dimensional vector. The state space S is defined as:

$$S = N \times B_1 \times D_1 \times B_2 \times D_2 \times \dots \times B_N \times D_N$$

where B and D denotes the available bandwidth and delay. To reduce the number elements in the state space, we have considered large unit values so that each parameter space is divided into 15 discrete levels.

$A = \{a_1, a_2, \dots, a_n\}$ represents the all possible action set where N denotes the total number of collocated networks in the coverage area of interest. $U(s)$ stands for expected total reward. The reward function $R(s, a)$ of MDP_SAW and MDP_TOPSIS can be augmented to consider the effect of switching cost as

$$R(s, a) = f(s, a) - K(s, a) \quad (2)$$

where $K(s, a)$ is the signalling cost function,

$$K(s, a) = \begin{cases} K, & i \neq a \\ 0, & i = a \end{cases} \quad (3)$$

and, link reward function $f(s, a)$ is equal to $f_{SAW}(s, a)$ and $f_{TOPSIS}(s, a)$ given by (4) and (5), respectively for MDP_SAW and MDP_TOPSIS using current state s and the chosen action a .

$$f_{SAW}(s, a) = \omega_b f_{b_SAW}(s, a) + \omega_d f_{d_SAW}(s, a) \quad (4)$$

$$f_{TOPSIS}(s, a) = I^-(s, a) / (I^-(s, a) + I^+(s, a)) \quad (5)$$

where ω_b and ω_d are the weight of bandwidth and delay, respectively. In (4) $f_{b_SAW}(s, a)$ and $f_{d_SAW}(s, a)$ are the bandwidth and delay reward function for MDP_SAW and are defined as follows:

$$f_{b_SAW}(s, a) = b_a / b_{\max} \quad (6)$$

$$f_{d_SAW}(s, a) = d^{\min} / d_a \quad (7)$$

where b_a and d_a are the bandwidth and delay, respectively of the network selected by the action a , and b^{\max} and d^{\min} are the maximum bandwidth and the minimum delay, respectively, among all the networks.

In (5) $I^+(s, a)$ and $I^-(s, a)$ are close to ideal solution and far from ideal solution, respectively and are calculated using the formula given below.

$$I^+(s, a) = \sqrt{(f_{b_TOPSIS} - f_{b_TOPSIS}^{\max})^2 + (f_{d_TOPSIS} - f_{d_TOPSIS}^{\min})^2}$$

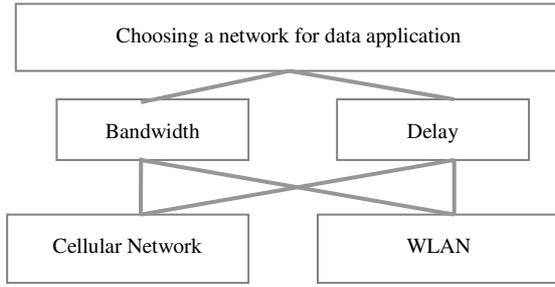


Figure 2: AHP hierarchy establishment

$$I^-(s, a) = \frac{1}{\sqrt{(f_{b_TOPSIS} - f_{b_TOPSIS}^{min})^2 + (f_{d_TOPSIS} - f_{d_TOPSIS}^{max})^2}}$$

where $f_{b_TOPSIS}(s, a)$ and $f_{d_TOPSIS}(s, a)$ are the bandwidth and delay reward function for MDP_TOPSIS and are defined as follows:

$$f_{b_TOPSIS}(s, a) = \frac{b_a}{\sqrt{\sum b^2}} \times \omega_b \quad (8)$$

$$f_{d_TOPSIS}(s, a) = \frac{d_a}{\sqrt{\sum d^2}} \times \omega_d \quad (9)$$

where ω_b and ω_d are the weight of bandwidth and delay, respectively and b_a and d_a are the bandwidth and delay, respectively of the network selected by the action a .

This model also introduces decision epoch E . At each decision epoch, the mobile terminal has to decide whether the connection should use the current chosen network or be rerouted to another network (i.e., execute a vertical handoff). The number of epochs before reaching equilibrium is distributed with mean $1/(1 - \lambda)$ where $0 \leq \lambda < 1$. When expected service duration is known, λ is set accordingly to match the equivalent number of epochs. For example, we have put $\lambda = 0.96$ to model mean service time of 25 epochs. Finally, given that the current state s and the chosen action is a , the probability function that the next state, s' is given by:

$$Pr[s'|s, a] = \begin{cases} \prod_{n=1}^N Pr[b_n', d_n' | b_n, d_n], & j = a \\ 0, & j \neq a \end{cases}$$

We also integrate Analytic Hierarchy Process (AHP) (Alexander and Saaty 1989) in the proposed scheme to derive the weights of each QoS parameters based on user's preference. The AHP is based on the structuring a problem in a hierarchical form. The hierarchy is structured on different levels. The overall goal is the first level of the hierarchy. The decision factors are presented in the intermediate level. The solution alternatives are located at the lowest level. For instance, a mobile unit is trying to make a selection among two networks cellular network and WLAN. The preferences are bandwidth and delay. The hierarchy on "choosing a network" is established as shown in Fig. 1. In second step the objectives are compared with each other in order to determine their

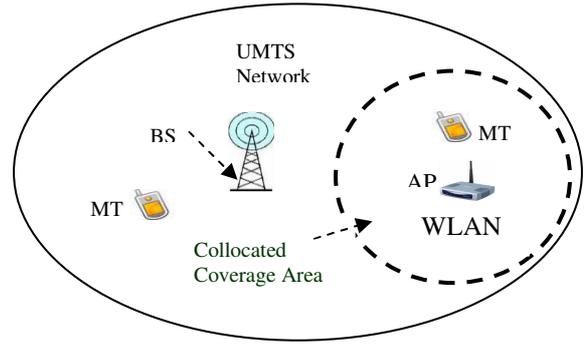


Figure 1: Collocated heterogeneous wireless networks.

relative importance by using fundamental 1 to 9 scales. The numbers from 1 to 9 are used to respectively present equally, weakly moderately, moderately, moderately plus, strongly, strongly plus, very strongly, very strongly, and extremely important to the objective. In the last step, the overall weights of the factors are achieved by computing the values of AHP matrix. For example, we fixed BW to 1 and vary delay from 1 to 9. i.e. 1_1 denote bandwidth and delay is equally important to the objective and 1_2 denote bandwidth and delay is weakly moderately important to the objective and so on. Hence 1_1 to 1_9 scale denote different weight level.

3 Simulation Setup

We consider a scenario of collocated networks (i.e., $n=2$) as shown in fig. 2 which is composed of a WLAN and a cellular system. WLAN is considered as network 1 and cellular network is 2. The application is assumed to be voice (i.e., conversational). To evaluate the state transition probability function of the WLAN, a typical IEEE 802.11b WLAN is simulated using ns-2(2.29) (*The Network Simulator ns-2*); where users arrive and depart from the network according to an exponential distribution with an average inter arrival time 1.6 s. The basic rate and data rate of the WLAN are 1 Mbps and 11 Mbps, respectively. The counting of transitions among states is performed to estimate the state transition probabilities. For the state transition probability function of the wireless cellular system, the values of bandwidth and delay are assumed to be guaranteed for the duration of the connection (Stevens-Navarro, Lin, and Wong 2008). Thus

$$Pr[b_2', d_2' | b_2, d_2] = \begin{cases} 1, & b_2' = b_2, d_2' = d_2 \\ 0, & \text{otherwise} \end{cases}$$

The performance metrics that we consider in our experiment are the expected total bandwidth and the expected number of vertical handoff. Here we consider two QoS parameters, bandwidth and delay. At first reward function of each QoS parameter is defined for each algorithm using (6) and (7) for SAW and (8) and (9) for TOPSIS. Then these functions are used to determine the corresponding link reward function, $f_{SAW}(s, a)$ and $f_{TOPSIS}(s, a)$ for each of the candidate networks. For SAW and TOPSIS selected network is chosen from the

Simulation Parameters	Values
Decision epochs, E	25
Average time between successive decision epochs	15s
Discount Factor, λ	0.96
Switching cost from network 1 to network 2, $K_{1,2}$	0.25
Switching cost from network 2 to network 1, $K_{2,1}$	0.25
Maximum available bandwidth	15 unit
Maximum available delay	15 unit
Unit of bandwidth	335kbps
Unit of delay	35 ms
ϵ	10^{-3}
Relative importance between bandwidth and delay	1.4
Weight factor for Bandwidth using AHP	0.2
Weight factor for Delay using AHP	0.8

Table 1: Simulation Parameters.

network which provides highest link reward. For MDP_SAW and MDP_TOPSIS a switching cost is included with link reward function of SAW and TOPSIS, respectively to derive reward function of MDP_SAW and MDP_TOPSIS. MDP also include switching cost function with their own link reward function (Stevens-Navarro, Lin, and Wong 2008).

Finally candidate network for MDP_SAW, and MDP_TOPSIS is the one, which gives highest total whereas MDP_SAW and MDP_TOPSIS use VIA for obtaining optimal network. Based on network selection our further step is to determine expected total bandwidth and expected number of vertical handoff for each algorithms using VIA by solving equations (10) and (11), respectively.

$$B(s) = \{b + \sum_{s' \in S} \lambda Pr[s'|s, a]B(s')\} \quad (10)$$

$$V(s) = \{v + \sum_{s' \in S} \lambda Pr[s'|s, a]V(s')\} \quad (11)$$

where b and v are the bandwidth unit (1 to 15) and number of vertical handoff (0 or 1), respectively of the corresponding optimal network. For solving (10), we replace (10) with (1) and $B(s)$ with $U(s)$ and follow steps 1 to 3 in VIA (section 2). Thus we get expected total bandwidth $B(s)$ after final iteration. Same steps are followed for (11) to solve it.

The parameters of the experiment used in the numerical results are summarized in Table 1. Our proposed method MDP_SAW and MDP_TOPSIS is tested against individual SAW, TOPSIS, and MDP and tested with different aspects. At first the performance matrices are investigated with respect to discount factor, λ . Next we took into account the influence of switching cost, K on the expected total bandwidth and expected number of vertical handoff. We do not consider expected total reward because it does not translate to optimal QoS offerings at different user perception levels which have been presented in our previous works (Sharna and Murshed 2010). Furthermore we provide the result of performance metrics with different weight factor. The weight factor considers here is the scale of AHP from 1_1 to 1_9. Finally we see the variation of expected number of vertical handoff on state space.

4 Experiment Result

With the discount factor increasing from 0.94 to 0.98 MDP_SAW and MDP_TOPSIS provide more expected total bandwidth than SAW, TOPSIS, and MDP as shown in fig. 3. The initial state vector at the beginning of connection is assumed to be $\{2, 5, 5, 7, 8\}$. When λ is varied from 0.94 to 0.98, it corresponds to the variation of the average connection duration from 4 to 12.5 min. We also observed that when λ increases expected total bandwidth is also increases and MDP_SAW provides highest values for all values of λ .

Fig. 4 shows the variation of λ versus expected number of vertical handoff where MDP_SAW shows significant improvement because it provides far lower values of vertical handoff than SAW, TOPSIS, and MDP. Fewer handoffs indicate better handoff algorithm because it can avoid ping-pong effect.

Fig. 5 and fig. 6 show the expected total bandwidth and expected number of vertical handoff versus switching cost. The discount factor is 0.96. MDP_SAW gives better result in both cases. That means higher expected total bandwidth and lower expected number of vertical handoff.

In fig. 5 MDP perform better than MDP_TOPSIS but there is sudden degradation of performance at switching cost 1. On the other hand SAW and TOPSIS choose the candidate network based on QoS parameter and do not consider switching cost. Thus expected values are constant for those algorithms.

Fig. 7 and fig. 8 show the variation versus different weight level. Here weight 1_1 denote bandwidth and delay is equally important to the objective and 1_2 denote bandwidth and delay is weakly moderately important to the objective and so on. In both cases MDP_SAW provide far better results than all other algorithms.

Fig. 9 shows the expected number of vertical handoff varies in state space. Here state vector $\{2, 14, 2, 7, 8\}$ to $\{2, 15, 15, 7, 8\}$ has been consider for space limitation. Thus their corresponding integer values are 421 to 449. Expected number of vertical handoff is varied in different state space. This is because different state has different optimal policy for vertical handoff. Beyond them MDP_SAW gives lowest values than all other algorithms.

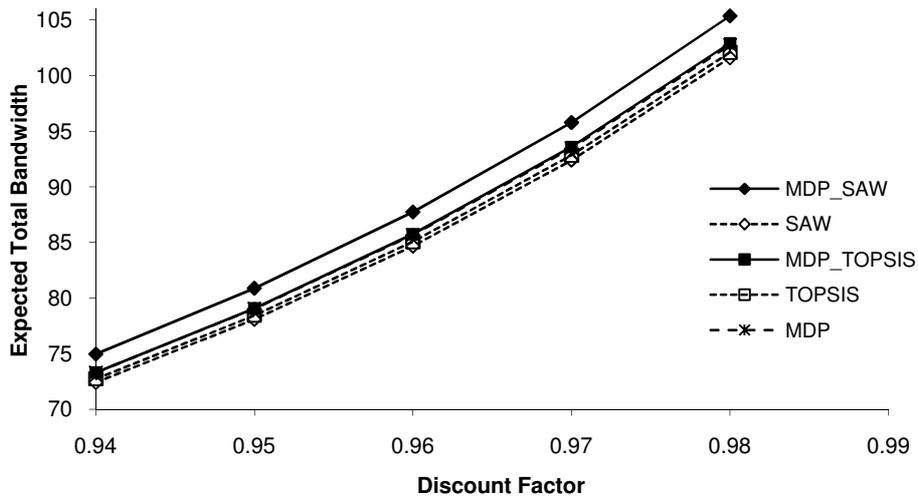


Figure 3: Effect of expected total bandwidth under different discount factor, λ in state {2, 5, 5, 7, 8}.

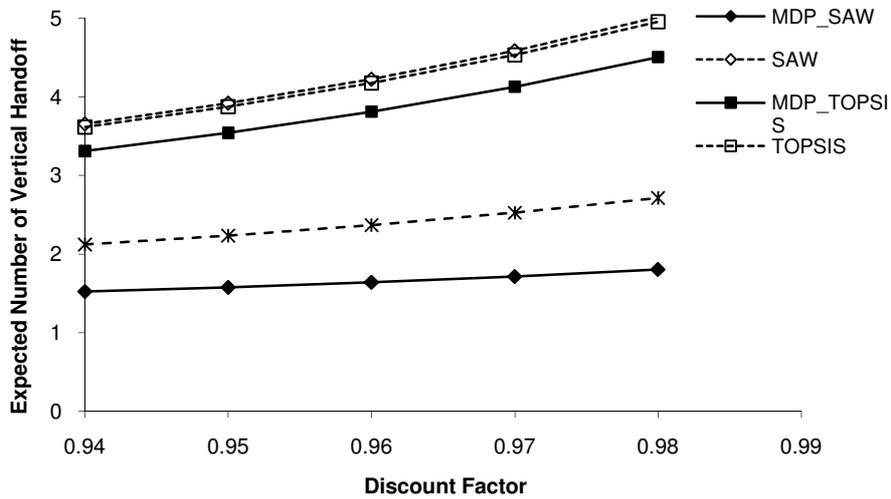


Figure 4: Effect of expected number of vertical handoff under different discount factor, λ in state {2, 5, 5, 7, 8}.

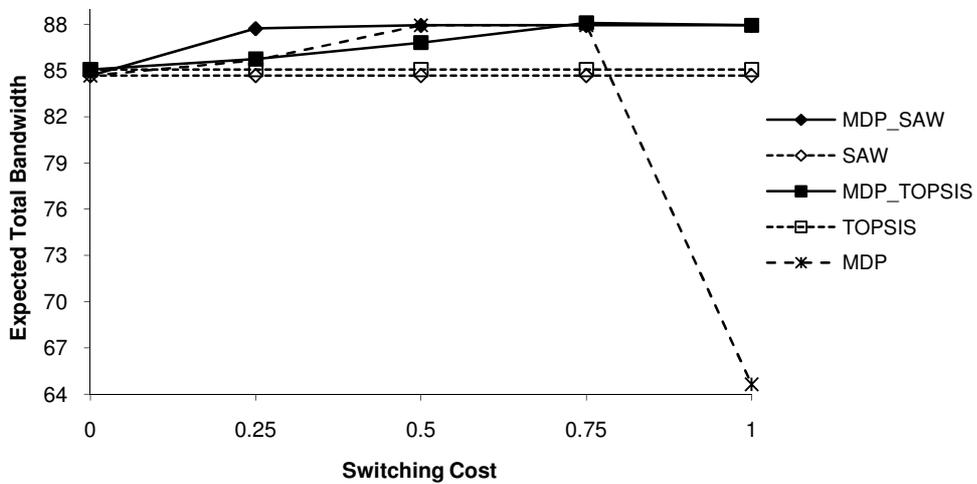


Figure 5: Effect of expected total bandwidth under different switching cost, K in state {2, 5, 5, 7, 8}.

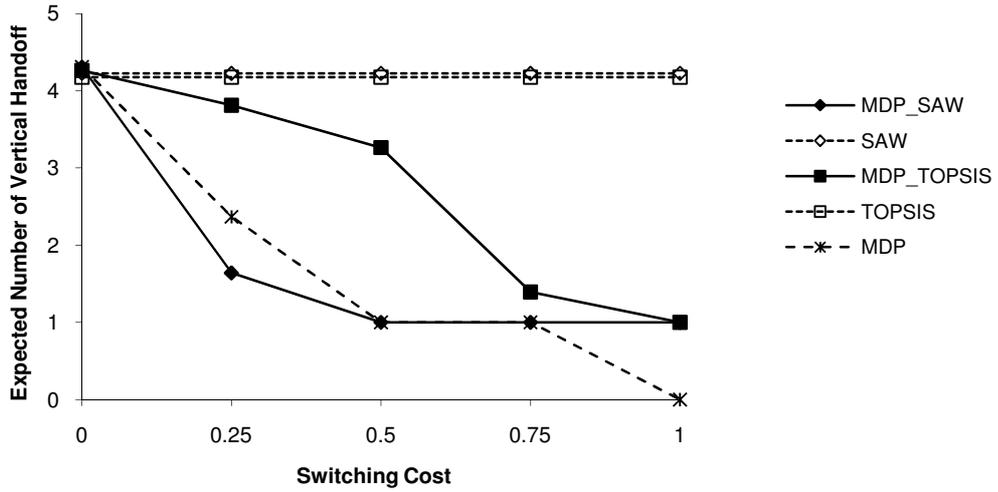


Figure 6: Effect of expected number of vertical handoff under different switching cost, K in state $\{2, 5, 5, 7, 8\}$.

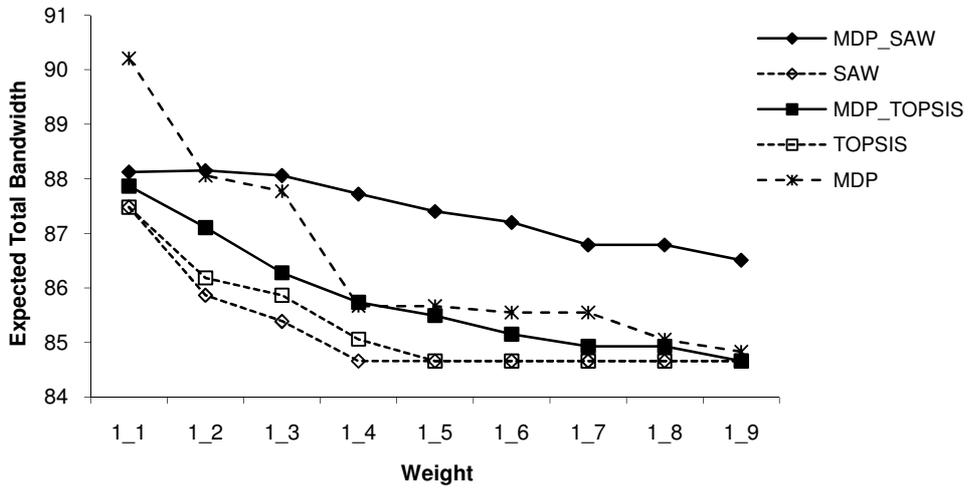


Figure 7: Effect of expected total bandwidth under different weight factor in state $\{2, 5, 5, 7, 8\}$.

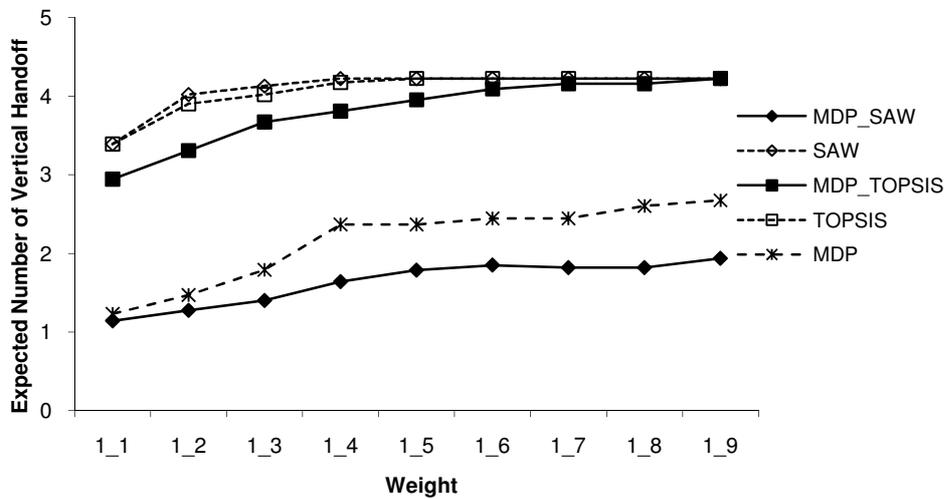


Figure 8: Effect of expected number of vertical handoff under different weight factor in state $\{2, 5, 5, 7, 8\}$.

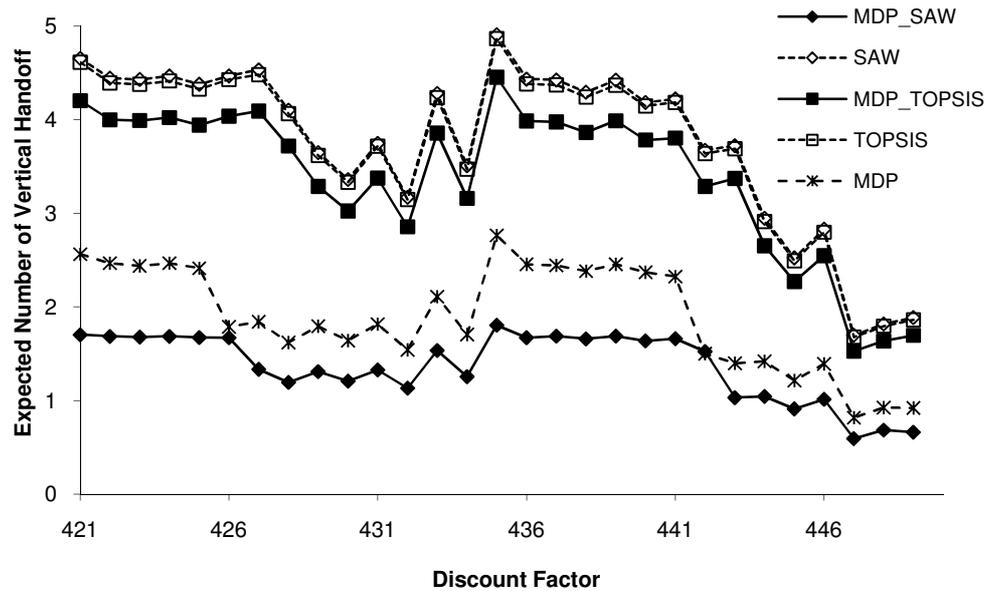


Figure 9: Effect of expected number of vertical handoff under states {2, 14, 2, 7, 8} to {2, 15, 15, 7, 8}.

5 Conclusion

We have presented a novel algorithm for vertical handover which integrate the Markov decision process (MDP) on traditional algorithms SAW and TOPSIS. Although we propose both MDP_SAW and MDP_TOPSIS, MDP_SAW outperforms all other algorithms. Numerical results show that our proposed MDP_SAW gives a higher expected total bandwidth. In addition from our experiment, it is observed that policy from MDP_SAW gives the lower expected number of vertical handoff per connection. The ping-pong effect occurs when a mobile node moves around the overlay area between two networks, causing unnecessary handoffs and increasing the handoff overhead. Fewer handoffs indicate a better handoff algorithm. Numerical results also revealed interesting behavioural patterns for the expected number of vertical handoff as it varied at different states of mobile user. This means that it not only depend on the current network but also the QoS offered by the candidate network, which can be changed at each decision epoch. This model could be successfully used for designing, evaluating and optimizing cost effective handoff mechanisms in wireless overlay networking environments.

6 References

- McNair, J. and Zhu, F. (2004): Vertical Handoffs in Fourth-generation Multinetwork Environments. *IEEE Wireless Communications*, 11(3): 8-15.
- Stemm, M. and Katz, R. (1998): Vertical Handoffs in Wireless Overlay Networks. *ACM Mobile Networks (MONET) Special Issue on Mobile Networking in the Internet*, 3(4): 335-350.
- Zhang, W. (2004): Handover decision using fuzzy MADM in heterogeneous networks. *Proc. IEEE WCNC, Atlanta, GA*, 653-658.
- Tawil, R., Pujolle, G. and Salazar, O. (2008): A Vertical Handoff Decision Scheme In Heterogeneous Wireless Systems. *IEEE Vehicular Technology Conference*, 2626-2630.
- Xia, L., Ling-ge, J., Chen H. and Hong-wei, L. (2008): An Intelligent Vertical Handoff Algorithm In Heterogeneous Wireless Networks. *International Conference on Neural Networks and Signal Processing*, 550-555.
- Stevens-Navarro, E., Lin, Y. and Wong, V.W.S. (2008): An MDP-based vertical handoff decision algorithm for heterogeneous wireless network. *IEEE Transaction of Vehicular Technology*, 57(2): 1243-1254.
- Sun, C., Stevens-Navarro, E. and Wong, V.W.S. (2008): A Constrained MDP-based Vertical Handoff Decision Algorithm for 4G Wireless Networks. *Proc. IEEE ICC*, 2169-2174.
- Song, Q. and Jamalipour, A. (2005): A network selection mechanism for next generation networks. *Proc. IEEE ICC, Seoul, Korea*, 1418-1422.
- Kibria, M. R., Jamalipour, A. and Mirchandani, V. (2005): A location aware three-step vertical handoff scheme for 4G/B3G networks. *Proc. IEEE GLOBECOM, St. Louis, MO*, 2752-2756.
- Yang, K., Gondal, I. and Qiu, B. (2008): Multi-Dimensional Adaptive SINR Based Vertical Handoff for Heterogeneous Wireless Networks. *IEEE Communications Letters*, 12(6): 438-440.
- Ormond, O., Murphy, J. and Muntean, G. (2006): Utility-based intelligent network selection in beyond 3G systems. *Proc. IEEE ICC, Istanbul, Turkey*, 1831-1836.
- Liu, M., Li, Z., Guo, X. and Dutkiewicz, E. (2008): Performance Analysis and Optimization of Handoff

- Algorithms in Heterogeneous Wireless Networks. *IEEE Trans. on Mobile Computing*, 7(7): 846-857.
- Shen, W. and Zeng, Q. (2008): Cost-Function-Based Network Selection Strategy in Integrated Wireless and Mobile Networks. *IEEE Trans. on Vehicular Technology*. 57(6): 3778-3788.
- Alexander, J. M. and Saaty, T. L. (1989): *Conflict Resolution – The Analytic Hierarchy Proces*. Praeger, NY.
- Puterman, M. (1994): *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ: Wiley.
- The Network Simulator ns-2*. [Online]. Available: <http://www.isi.edu/nsnam/ns>.
- Sharna, S. A. and Murshed, M. (2010): Performance Analysis of Vertical Handoff Algorithms with QoS Parameter Differentiation. Accepted in proceedings *HPCC*.

Resource Provisioning based on Lease Preemption in InterGrid

Mohsen Amini Salehi Bahman Javadi Rajkumar Buyya

Cloud Computing and Distributed Systems (CLOUDS) Laboratory
 Department of Computer Science and Software Engineering
 The University of Melbourne, Australia
 Email: {mohsena, bahmanj, raj}@csse.unimelb.edu.au

Abstract

Resource provisioning is one of the main challenges in resource sharing environments such as InterGrid. Recently, many resource management systems in resource sharing environments use lease abstraction and virtual machines for provisioning. In resource sharing environments resource providers serve requests from external (grid) users along with their own local users. The problem arises when there is not sufficient resources for local users, which have higher priority than grid users, and need resources urgently. This problem could be solved by preempting leases from grid users and allocating them to the local users. However, preempting leases entails determining which lease(s) are better choices to be preempted and what should be done with the preempted leases. To answer these questions, in this work, we propose different request types in the InterGrid environment. Then, we propose and compare several policies that determine the proper set of lease(s) for preemption. The first policy increases resource utilization as a system centric criterion. The second policy improves user satisfaction by decreasing the number of preempted leases. The third policy makes a trade-off between resource utilization and the number of lease preemption. Simulation results demonstrate that the proposed preemption policies serve up to 72% more local requests without increasing the rejection ratio of grid requests.

1 Introduction

Managing and providing computational resources for user applications is one of the challenges in the high performance computing community. Resource sharing environments enable sharing, selection, and aggregation of different resources across several Resource Providers (RP), which are also called sites, and usually scattered over a geographical region. These RPs are connected through high bandwidth network connections. Nowadays, heavy computational requirements, mostly from scientific communities, are supplied by these RPs, such as Grid 5000 in France and DAS-2 in the Netherlands.

InterGrid (De Assunção et al. 2008), provides an architecture and policies for inter-connecting different Grids. As shown in Figure 1, in InterGrid computational resources in each RP are shared between grid users as well as local users. The provisioning rights

over the resources from several RPs inside a Grid are delegated to the InterGrid Gateway (IGG). IGGs coordinate resource allocation for requests through pre-defined contracts between Grids (De Assunção et al. 2008). On the other hand, local users send their requests directly to the local scheduler of the RP.

Hence, resource provisioning in InterGrid is done for two different types of users, namely: local users and external (grid) users. As illustrated in Figure 1, local users (hereafter termed local request), refer to users who ask their local RP for resources. Grid users (hereafter termed grid request) are those users who send their requests to the IGG to get access to larger amount of resources. Typically, for an RP local requests have more priority than grid requests (Chase et al. 2003). However, removing the contention between the local request and grid request is challenging. In other words, the organization that owns the resources would like to ensure that its community has priority access to the resources. In this circumstance, grid requests are welcome to use resources if they are available. Nonetheless, grid requests should not delay the execution of local requests.

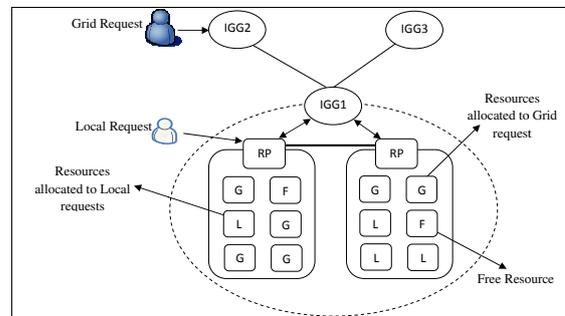


Figure 1: A scenario that shows the contention between local and grid requests in InterGrid.

In InterGrid the resource provisioning is based on the *lease* abstraction. A lease is an agreement between resource provider and resource consumer whereby the provider agrees to allocate resources to the consumer according to the lease terms presented by the consumer (Sotomayor et al. 2008, 2009). Virtual Machine (VM) technology is a way to implement lease-based resource provisioning (Sotomayor et al. 2009). The capabilities of VMs in getting suspended, resumed, stopped, or even migrated (when there is enough bandwidth) have been extensively studied and have shown to be useful in resource provisioning without major utilization loss (Chase et al. 2003, Sotomayor et al. 2008, Zhao & Figueiredo 2007). InterGrid makes one lease for each user VM request.

In this paper, we solve the problem of contention between local and grid requests in InterGrid. Given the fact that local requests have more priority rather than local requests we preempt grid leases in favor

Copyright ©2011, Australian Computer Society, Inc. This paper appeared at the 32nd Australasian Computer Science Conference (ACSC 2011), Perth, Australia, January 2011. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 113, Mark Reynolds, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

of local requests. More specifically, we propose three policies to determine an appropriate set of leases for preemption. Finally, we make an appropriate decision for the preempted lease.

The rest of this paper is organized as follows: In Section 2, the problem we are investigating is discussed, then in Section 3 related research works are introduced. Proposed policies are described in Section 4 and detailed evaluations are mentioned in Section 5. Finally, conclusion and future works are provided in Section 6.

2 Problem Statement

As mentioned earlier, local requests have higher priority than grid requests in InterGrid. The problem we are dealing with in this paper is resource provisioning for local requests when existing resources have been allocated to grid requests and the rest of resources are not adequate to serve the local requests. In this situation, one solution is to preempt some of the leases allocated to grid users. However, there are some challenges in preempting leases.

The first challenge is that there are some restrictions in preempting leases. In fact, one difference between job-based resource provisioning and lease-based resource provisioning is that jobs can be preempted without notifying the user (job owner). Nevertheless, this is not the case for leases (Grit et al. 2007). Therefore, the first challenge is coming up with regulations in the lease terms to make lease preemption possible.

Moreover, making a proper decision for a preempted lease, given the characteristics of a resource sharing environment such as InterGrid, is challenging.

Since lease preemption has some side-effects, the second challenge we consider is how to minimize these side-effects. These side-effects and the challenge ahead are discussed over the next paragraphs.

In the case of job-based resource provisioning, many distributed systems do not provide the ability of preempting jobs (Snell et al. 2002). This is mainly because the operating system has to provide the security of not accessing files and data of the preempted processes. Additionally, since the operating systems mostly do not provide the checkpointing facilities, the preempted jobs have to be killed, which is a waste of resources (Snell et al. 2002). These problems are obviated by leveraging VMs in lease-based resource provisioning (Sotomayor et al. 2008). However, preempting leases and subsequently VMs is not free of cost and imposes time overhead to the system which is one of the side-effects of preemption.

Since the next time-slots are already reserved for other requests, preempting leases and allocating to new requests can potentially affect these reservations. Re-scheduling preempted leases and affected reservations are also side-effects of preempting leases. Preempting leases also makes grid users wait for a longer time to get their leases completed.

By getting lease preemption possible in an RP, there is a possibility that several leases have to be preempted to make sufficient vacant resources for a local request. Therefore, there are potentially several sets of candidate leases that can be preempted. In this paper each set of the candidate leases is termed a *Candidate Set*. However, selecting different candidate sets affects the amount of imposed overhead as well as the number of grid users who get affected by preemption.

These issues bring another challenge into the picture. The challenge is choosing the optimal candidate set for preemption in a way that minimizes the side-effects of preempting leases.

In summary, there are two main challenges ahead:

1. How to make lease preemption possible and what are the potential decisions for preempted leases?
2. Which candidate set is a better choice for preemption? (Preemption policy)
3. We evaluate the impact of the proposed policies upon key performance metrics in a simulation environment.

Formal definition of the problem can be stated as follows:

- L_i : Lease i .
- R_j : Local request j
- $\tau(L_i) \in \{gridCancelable, gridSuspendable, gridMigratable, gridNonPreemptable, localNonPreemptable\}$
- $v(x_i)$: Number of VMs in the lease/request i .
- $h(L_i)$: Overhead of preempting lease i .
- $p(L_i)$: Category of lease i (local or grid) and defined as follows:

$$p(L_i) = \begin{cases} 1 & \text{if grid request} \\ 0 & \text{if local request} \end{cases}$$

According to the above definitions, candidate set m for allocating request R_j can be presented as follows:

$$C_m : \{\forall L_i | p(L_i) = 1 \ \& \ v(R_j) \leq \sum_{i=1}^{N_m} v(L_i)\} \quad (1)$$

where N_m is the number of leases involved in candidate set C_m . If there are S candidate sets, then all candidate sets can be peresented as:

$$A : \{C_m | 0 \leq m \leq S - 1\} \quad (2)$$

Finally, a preemption policy can be presented as a function that selects an appropriate candidate set out of all candidate sets (i.e., $policy(A) = C_m$).

We believe that in an RP with many nodes and requests, selecting a proper candidate set for preemption is crucial and leads to significant reduction in preemption overhead time and increases user satisfaction. Although the problem we are investigating in this paper is in InterGrid context, it could be also applied to other lease-based Grid/Cloud resource providers where requests with higher priority (such as local or organisational requests) coexist with other requests.

3 Related Work

Although preemption was not extensively studied in distributed computing previously (Snell et al. 2002), recently many researches have been undertaken in the area.

Haizea (Sotomayor et al. 2008) is a lease scheduler which schedules advanced reservation and best effort leases. Haizea preempts best effort leases in favor of advance reservation requests. In case of preempting a lease, Haizea just considers the preemptability of the lease. In other words, when there are several candidate sets to be preempted, Haizea does not have any specific policy to determine which candidate set is a better choice for preemption. In contrast, we propose and compare policies that determine which candidate set is a better choice for preemption. In Haizea, preempted leases are suspended and put in the queue to

be resumed in another available time-slot later. However, we consider a diversity of leases (Cancelable, Suspendable, Migratable, and non-preemptable) that gives the scheduler more options than just suspending the lease. Sotomayor et al. (Sotomayor et al. 2008, 2009) have directly mentioned that lease requests should be categorized and decided based on user-provided priorities. Therefore, our work can be considered as complementary for the research undertaken by Sotomayor et al..

In another research undertaken by Sotomayor et al. (Sotomayor et al. 2009), the overhead time imposed by preempting a lease (suspending and resuming a VM) is estimated. We use the same model to consider the overhead in our evaluations. This overhead should be taken into consideration in lease scheduling where some leases should be preempted in favor of other requests. The proposed model is based on the amount of memory that should be de-allocated, number of VMs mapped to each physical node, local or global memory used for allocating VMs, and the delay related to commands being enacted. In evaluation of the proposed preempting policies, we consider the overhead involved in preempting leases based on this model.

Walters et al. (Walters et al. 2008) introduced a preemption-based scheduling policy for batch and interactive jobs inside a cluster. In this work batch jobs are preempted in favor of interactive jobs. The authors introduce different challenges in preempting jobs including selecting a proper job to be preempted, how to checkpoint the preempted job, how to provision VMs, and how to resume the preempted job. Their preemption policy is based on weighted summation of several factors such as the time spent in the queue.

One difference of our work with Walters et al. is that our preemption policy is based on lease based resource provisioning, while Walter’s research is based on job. Moreover, Walters et al. do not consider the circumstance that several jobs should be preempted to make room for the higher priority jobs. Another difference is that Walters et al. do not consider the impact of preemption on reservations in the queue. By contrast, our work considers both the running leases as well as reservations in the queue.

Kettimuthu et al. (Kettimuthu et al. 2005) focused on the impact of preempting parallel jobs in supercomputers for improving the average and worst case slow down of jobs. The authors also suggested a preemption policy, which is called Selective Suspension, where an idle job can preempt a running job if the suspension factor is adequately more than running job.

However, the authors do not specify which job should be preempted instead they decide when to do the preemption. The proposed policy is starvation free since it operates based on the response ratio of jobs.

Isard et al. (Isard et al. 2009) investigate the problem of optimal scheduling for data intensive applications such as Map-Reduce (Isard et al. 2009) on the clusters that computing and storage resources are close together. This work provides an example of Cancelable leases that can be terminated without any notification to the job owner. Achieving the optimal resource allocation, they propose a scheduling policy that preempts the currently running job in order to maintain data locality for a new job. Although the scheduling policy is based on job preemption, the authors do not discuss which job is selected to be preempted amongst several candidates.

Snell et al. (Snell et al. 2002) consider the impact of preemption on backfilling scheduling. They provide policies to select the set of jobs for preemption

in a way that the jobs with higher priority jobs are satisfied and at the same time the utilization of resources increase. In this work the preempted job is restarted and rescheduled in the next available time-slot. Our work is different with Snell et al. from several aspects. Firstly, since we consider lease based resource provisioning, there are more choices for the preempted lease (such as suspended, migrated, etc). Secondly, Snell et al. recognize the best set of running jobs for preemption. However, in our contribution the preemption policy considers the best candidate set as well as the impact of preempting on the reservations made for the leases in the queue. The third difference is that Snell et al. does not consider the overhead of preempting jobs. In fact, by killing the preempted jobs they reduced the overhead to zero. Nonetheless, the computational power is wasted in that case.

4 Proposed Solution

4.1 Introducing Different Lease Types

To tackle challenges mentioned in Section 2, we should first make the preemption possible in lease terms agreed between resource provider and consumer. For that purpose, we introduce different request types in InterGrid. After reservation is done for a request, the “request type” is mapped to “lease type”.

At the moment, a request issued by a user in InterGrid is composed of the following characteristics:

- Virtual Machine (VM) name needed by the user.
- Number of VMs needed.
- Ready time: the time that requested VMs should be ready.
- Wall time: duration of the lease.
- Deadline: the time that serving the request must be finished.

We extend the InterGrid request by adding the “request type” to it.

Considering the characteristics of a resource sharing environment, proposed request (lease) types give more choices to the scheduler rather than just suspending and rescheduling the preempted leases (Sotomayor et al. 2008). Based on the lease type, the scheduler determines how to schedule the lease and what to be done with a preempted lease.

Different request types we consider for requests in InterGrid are broadly classified as best effort and deadline constraint requests. More details of different request types are as follows:

- *Best Effort-Cancelable*: these requests can be scheduled at any time after their ready time. Leases of such type can be canceled without notifying the lease owner. Cancelable leases neither guarantee the deadline nor the wall time of the lease. Such leases are suitable for map-reduce-like requests (Isard et al. 2009). Spot instances in Amazon EC2¹ are also another example of Cancelable leases. Cancelable leases impose the minimum overhead time at the time of preemption. This overhead is related to the time needed to terminate VMs allocated to the lease.
- *Best Effort-Suspendable*: leases of this type can be suspended at any time but should be resumed later. This type of lease guarantees the wall time of the lease but not in a specific deadline.

¹<http://aws.amazon.com/ec2/spot-instances>

Suspendable leases are flexible in start time and they can be scheduled at any time after their ready time. In the case of preemption, these leases should be rescheduled to find another free time-slot for the remainder of their execution. The overhead time of preempting a Suspendable lease is sum of the time needed to suspend a VM, reschedule the lease, and resume it later. Suspendable leases are suitable for Bag-of-task (BOT) and Parameter Sweep type of applications (Buyya et al. 2005).

- *Restartable* (Redirectable): In this case the preempted lease can be canceled and restarted in another Grid later on. In InterGrid IGGs can redirect requests to other Grids through peering adjustments (De Assunção et al. 2008). Restartable requests can be either best-effort or deadline-constraint. In the former case, the wall time of the request and in the latter both wall time and deadline of grid request is guaranteed. However, we do not consider this type of grid leases in our preemption policies and leave this choice for the future work.
- *Deadline Constraint-Migratable*: These leases guarantee both the wall time and deadline of the lease. However, there is no guarantee that they will be run on a specific resource(s). In other words, there is always a chance for the lease to be preempted but it will be resumed and finished before its deadline, either on the same resource or on another resource. Nonetheless, migrating VMs involves VM transferring overhead. One solution to mitigate this overhead is migrating to another RP inside the same Grid of InterGrid which has a high bandwidth connection. Multiple reservation is also a conceivable strategy to serve such kind of leases (Sotomayor et al. 2008). We leave the details of migration issues involved as a future work. Migratable requests are needed by steerable applications (Costanzo et al. 2009). In these applications, which are already implemented in InterGrid, the workload can be migrated to more powerful sites to meet user constraints such as deadline (Costanzo et al. 2009).
- *Deadline Constraint-Non-Preemptable*: The leases associated with such requests cannot be preempted at all. These leases guarantee both deadline and wall time without being preempted during the lease. This type of lease is useful for critical tasks in workflows where some tasks have to start and finish at exact times to prevent delaying the execution of the workflow (Kwok & Ahmad 1996).

We assume that local requests are all deadline constraint non-preemptable. However, grid users can send all request types mentioned above. Different lease types correspond to different prices. Thus, users are motivated to associate their requests to different request types. Unarguably, the more flexible request type the less expensive the lease. However, we leave the market-oriented implications of the lease-based scheduling as a future work.

4.2 Preemption Policies

In this section we discuss policies for choosing the best candidate set for preemption.

As mentioned earlier, the scheduler in this system faces with two types of requests: grid requests and local requests.

If the scheduler receives a non-preemptable or Migratable grid request, the scheduler must determine

whether there are adequate free resources available for the requested wall time from the requested start time. If the request is found not to be possible, it will be rejected. Nonetheless, there is more flexibility for Suspendable and Cancelable requests. In fact, since such requests are not restricted to any specific deadline, the scheduler tries to find a vacant place for the requested wall time in any available time-slot.

If the scheduler cannot find any vacant resource for a local request, then the scheduler tries to make room for the local request by preempting the Suspendable, Cancelable, and/or Migratable leases that coincide with the local request's needed interval. Thus, leases that their preemption makes enough space for the local request are selected and form all candidate sets. Each candidate set contains a set of leases that their preemption makes enough space for an incoming local request. However, if resources are occupied by non-preemptable grid leases or leases from other local requests, then the local request inevitably gets rejected. Preemption policy in this situation determines the proper candidate set for preemption.

Choosing different candidate sets affects the number of VMs to be preempted, which turns out to present different time overheads. On the other hand, selecting different candidate sets leads to a different number of leases to be preempted, which adds more waiting time and, consequently, more grid user dissatisfaction.

We propose three preemption policies that result in different candidate sets to be preempted. They lead to a different amount of time overhead and a different number of leases to get preempted. One policy focuses on system centric criteria by trying to increase resource utilization. The second policy focuses on user centric criteria and tries to preempt fewer leases to make more user satisfaction. The third policy makes a trade-off between resource utilization and user satisfaction.

4.2.1 Minimum Overhead Policy (MOV)

As a system centric policy, this policy aims at maximizing resource utilization. Therefore, this policy tries to minimize the time overhead imposed to the underlying system by preempting a candidate set that leads to the minimum overhead. For this purpose the total overhead imposed to the system by each candidate set is calculated and a set with minimum overhead is selected. According to the notation we defined in Section 2, MOV policy can be presented based on Equation 3.

$$MOV(A) = \min_{m=0}^{S-1} \{h(C_m)\} \quad (3)$$

The overhead time imposed by each candidate set varies based on the type of leases involved in that candidate set (Sotomayor et al. 2009). For *Cancelable* leases the overhead is the time needed to terminate the lease and shut down its VM(s). This time is usually much lower than the time needed for Suspending or Migrating leases (Sotomayor et al. 2009).

The overhead imposed by preempting a *Suspendable* lease includes the time needed to suspend and resume VM(s) plus a time for re-scheduling the remaining time of the lease. The estimation of these times has been carried out by Sotomayor et al. (Sotomayor et al. 2009). We use the same method to work out the overhead time imposed by preempting Suspendable leases. Therefore, the overhead of suspension is $t_s = \frac{mem}{s}$ and resumption time is $t_r = \frac{mem}{r}$. Where *mem* is the amount of VM memory, *s* is the rate of suspending megabytes of VM memory per second,

and r is the rate of re-allocating megabytes of VM memory per second (Sotomayor et al. 2009). Preempting a *Migratable* lease enforces VM(s) transferring overhead in addition to the overheads mentioned for Suspendable leases (Zhao & Figueiredo 2007).

Since we consider a global file system, if there are several VMs in a lease (i.e., K), then the preemption overhead time is multiplied by K (Sotomayor et al. 2009).

4.2.2 Minimum Leases Involved Policy (MLIP)

Users do not like that their leases get affected by preemption. In fact, preempting leases makes longer waiting times for Suspendable and Migratable leases to get completed. In the case of Cancelable leases, preempting the lease results in terminating the lease. Therefore, as a user centric policy, MLIP tries to satisfy more users by preempting fewer leases.

In this policy a candidate set that contains minimum number of leases is selected from all the candidate sets. MLIP disregards the type of leases involved in a candidate set during decision making. Based on the notation introduced in Section 2, MLIP can be presented according to Equation 4.

$$MLIP(A) = \min_{m=0}^{S-1} \{|C_m|\} \quad (4)$$

where $|C_m|$ gives the number of leases involved (cardinality) in each candidate set C_m .

4.2.3 Minimum Overhead Minimum Lease Policy (MOML)

The two proposed policies mentioned earlier aim to either improve resource utilization (as a system centric criterion) or minimize the number of preempted leases (as a user centric criteria). However, in MOML we propose an approach that can fulfill both system and user centric criteria at the same time. This policy is depicted in Figure 2 and elaborated in Algorithm 1. In fact, MOML is a balance between MOV, which minimizes the imposed overhead, and MLIP, which attempts to minimize the number of requests affected by preemption.

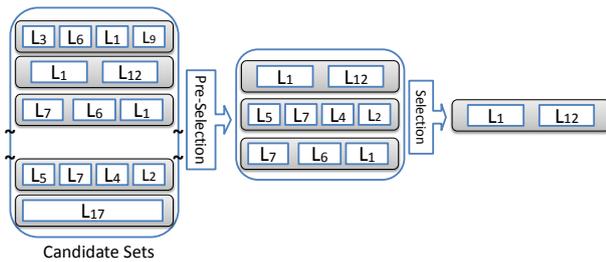


Figure 2: Pre-selection and final selection steps of MOML policy.

According to Figure 2 and Algorithm 1, in MOML the selection of a candidate set is carried out in two steps. In the first step (pre-selection) all candidate sets which have a total overhead less than a certain threshold (α) are pre-selected for the second step (lines 5 to 8 in Algorithm 1). The first step increases the tolerance of acceptable overhead in comparing with MOV. In the second step, to have fewer leases affected, a candidate set that contains minimum number of leases is selected (lines 9 to 11 in Algorithm 1).

Selecting a proper value for α determines the behaviour of MOML policy. More specifically, if the

$\alpha \rightarrow \infty$, then MOML behaves the same as MLIP. On the other hand, if $\alpha \rightarrow 0$, then MOML behaves the same as MOV. Thus, to keep the trade-off between MOV and MLIP, we consider α as the *median* value of the overheads (lines 1, 2, and 4 in Algorithm 1). By choosing $\alpha = \text{median}$ we ensure that just half of the candidate sets that have lower overheads are considered in the second step for having a minimum number of leases.

Algorithm 1: MOML Preemption Policy.

```

Input: Candidate Sets
Output: Selected Candidate Set
1 foreach candidateSet  $\in$  Candidate Sets do
2   Overheads.Add(getOverhead(candidateSet));
3 min  $\leftarrow$   $\infty$ ;
4  $\alpha$   $\leftarrow$  getMedian(Overheads);
5 foreach candidateSet  $\in$  Candidate Sets do
6   ovhd  $\leftarrow$  getOverhead(candidateSet);
7   NoLeases  $\leftarrow$  Cardinality(candidateSet);
8   if ovhd  $\leq$   $\alpha$  then
9     if NoLeases  $<$  min then
10      selected  $\leftarrow$  candidateSet;
11      min  $\leftarrow$  NoLeases;
    
```

5 Performance Evaluation

In this section we discuss different performance metrics considered, the scenario in which the experiments are carried out, and experimental results obtained from simulation.

5.1 Performance Metrics

Introducing different types of leases along with preemption policy are expected to affect different parameters, which are described over the next subsections.

5.1.1 Local and Grid Request Rejection Rate

The initial goal of this research is serving more local requests by preempting resources from grid leases. Therefore, as a result of our research, it is interesting for us to find out how efficient these preemption policies are in terms of serving more local requests.

The “local request rejection rate” is the fraction of local requests which are rejected, possibly because of allocating resources to non-preemptable grid requests or other local requests.

Additionally, we are interested to see if decreasing local request rejection rate comes with the cost of rejecting more grid requests. Grid Request Rejection Rate describes this metric and shows the percentage of grid requests that are rejected. The ideal case is that local request rejection rate is reduced without increasing the grid request rejection rate.

5.1.2 Resource Utilization

Time overhead is a side-effect of preempting leases that results in resource under-utilization. Therefore, we are interested to see how different preemption policies affect the resource utilization.

Resource utilization is defined according to the Equation 5.

$$Utilization = \frac{computationTime}{totalTime} * 100 \quad (5)$$

Table 1: Detailed specifications of the generated workloads. $|BE|$ stands for the number of best effort grid requests, $|DC|$ stands for the number of deadline constraint grid requests, and $|Local|$ stands for the number of local request.

Modified Parameter	Distribution	Other Constant Parameters
Average No. VMs	Two-stage uniform	$ BE = 1000$, $ DC = 1000$, $ Local = 1000$
No. BE Requests	Uniform	Average No. VMs=4, $ DC = 2000 - BE $, $ Local = 1000$
No. DC Requests	Uniform	Average No. VMs=4, $ BE = 2000 - DC $, $ Local = 330$
No. Local Requests	Uniform	Average No. VMs=4, $ BE = DC = (totalRequest - local)/2$

Where:

$$computationTime = \sum_{i=1}^{|\lambda|} v(l_i) \cdot d(l_i) \quad (6)$$

Where $|\lambda|$ is the number of leases, $v(l_i)$ is the number of VMs in lease l_i , $d(l_i)$ is the wall time of lease l_i .

5.1.3 Number of Lease Preemption

Preempting grid leases has different impacts on different lease types. For Suspendable and Migratable leases, preemption leads to increasing completion time. For Cancelable leases preemption results in terminating that lease. Since users of different lease types have distinct expectation from the system, it is not easy to propose a common criterion to measure user satisfaction. For instance, owners of Migratable leases expect to get their accepted and meet the deadline while owners of Suspendable leases like to have short waiting times. Nonetheless, in all types of leases grid users suffer from lease preemption. Therefore, to have a generic metric to measure the user satisfaction, we are interested to see the total number of preemptions resulted by different policies.

5.2 Experimental Setup

We used a discrete event simulator to evaluate the performance of the preemption policies. These preemption policies are implemented in the context of InterGrid. In the experiments conducted, Lublin99 (Lublin & Feitelson 2001) has been configured to generate 3000 parallel jobs.

Lublin99 is the workload model based on the San Diego Super Computer (SDSC) Blue Horizon machine. Job traces collected from this supercomputer are publicly available and have been studied extensively in the past.

We consider a cluster with 32 nodes as an RP. We assume all nodes of the RP as single core with one VM. The maximum number of VM(s) in generated requests is also 32.

We consider each VM of 1024 MB and a 100 Mbps network bandwidth. Hence, according to Section 4.2.1, in our experiments, suspension time (t_s) and resumption time (t_r) are 161 and 126 seconds respectively. The time overhead for migrating a VM with similar configuration is 160 seconds (Zhao & Figueiredo 2007).

We intend to study the behavior of different policies when they face workloads with different characteristics. More specifically, we study situations where workloads have:

- different number of VMs needed.
- different number of best effort grid requests (including Cancelable and Suspendable).

- different number of deadline constraint grid requests (including Migratable and Non-Preemptable).
- different number of local requests.

Each experiment is carried out on each of these workloads separately. To generate these workloads, we modify parameters of Lublin99's model. The way these workloads are generated and other detailed specifications of these workloads are described in Table 1.

5.3 Experimental Results

5.3.1 Local and Grid Request Rejection Rate

The primary goal of this paper is to show the impact of preempting grid leases to allocated resources to local requests. In Table 2, the mean difference of decrease in local requests rejection rate is reported along with a 95% confidence interval of the difference. We report the difference between rejection rate in two situations; First, when no preemption policy is used and second, when MOML is used as a preemption policy. Since all proposed preemption policies resulted in similar local and grid rejection rate we have just reported the result for MOML. We use a T-test to work out the mean difference between these two policies. To perform the T-test we have ensured that the distribution of difference is normal.

According to Table 2, local request rejection rate has statistically and practically significantly decreased by applying preemption in all cases. More importantly, this reduction in local request rejection rate has not been with the cost of rejecting more grid requests. Based on Table 2, it can be noted that in all experiments grid request rejection rate does not change significantly.

Based on this experiment, the maximum decrease in local request rejection rate occurs when the percentage of best effort grid requests is higher (second row in Table 2). In this circumstance, more local requests can be accommodated by preempting these best effort leases.

5.3.2 Resource Utilization

In this experiment we measure the resources utilization when different preemption policies are applied. As illustrated in all sub-figures of Figure 3, we explore the impact of altering workload parameters pointed out in Table 1 on resource utilization when different preemption policies are applied.

This experiment indicates that resource utilization increases almost linearly by increasing the average number of VMs in requests (Figure 3(a)). Although preempting best effort leases make some overhead, we can see in Figure 3(b) that increasing the number of best effort requests improves resource utilization;

Table 2: Mean difference and 95% confidence interval (CI) of decrease in local requests rejection rate and grid requests rejection rate as a result of applying preempting leases in an RP of InterGrid.

Modified Parameter	Mean Decrease in Local Requests Rejection Rate	CI of Decrease in Local Requests Rejection Rate	Change in Grid Requests Rejection Rate
Average No. of VMs	55.1%	(47.2,62.9), P-Value<0.001	Equal
Percentage of BE Grid Requests	72.0%	(51.1,92.8), P-Value=0.001	Not statistically significant, P-Value=0.6
Percentage of DC Grid Requests	54.3%	(35.0,73.7), P-Value=0.001	Not statistically significant, P-Value=0.3
Percentage of Local Requests	58.2%	(40.3,75.9), P-Value<0.001	Not statistically significant, P-Value=0.6

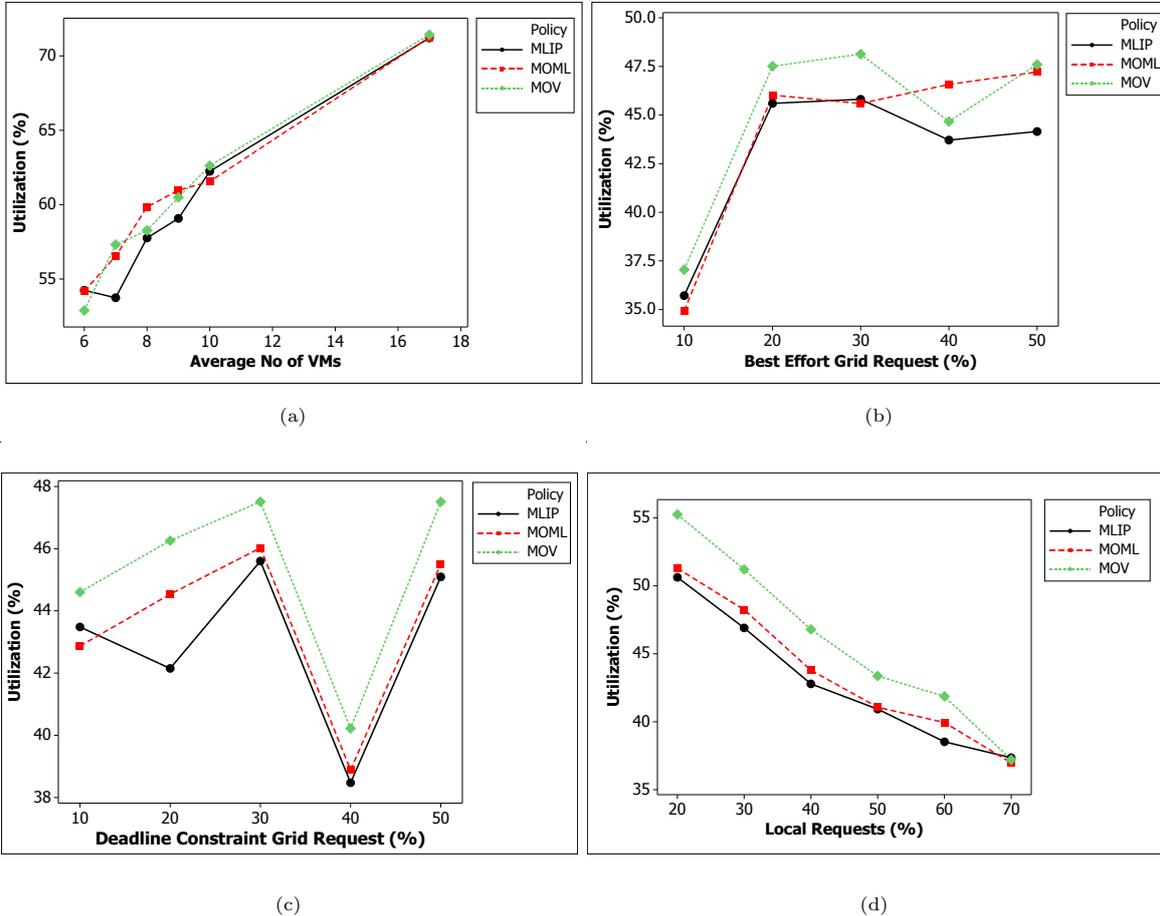


Figure 3: Resource utilization results from different policies. This experiment was carried out by modifying (a) the average number of VMs, (b) the percentage of best effort grid requests, (c) the percentage of deadline constraint grid requests, and (d) the number of local requests.

however, after a certain percent (best effort > 20%) resource utilization does not fluctuate significantly in different policies. The reason is that, allocating other (unused) time-slots to the preempted leases result in less resource fragmentation and therefore resource utilization does not decrease by increasing the percentage of the grids best effort requests.

In Figure 3(c), we can see that resource utilization increases by increasing the percentage of deadline constraint requests in all policies. However, there is a sharp decrease (from around 45% down to 38%) when 40% of requests are deadline constraint. In fact, in this point there are many best effort and deadline constraint requests in the system at the same time. Hence, more preemption occurs and subsequently more overhead is imposed to the system. We can conclude that the system would result in minimum utilization when 40% of requests are deadline constraint and the rest are best effort.

By increasing the number of local requests the number of preemption and subsequently the amount of overhead increases. Therefore, as we can see in

Figure 3(d), by increasing the number of local requests, resource utilization decreases almost linearly in all policies.

In all sub-figures of Figure 3 it can be observed that MOV result in better utilization comparing with the other policies. However, in a few points (e.g., 40 in Figure 3(b)), MOV has slightly less utilization than MOML. This happens mainly because of the resource fragments that occur at scheduling time which leads to resource under-utilization. Sub-figures of Figure 3 also demonstrates that resource utilization MOML lies between MLIP and MOV.

5.3.3 Number of Lease Preemptions

In this experiment we investigate the number of grid leases that get preempted by applying different preemption policies.

From Figure 4(a) we can infer that, in general, larger leases (i.e. leases with more number of VMs) lead to fewer preemptions. In fact, in this situation fewer of leases are needed to be preempted to make

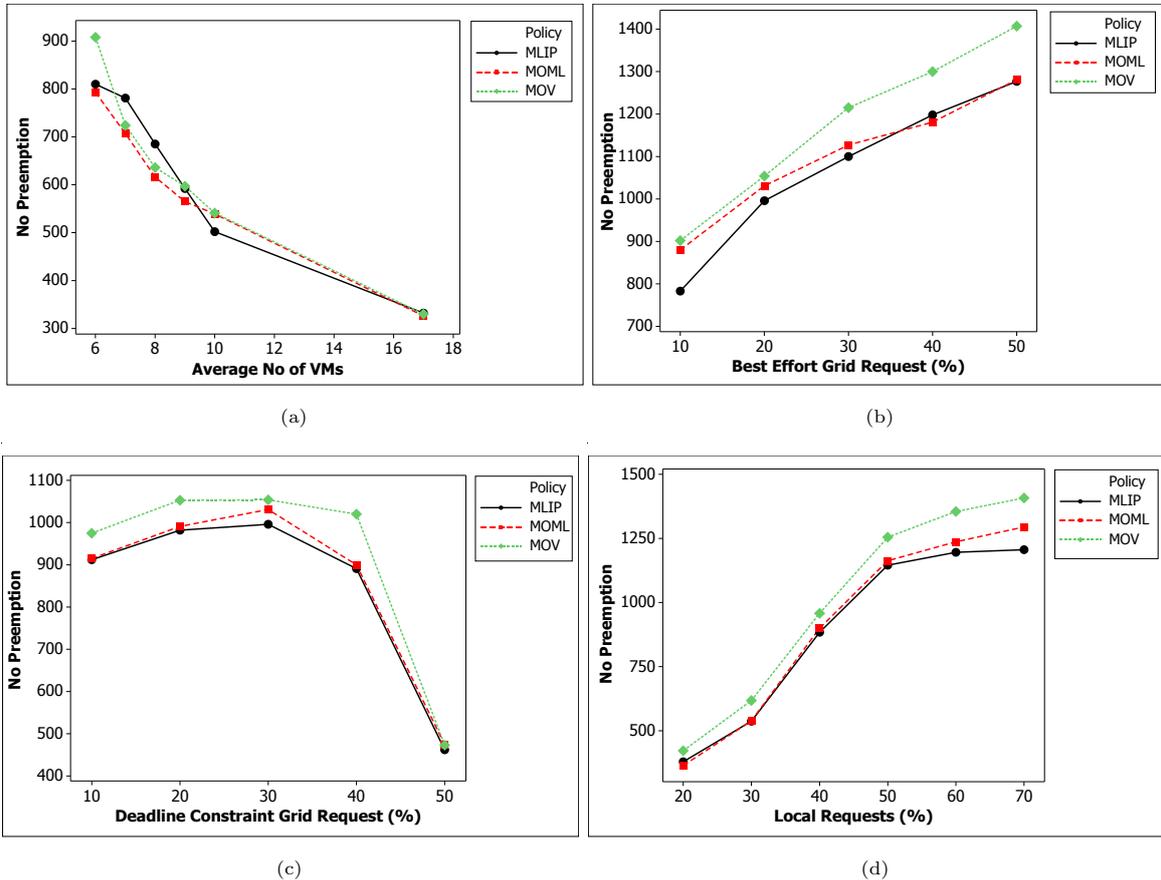


Figure 4: Number of lease preemption resulted from different policies by changing (a) the average number of VMs, (b) percentage of best effort grid requests, (c) percentage of deadline constraint grid requests, and (d) the number of local requests.

room for incoming local requests.

Figure 4(b) shows that by increasing the number of best effort grid requests the number of preemptions increases almost linearly. For a lower percentage of best effort grid requests (percentage best effort < 30%), MOML behaves similar to MOV, however, after that point MOML approaches to MLIP. The reason is that when the number of Suspendable leases is high, the likelihood of having a candidate set with minimum number of leases and not large overall overhead is high. Thus, MOML and MLIP approach each other.

In Figure 4(c), the number of preemptions does not vary significantly when the percentage of deadline constraint grid requests is increased. However, when the percentage of deadline constraint grid requests exceeds 40% the number of preemptions drops sharply (from around 1000 to 500). In fact, best effort grid requests result in an excessive number of preemptions and therefore, when the proportion of deadline constraint requests increases, the number of preemptions decrease significantly.

Figure 4(d) demonstrates that the number of preemptions increases by increasing the number of local requests in all policies almost linearly.

As illustrated in all sub-figures of Figure 4, most of the time MLIP results in a minimum number of preemptions and MOML operates between MLIP and MOV. The only exceptions are in points 7, 8 of Figure 4(a) where MLIP makes more preemptions rather than MOV and MOML. We believe that on these points MLIP has preempted some leases which had short wall times. Therefore, after preemption they are allocated in a close time-slot and again these leases are preempted by MLIP.

6 Conclusion and Future Work

In this research we explored how local requests of an RP can get access to occupied resources in InterGrid. For this purpose we leveraged preempting grid leases in favor of local requests. We proposed different types of leases plus different policies to decide which lease(s) are better choices for preemption. More specifically, we investigated three policies for lease preemption. MOV as a policy that improves system utilization, MLIP that results in fewer preemption and increasing user satisfaction, and MOML which makes a trade-off between resource utilization and user satisfaction.

We observed that preempting leases substantially decrease the rejection of local requests (up to 72% with 95% CI:(51.1,92.8)) without increasing grid requests rejection rate. These results are the same for all proposed preemption policies. We also noticed that MOV performs better in terms of resource utilization in comparing with other policies. On the other hand, MLIP is a better policy in the sense that it preempts fewer leases and therefore causes more user satisfaction. MOML is a policy which satisfies both resource utilization and the user at the same time.

Although the problem we are investigating in this paper is in InterGrid context, it could be also applied to other lease-based Grid/Cloud resource providers where requests with higher priority (such as local or organisational requests) coexist with other requests.

In the future, we plan to extend the current work by considering circumstances where there is a dependency between leases. Furthermore, we are interested in scenarios where local requests are also from different types (e.g. local Suspendable and local Migrateable).

References

- Buyya, R., Murshed, M. M., Abramson, D. & Venugopal, S. (2005), 'Scheduling parameter sweep applications on global grids: a deadline and budget constrained cost-time optimization algorithm', *Softw., Pract. Exper.* **35**(5), 491–512.
- Chase, J. S., Irwin, D. E., Grit, L. E., Moore, J. D. & Sprenkle, S. E. (2003), Dynamic virtual clusters in a grid site manager, *in* 'Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing', Washington, DC, USA, pp. 90–98.
- Costanzo, A. d., Jin, C., Varela, C. A. & Buyya, R. (2009), Enabling computational steering with an asynchronous-iterative computation framework, *in* 'E-SCIENCE '09: Proceedings of the 2009 Fifth IEEE International Conference on e-Science', IEEE Computer Society, Washington, DC, USA, pp. 255–262.
- De Assunção, M., Buyya, R. & Venugopal, S. (2008), 'InterGrid: A case for internetworking islands of Grids', *Concurrency and Computation: Practice and Experience* **20**(8), 997–1024.
- Grit, L., Ramakrishnan, L. & Chase, J. (2007), On the duality of jobs and leases, Technical Report CS-2007-00, Duke University, Department of Computer Science.
- Isard, M., Prabhakaran, V., Currey, J., Wieder, U., Talwar, K. & Goldberg, A. (2009), Quincy: fair scheduling for distributed computing clusters, *in* 'Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles (SOSP)', ACM, pp. 261–276.
- Kettimuthu, R., Subramani, V., Srinivasan, S., Gopalsamy, T., Panda, D. K. & Sadayappan, P. (2005), 'Selective preemption strategies for parallel job scheduling', *IJHPCN* **3**(2/3), 122–152.
- Kwok, Y.-K. & Ahmad, I. (1996), 'Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors', *IEEE Trans. Parallel Distrib. Syst.* **7**(5), 506–521.
- Lublin, U. & Feitelson, D. G. (2001), 'The workload on parallel supercomputers: Modeling the characteristics of rigid jobs', *Journal of Parallel and Distributed Computing* **63**, 1105–1122.
- Snell, Q., Clement, M. J. & Jackson, D. B. (2002), Preemption based backfill, *in* 'Job Scheduling Strategies for Parallel Processing (JSSPP)', Springer, pp. 24–37.
- Sotomayor, B., Keahey, K. & Foster, I. (2008), Combining batch execution and leasing using virtual machines, *in* 'Proceedings of the 17th International Symposium on High Performance Distributed Computing', ACM, New York, NY, USA, pp. 87–96.
- Sotomayor, B., Montero, R. S., Llorente, I. M. & Foster, I. (2009), Resource leasing and the art of suspending virtual machines, *in* 'Proceedings of the 11th IEEE International Conference on High Performance Computing and Communications', Washington, DC, USA, pp. 59–68.
- Walters, J., Bantwal, B. & Chaudhary, V. (2008), 'Enabling interactive jobs in virtualized data centers', *Cloud Computing and Applications* pp. 21–26.
- Zhao, M. & Figueiredo, R. (2007), Experimental study of virtual machine migration in support of reservation of cluster resources, *in* 'Proceedings of the 3rd International Workshop on Virtualization Technology in Distributed Computing', ACM, pp. 5–11.

Jointly Compatible Pair Linking for Visual Tracking with Probabilistic Priors

Robert Reid

School of Computer Science & Software Engineering
The University of Western Australia
35 Stirling Highway, Crawley, W.A. 6009, Australia.
Email: rob@rrfx.net

Abstract

Video sequences of real-world situations are often difficult to track with machine vision. Scenes frequently contain visual clutter, repetitive textures and occlusions that make online visual feature tracking difficult. If the camera is allowed to shake or moving objects are present, the exponential search-space of potential feature matches rapidly becomes intractable for real-time applications. In this paper we introduce “Jointly Compatible Pair Linking” (JCPL), an algorithm that efficiently and deterministically identifies the most globally consensual set of feature-measurement matches in tracking problems with probabilistic priors. We demonstrate JCPL as part of a two-stage visual tracking algorithm, showing it correctly resolving significant matching ambiguities in sequences with highly dynamic camera motion while robustly ignoring moving scene objects. In these experiments we show JCPL and the two-stage tracker evaluating a fixed number of tests in an exponential search-space. In one experiment JCPL tested less than $1/200th$ of the total search space and executed 4.6 times faster than the current gold-standard algorithm “Joint Compatibility Branch and Bound” (JCBB). Given highly ambiguous sequences we show JCPL tracking successfully while standard JCBB chooses incorrect matches and fails. Throughout our experiments the number of costly image matching operations are minimised, where in a typical sequence only 20.4% of the full image matching operations are required.

1 Introduction

Visual tracking algorithms are found in many areas of computer vision and robotics. In the general case image features such as corners, edges or textured segments are selected and repeatedly searched for in following images. Exhaustive “bottom-up” searches proceed by matching correlations over the whole image in pixel-space or by extracting a set of high-level descriptors, such as SIFT (Lowe, 2004), and matching them in descriptor-space. The bottom-up approach blindly searches for every feature over each complete image and can be computationally expensive or even prohibitive if real-time visual tracking is desired.

While matching accuracy is dependent on feature descriptor quality, incorrect matches eventually occur and the correct set of matches need to be identified using a *global consensus* technique. A model

of the underlying scene is formed and used to verify potential matches. The best model with the largest (or most consensual) set of matches is sought, however the solution-space is often too large to search exhaustively. Various sampling techniques, such as the simple random sampling approach of RANSAC (Fischler and Bolles, 1981), reduce this search-space significantly.

Video sequences are often 2D projections of an underlying 3D scene, such that image features that are being tracked tend to move in predictable ways. As an example, a video taken while traveling down a corridor will have features that generally move outward from the center of successive frames. Here prior knowledge of the camera and scene dynamics would allow us to create a motion model that describes how features are likely to move between images. The model can be a simple, such as the 2D smoothing techniques used in optical flow (Fleet and Weiss, 2006), or complex like the 3D visual simultaneous localisation and mapping (SLAM) methods that have been the focus of considerable research over the last decade (Davison et al., 2007). If the model’s state parameters are estimated using a Bayesian estimator such as the Extended Kalman Filter (EKF) or a particle filter, we have probabilistic information, or *priors* available at each step of the visual matching process. These priors can indicate whether a particular set of matches might be correct and also where and how widely to search for features in each image in a “top-down” *active search* (Davison, 2003) that significantly minimises the search effort.

Each feature’s prior describes an active search region where we expect to find any nearest-neighbour, or *individually compatible* (IC) matches. The IC regions for point features with Gaussian priors resemble ellipses in image-space, where the size of the search ellipse is determined by the desired confidence interval, often 3σ or 99.7%. Figure 1(a) shows a real-world scene with several corner features being tracked. This is a relatively difficult scene to track since many of the IC regions have multiple possible matches. Often identical objects, or objects with repeated textures, can produce multiple matches or “perceptual aliasing” like this. The problem is made worse when highly-dynamic motion models, or slow camera frame-rates, require large IC search regions that cover significant amounts of each image. In cluttered scenes occlusions may produce no valid matches along with multiple incorrect matches. Finally when features diverge from the motion model, or the model starts to become inconsistent, large outliers make tracking very difficult. Identifying set of matches and resolving global consensus is an exponentially complex search problem. As an example, if we find [0,3] potential matches for 100 image features there exists $4^{100} \simeq 10^{60}$ combinations of matches, and only one of these sets is likely to be correct.

Copyright ©2011, Australian Computer Society, Inc. This paper appeared at the 34th Australasian Computer Science Conference (ACSC 2011), Perth, Australia, January 2011. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 113, Mark Reynolds, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

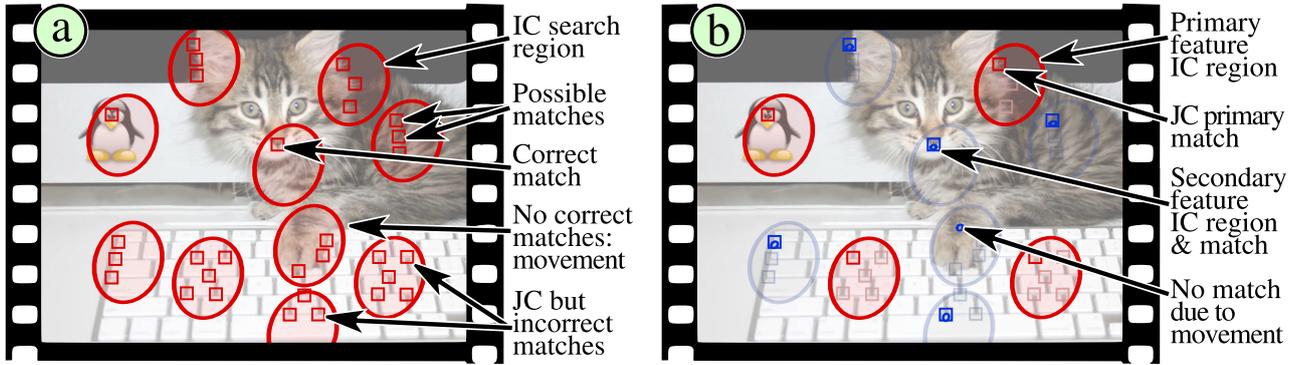


Figure 1: Visually tracking point features with Gaussian priors. A difficult image to track where many of the feature’s individually compatible (IC) search regions (shown as ellipses) have multiple candidate matches (small squares). (a) shows a full, naive, IC search (b) illustrates our JCPL algorithm selecting a consensual set of matches (red ellipses) and the greatly reduced search areas in the second stage (blue ellipses). The online version of this paper includes colour information.

A naive approach to finding global consensus when priors exist and multiple candidate feature measurements are found would be to choose the match closest to the predicted value (the most IC match). However, in many situations this can produce an inconsistent set of matches with many outliers. For example, any matches from a sequence taken by a panning camera are likely to have residuals that are all biased in a similar manner as a result of the common noisy camera motion estimate.

A joint probability distribution over all feature predictions, derived from the underlying model and state estimate, can help identify sets of matches that are in consensus. Neira and Tardós (2001) described the *joint compatibility* test for a potential sets of matches (section 2.2.1). The test takes into account the potentially strong correlations between the feature’s measurement residuals and when a set of matches pass the test they are said to be *jointly compatible* (JC). The JC test can be made very sensitive such that a single outlier will cause it to fail. While it is relatively cheap to perform, a failed test gives no indication of which match or matches were outliers. The JC test could be used to test all possible sets of matches, however the exponential solution-space described above makes this impractical in most cases. Neira and Tardós (2001) also introduced the joint compatibility branch and bound (JCBB) algorithm which makes the search much more efficient. However, as discussed in section 2.2.1 it has issues when multiple “close” measurements are considered for each feature.

Our contribution, JCPL, is an algorithm that efficiently builds the largest possible sets of JC feature matches by linking pairs of JC matches. It executes a computationally bounded search and makes extensive use of information available in the feature’s joint priors. The JCPL algorithm is designed to perform robustly in complex environments with clutter, perceptual aliasing, occlusions and is also tolerant to groups of features that completely break the motion model. We demonstrate JCPL within a two-stage visual tracking algorithm. It identifies globally consensual matches, and reduces both the number of match combinations considered and the total area of IC image regions searched. Our work is motivated by the need to efficiently track large numbers of visual features within practical real-world scenes while operating within hard real-time constraints.

In this paper we refer to multivariate Gaussian probability distributions and follow the notation from Bar-Shalom et al. (Bar-Shalom et al., 2001). As an example the joint prior $\mathcal{N}(\bar{\mathbf{x}}, \mathbf{P}_{\mathbf{xx}})$ is normally dis-

tributed with an predicted value of $\bar{\mathbf{x}} = [\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n]^T$ and square covariance matrix $\mathbf{P}_{\mathbf{xx}}$.

2 Related Work

2.1 Visual SLAM

Visual tracking can be found in most of the vision-based Simultaneous Localisation and Mapping (SLAM) techniques that have received increasing attention over the last decade. The visual SLAM problem aims to sequentially estimate the ego-motion of a camera at the same time as estimating a 3D map of the environment. Here visual tracking is required to identify the feature correspondences, or in SLAM terms the “data association”, between 3D features stored in a map and their projection into the camera’s current 2D image. It is a particularly challenging problem since a single camera is unable to perceive depth, providing “bearing only” information. A probabilistic visual SLAM system could be described as a visual tracker with a very comprehensive 3D motion model. We use a visual SLAM framework in our work to test JCPL and our visual tracking system.

Davison (2003) reported the first real-time monocular visual SLAM system that was later refined by Davison et al. (2007) in their MonoSLAM system. Their approach models a single camera moving with 6 degrees of freedom (DOF) at constant linear and angular velocities within a static 3D environment. The 3D feature map is restricted to a small and sparse set of m corner features due the $O(m^2)$ computational cost of the EKF estimator. Salient corner features are detected using the corner detector of Shi and Tomasi (1994) and small 11×11 pixel image patch “templates” extracted and stored for subsequent visual tracking. In their work they show that the prior for each feature defines an elliptical “active search” region (here referred to as the IC region) that significantly reduces the number of expensive cross-correlation image search operations required. There are many examples of visual SLAM systems in the literature that use some form of top-down active search (Williams et al., 2007; Sola, 2007; Pietzsch, 2008; Paz, Piniés, Tardos and Neira, 2008; Davison et al., 2004; Clemente et al., 2007; Civera et al., 2008), including most real-time implementations.

Figure 2 shows the data flow within a generic Bayesian 3D visual SLAM system. Our JCPL algorithm can be used for visual tracking at ②. The system’s state estimate is a multivariate Gaussian $\mathbf{x} \sim \mathcal{N}(\bar{\mathbf{x}}, \mathbf{P})$. The camera pose $\mathbf{x}_c \sim \mathcal{N}(\bar{\mathbf{x}}_c, \mathbf{P}_{cc})$

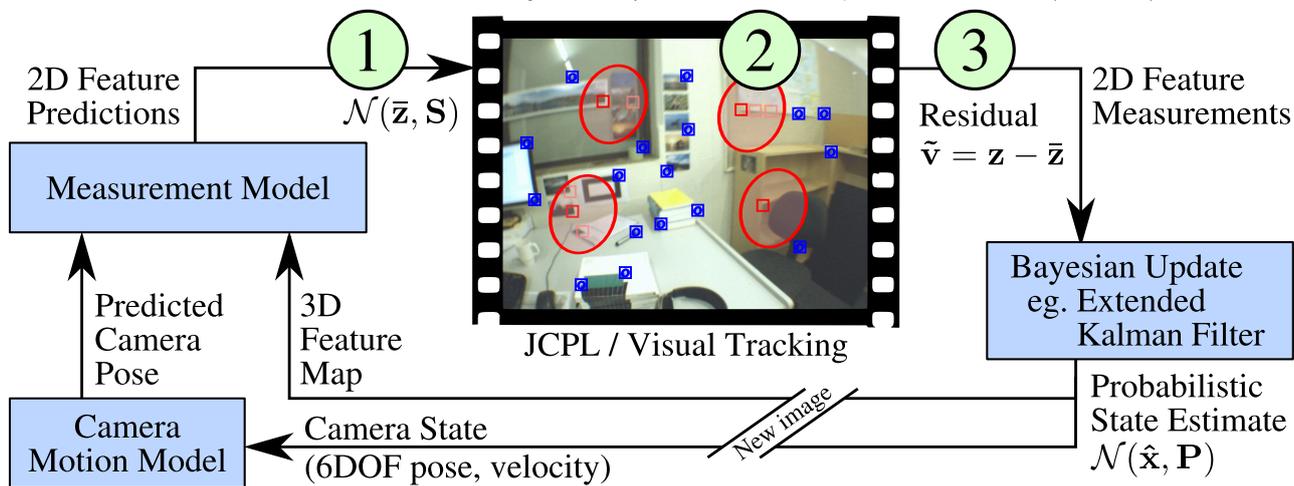


Figure 2: Visual tracking within a 3D visual SLAM framework. The multivariate Gaussian prediction $\mathcal{N}(\bar{\mathbf{z}}, \mathbf{S})$ shown at ① is used to perform a “top-down” search within each feature’s IC ellipse, shown in red at ②. The measurement residuals at ③ are passed back to the Bayesian estimator. The online version of this paper includes colour information.

is predicted for the current frame using a camera motion model (eg. 6DOF constant velocity) while the pose uncertainty \mathbf{P}_{cc} is increased to allow for unknown normally distributed noise. Through a measurement model \mathbf{h}_i each feature $\hat{\mathbf{y}}_i$ is projected into the 2D image “measurement-space” using the predicted camera pose, pose uncertainty and a photometric model. Each feature’s bi-variate Gaussian prediction $\mathbf{z}_i \sim \mathcal{N}(\bar{\mathbf{z}}_i, \mathbf{S}_i)$ is calculated:

$$\bar{\mathbf{z}}_i = \mathbf{h}_i(\bar{\mathbf{x}}_c, \hat{\mathbf{y}}_i) = [\bar{u}, \bar{v}]^T \quad \mathbf{S}_i = \mathbf{H}_i \mathbf{P} \mathbf{H}_i^T + \mathbf{R}_i \quad (1)$$

where \mathbf{H}_i is the Jacobian $\frac{\partial \mathbf{h}_i}{\partial \mathbf{x}}$ and \mathbf{R}_i is the measurement noise estimate. For each corner (point) feature, the innovation covariance matrix \mathbf{S}_i describes an elliptical IC region in the image centered around $\bar{\mathbf{z}}_i$. Its measurement \mathbf{z}_i is IC within a 3σ confidence interval if it satisfies the chi-square test: $[\mathbf{z}_i - \bar{\mathbf{z}}_i]^T \mathbf{S}_i^{-1} [\mathbf{z}_i - \bar{\mathbf{z}}_i] \leq \chi_{2;0.997}^2$. A block matrix expansion of \mathbf{H}_i and \mathbf{P} in equation (1) reveals that \mathbf{S}_i is the summation of 5 terms that each contribute to the overall size of the elliptical search region (Davison, 2003). The dominant term is normally due to the uncertainty in the camera’s pose and is approximately the same size for each predicted feature. The larger the camera’s pose uncertainty, the larger the features’ IC regions become. Similarly the IC ellipse size is made larger by more dynamic camera motion models or slower frame-rates.

The active search approach presented by Davison et al. picks the “best” image match in each feature’s IC region ranked by image correlation score. This works well for uncluttered scenes and low-dynamic camera motion given the small IC regions that are searched. However, if the IC regions are large, and in the difficult situations described earlier, multiple equally valid measurements can arise. Here the set of “best” matches are not necessarily going to be JC, or in global consensus. If they are passed into the Bayesian estimator, it is highly likely it will become corrupted and the inconsistencies will persist until tracking fails.

Referring again to figure 2, our two-stage tracking algorithm with JCPL can be inserted into a visual SLAM framework between the points labeled ① and ③. The input to JCPL is the joint PDF $\mathcal{N}(\bar{\mathbf{z}}, \mathbf{S})$, created by stacking the predictions for each feature from (1). The full innovation covariance $\mathbf{S} = \mathbf{H} \mathbf{P} \mathbf{H}^T + \mathbf{R}$

is calculated by stacking each feature’s Jacobian \mathbf{H}_i , with $\mathbf{R} = \text{diag}(\mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_i)$. If the majority of features are successfully matched, there is little additional cost incurred generating the input priors for JCPL since \mathbf{S} is required for the Bayesian update step after ③. Results using JCPL and the two-stage tracking algorithm are given in section 4. We note here that the cost of our visual tracking approach is normally lower than the full IC active search in (Davison, 2003; Davison et al., 2007) due to the greatly reduced IC image search regions in the secondary set.

2.2 Global Consensus in Visual Tracking

As described in the previous section, performing top-down searches for the “best” match within each feature’s IC region could be considered a basic approach to ensuring global consensus given a large search-space of potential feature-measurement matches. However, as described in section 1, it will fail in complex scenes or with highly-dynamic cameras. While JCPL provides an optimal solution to finding global consensus, we first review other works in this section.

2.2.1 Joint Compatibility & JCBB

The term “joint compatibility” (JC) is first seen in the robotics literature in the work by Neira and Tardós (2001). It describes a chi-squared test (Bar-Shalom et al., 2001) using the joint Gaussian prior $\mathcal{N}(\bar{\mathbf{z}}, \mathbf{S})$, a set of measurement residuals $\tilde{\mathbf{v}} = \mathbf{z} - \bar{\mathbf{z}}$, and a chosen confidence interval (eg. 3σ or 99.7%) to identify whether the measurements contain one or more outliers:

$$D^2 = \tilde{\mathbf{v}}^T \mathbf{S}^{-1} \tilde{\mathbf{v}} \quad D^2 \leq \chi_{\dim(\mathbf{z});0.997}^2 \quad (2)$$

The set of measurement residuals are said to be JC if the Mahalanobis distance D^2 is less than the corresponding chi-square value $\chi_{\dim(\mathbf{z});0.997}^2$. Neira and Tardós (2001) describe the JC test within a SLAM context using an implicit measurement function. In (2) we pose the test within measurement-space, similar to Clemente et al. (2007), and note its similarity to the IC region test in section 2.1.

In the same work Neira and Tardós (2001) describe the joint compatibility branch and bound (JCBB) algorithm, a tree-based search that attempts to efficiently identify the *largest* JC set within the exponential search-space of measurement-feature matches.

Their JCBB approach was presented for situations with weak-or-no data association, using an interpretation tree (Grimson, 1990) to recursively branch over each of the measurements attempting to find which feature they belong to. JCBB has proven to be very effective in many robotics and SLAM related areas where joint priors exist.

Several recent visual SLAM implementations (Clemente et al., 2007; Williams et al., 2007; Pietzsch, 2008) have effectively used JCBB to reject incorrect matches in real-time visual tracking. Clemente et al. (2007) report JCBB taking 2ms for 60 features, while Pietzsch (2008) reports 0.3ms for hundreds of features. However in each implementation they only consider the single “best” candidate measurement from each feature’s IC region producing a simple “yes match” or “null” binary interpretation tree. Each level in the tree has one feature and measurement such that the search is computationally bounded for a particular number of features. In other works where multiple matches per feature are considered, Chli and Davison (2009) report JCBB taking 97.1ms for 11 features. Civera et al. (2010) report JCBB taking up to 20 seconds per frame for 50 features and in one test 1544 seconds, demonstrating the effect of exponential growth in the solution-space. The standard JCBB algorithm requires all image measurements to be made before it resolves consensus. In visual tracking with large IC regions, this represents a lot of additional expensive image search operations.

To help limit the exponential explosion, the JCBB algorithm stops searching below any tree branch when 1) the JC test fails or 2) the size of the current set of non-null matches cannot be larger than the largest set already found. In our experiments where features have multiple possible IC measurements we observed that a given set of matches may *just* fail the JC test, however with the addition of one or more *very* compatible measurement the set could then pass the JC test again. This larger JC set would never be considered by JCBB. Further it is possible for many identically-sized sets to exist that are all JC. While an exhaustive search will pick the “best” match with the lowest D^2 score, the set-size bounding rule in JCBB non-deterministically returns the first of the largest JC sets it finds. For a small set of features (eg $n < 10$) and when candidate measurements are relatively close a sub-optimal set will often be returned. We compare our JCPL algorithm to a modified version of JCBB that is less greedy and returns the optimal set.

2.2.2 RANSAC with Priors

RANSAC (Fischler and Bolles, 1981) is a sampling approach to global consensus that attempts to randomly fit the largest set of measurements to a model. Adaptations using probabilistic information, such as KALMANSAC (Vedaldi et al., 2005) and Guided-MLESAC (Tordoff and Murray, 2005) have been described in the literature. Paz et al. use the JC test as a measure of the model-fit in their randomised joint compatibility (RJC) work (Paz, Pinies, Tardos and Neira, 2008; Paz, Tardos and Neira, 2008), describing its use in both visual tracking and map-aligning. Such non-deterministic, random, approaches are less desirable when an efficient path to the optimal solution (such as JCPL) is available.

Civera et al. (2010) recently described “1-Point RANSAC for EKF Filtering”, an approach that incorporates random sampling directly into an EKF predict-update loop. They show how a single “hypothesis” measurement with strong priors and a motion model can verify global consensus across the

remaining feature measurements. The approach requires only a handful of tests, and conveniently sidesteps the linearisation issues encountered in EKF measurement functions. However, it still searches each feature’s full IC image region, which is costly for large camera pose uncertainties and large numbers of feature measurements.

2.2.3 Active Matching

Chli and Davison (2009) describe their active matching (AM) approach as visual tracking using priors. Their algorithm searches each image efficiently while simultaneously resolving global consensus. It uses a mixture of Gaussians approach to track multiple feature match hypothesis. As features are matched the remaining predictions are repeatedly conditioned and their IC image search regions correspondingly reduced. It could be considered a multi-hypothesis version of either Davison’s original work (Davison, 2005) or sequential compatibility nearest-neighbour (SCNN) as described by Neira and Tardós (2001).

AM has been demonstrated tracking very well in difficult scenes. The original version consumed a large amount of time predicting information gains before deciding where to measure. Handa et al. (2010) recently published their improved CLAM and SubAM versions, where they make approximations that greatly speed up the AM approach. They report their SubAM approach visually tracking hundreds of features in real-time within the PTAM framework (Klein and Murray, 2007). The AM approach is likely to produce similar results to our JCPL work, however there is currently no reference implementation of AM available to perform a direct comparison.

3 JCPL for Visual Tracking

Our two-stage visual tracking algorithm with JCPL takes a camera image, a list of n feature descriptors and a multivariate Gaussian prior $p(\mathbf{z}) \sim \mathcal{N}(\bar{\mathbf{z}}, \mathbf{S})$ predicting the pixel location of these features. The algorithm is broken into 3 stages: 1) the first p “primary” features are selected, predicted and measured. 2) JCPL is used to find the best jointly compatible (JC) set of matches. 3) The remaining “secondary” features are conditioned, measured and matched with their best IC match. The complete set of primary and secondary matches, being in global consensus, are returned without outliers. In our experiments 4 to 8 primary features represented a good balance between tracking stability and computational cost. Refer to figure 1(b) during the following description.

3.1 Primary feature measurement

The visual tracker starts by choosing p primary features using a heuristic that aims to minimise both the cost of finding their JC set, while minimising the uncertainty in the conditioned joint distribution $p(\mathbf{z}_{sec}|\mathbf{z}_{pri})$. The more widely the primary features are spread over the image, the more stable and refined the secondary IC regions become. Conversely, image distortions at the edge of wide-angle lenses and large primary IC search regions suggest more central primary features are chosen.

We divide the image into p regions that the feature predictions $\bar{\mathbf{z}}_i$ are classified into. The features are each given a score proportional to their expected distance to the center of their region. Results from each previous feature measurement are recorded. Features that have previously been predicted but not measured, measured but failed their JC test, or had multiple potential measurements (costly for JC) are flagged

and a penalty is added onto their feature scores. The feature with the lowest score in each region is selected, corresponding to a spatially well distributed and more readily matched primary set.

Each primary feature's image descriptor is used to search its potentially large IC region for candidate measurements. The search results are filtered to find all local maxima and the best $[1, c]$ measurements are returned. The maximum measurement count c is determined to balance expected image clutter, IC search size and computational cost. In our experiments $c = 4$ provided a good balance. If no measurements are found then a replacement primary feature can be chosen and measured from the same image region.

3.2 Jointly Compatibility Pair Linking

JCPL uses the JC test $D^2 \leq \chi_{dim(\mathbf{z});0.997}^2$ to find the *best* and *largest* set of matches for the primary feature measurements. Here we define a *match* as a tuple $H = (F_i, M_j)$ connecting feature i to its measurement j . The small number of features and limited measurements create a solution-space of $(c+1)^p$ possible sets of matches, including sets with null-matches. An exhaustive search of this solution-space, while computationally bounded, is still expensive. For example with $p = 8$, $c = 4$, a full search requires 390625 sets to be tested.

3.2.1 Monotonicity of D^2

The JCPL algorithm deterministically performs a minimal number of JC tests on sets of matches using the monotonic nature of the Mahalanobis distance D^2 as a bounding rule. We use the fact that joining a match H_3 to a set of JC matches $\{H_1, H_2\}$, can only *increase* the distance $D_{H_1 H_2}^2 \leq D_{H_1 H_2 H_3}^2$. Here the conditioned prediction $p(\mathbf{z}_3 | \mathbf{z}_1, \mathbf{z}_2)$ is the only location where \mathbf{z}_3 could be measured that that would give no change in the distance. While attempting to build full p -sized sets of matches $\{H_1 \dots H_p\}$, we can use this property to reject any smaller sets. As an example, for the pair $\{H_1, H_2\}$ if $D_{H_1 H_2}^2 > D_{H_1 \dots H_p}^2$ it is not possible for a better full JC set to be formed using this pair as a base. Further, if we construct a set of matches by joining two pairs, eg. $\{H_1, H_2\} \cup \{H_3, H_4\}$, the new distance is $D_{H_1 H_2 H_3 H_4}^2 \geq \min(D_{H_1 H_2}^2, D_{H_3 H_4}^2)$, being greater than the minimum of the two pairs.¹

3.2.2 Identifying JC Pairs of Matches

JCPL starts by testing the JC of all *pairs* of matches in the primary set $\binom{p}{2}c^2$ (in our example 448 pairs). For each pair $\{H_a, H_b\}$ the $D_{H_a H_b}^2$ distance is recorded against its matches. For each match the best (minimum) distance from all the pairs it was a part of, $D_{pair-min}^2$ is stored. This value is the *minimum* D^2 distance any larger set containing the match can have. The matches are sorted by their $D_{pair-min}^2$ distance and any match without a JC pair at this step is rejected. In our tests for a cluttered scene 40% of pairs failed this first JC test, and up to 20% of the worst outliers are removed at this step. Testing all pairs is computationally inexpensive.

3.2.3 Linking Pairs of Matches

Using all of the JC match pairs, JCPL iteratively links the pairs into chains, attempting to form full p -sized

sets of matches. Only one match per feature is allowed such that the sequence is not allowed to loop back on itself. As JCPL discovers each potential set, the matches are ordered and unique sets are recorded. If a full p -sized set is encountered it is JC tested immediately.

The list of pairs is ordered with the most likely (minimum D_{pair}^2) matches first. With reasonable priors and measurements the matches in the $\binom{p}{2}$ best JC pairs are *very* likely to form all (or at least part) of the best p -sized set. Given these ordered matches $\{H_1 \dots H_p\}$ will be searched first, JCPL is very likely to join them together and test this set of matches *first*. Even if this set's $D_{H_1 \dots H_p}^2$ passes the JC test, we still need to keep searching to ensure we have the *best* full set. However the remaining search becomes very efficient, since we can immediately prune all potential matching pairs where their $D_{pair-min}^2$ is greater than our currently best $D_{H_1 \dots H_p}^2$.

In normal operation this search-space pruning is likely to remove a majority of incorrect matches, drastically reducing the search space. A handful more full JC tests are often all that's required to be certain the best and most JC set of matches have been found. This is the optimistic and most likely case.

If no full p -sized JC sets exist as a result of, for example, image occlusion or unmodelled behavior, the JCPL algorithm starts testing all the $p-1$ and smaller sets. The iterative search has already identified and stored the candidate sets formed by linked pairs of matches. They are also already partially sorted such that the remaining search proceeds efficiently. The largest sized sets (eg. $p-1$) are sequentially tested first, maintaining the secondary ordering. The same D^2 bounding rule is used and any JC sets found during this second stage are immediately used to prune the remaining search space. As in the optimistic case, the algorithm typically only searches a small fraction of candidate sets. However, if every one of these sets fail the JC test the algorithm can fall back to the best JC pair. If this occurs the underlying model and priors are likely to have become inconsistent.

3.3 Secondary feature measurement

The best set of primary matches \mathbf{z}_{pri} are then used to condition the secondary features' priors. The joint Gaussian prior is first reorganised (or indexed for efficiency):

$$p(\mathbf{z}) \sim \mathcal{N}(\bar{\mathbf{z}}, \mathbf{S}) = \mathcal{N}\left(\begin{bmatrix} \bar{\mathbf{z}}_{pri} \\ \bar{\mathbf{z}}_{sec} \end{bmatrix}, \begin{bmatrix} \mathbf{S}_{pp} & \mathbf{S}_{sp} \\ \mathbf{S}_{ps} & \mathbf{S}_{ss} \end{bmatrix}\right)$$

It is then conditioned (Bar-Shalom et al., 2001):

$$p(\mathbf{z}_{sec} | \mathbf{z}_{pri}) \sim \mathcal{N}(\bar{\mathbf{z}}_{sec|pri}, \mathbf{S}_{sec|pri})$$

Where:

$$\bar{\mathbf{z}}_{sec|pri} = \bar{\mathbf{z}}_{sec} + \mathbf{S}_{ps} \mathbf{S}_{pp}^{-1} [\mathbf{z}_{pri} - \bar{\mathbf{z}}_{pri}]$$

$$\mathbf{S}_{sec|pri} = \mathbf{S}_{ss} - \mathbf{S}_{ps} \mathbf{S}_{pp}^{-1} \mathbf{S}_{sp}$$

The refined IC regions given by $\bar{\mathbf{z}}_{sec|pri}$ and $\mathbf{S}_{sec|pri}$ are searched sequentially and the most IC match for each feature chosen. At this point the combined set of globally consensual primary and secondary matches are returned.

To provide a measure of how globally consensual the set is, the JC of the entire set (D^2) can be tested relatively cheaply, since \mathbf{S}^{-1} is usually required for a Bayesian update step.

4 Results

The JCPL algorithm was evaluated as part of a two-stage visual tracking front-end for a 3D visual SLAM

¹Code demonstrating the monotonicity of D^2 is available at the author's website <http://www.rffx.net/2010/08/jcpl.html>

	Two Walls & Moving Teapot				Office Workspace			
	$n = 82, p = 8, c = 8$				$n = 43, p = 8, c = 4$			
	JCPL	JCBB	JCBB ^T	Exh	JCPL	JCBB	JCBB ^T	Exh
Primary selection (ms)	4.83				1.37			
Pri. image search (ms)	64.22 (8.03 per feature)				34.40 (4.29 per feature)			
Consensus (JC) (ms)	31.40	92.63	106.40	550.18	33.44	152.83	154.63	1867.57
Sec. conditioning (ms)	3.28				0.96			
Sec. image search (ms)	179.67 (2.41 per feature)				55.98 (1.60 per feature)			
Total time (ms)	283.40	344.63	358.41	802.19	126.15	245.54	247.34	1960.28
JC tests performed	209	1136	1153	10342	198	1839	1841	42288
Primary pixels	181967 (22746 per feature)				79307 (9913 per feature)			
Secondary pixels	10642 (143 per feature)				7715 (221 per feature)			

Table 1: Timings comparing JCPL, JCBB^T and Exhaustive global consensus approaches. Note that standard JCBB failed to track the video sequences. Also due to the exponential nature of the search, the two-stage visual tracking process was necessarily used in all tests, without it JCBB and the Exhaustive approaches were many orders of magnitude slower.

system as described in section (2.1). The EKF-based visual SLAM implementation followed Davison et al’s MonoSLAM (Davison et al., 2007) with the inverse depth parameterisation (IDP) technique of Civera et al. (Civera et al. (2008)). A notable difference is the large increase (4×) in our camera’s dynamic process model noise. The increased angular velocity noise, in particular, allows for very dynamic camera motion. The system was designed and tested in Python using the Numpy, Scipy and OpenCV libraries. The visual SLAM implementation averages 420ms per frame with 50 features on a 2.50GHz Intel Core2 CPU. While faster visual SLAM implementations exist, this was sufficient to evaluate the JCPL and the tracker.

JCBB is currently the gold-standard and has been widely used for resolving consensus with priors (section 2.2.1), including in many visual SLAM systems (Clemente et al., 2007; Williams et al., 2007; Pietzsch, 2008). Given its prevalence, we chose to compare our work to it. However, in our difficult experimental video sequences it chooses a bad, but still JC, set of matches between 10.4% to 11.6% of the time. These errors causes the EKF state to become inconsistent and tracking fails. To compare our work to JCBB, we modified the overly greedy bounding conditions. We define JCBB^T with the bounding conditions modified to allow further recursions so that the best (and correct) set is found. To help out the measurements were pre-ordered by IC score also. To confirm that JCPL (and also JCBB^T) were returning the best set of matches we exhaustively test all combinations of matches.

It was not practical to test JCBB^T and the exhaustive approach on more than $p = 10$ features with $c = 8$ possible measurements each. The exponential search space made execution several orders of magnitude slower. To compare our work against JCBB^T and an exhaustive search, we gave them both a large advantage by testing them within our two-stage tracker, resolving consensus on the primary features only. All algorithms are treated to the same with caching of intermediate calculations.

We tested JCPL and the tracker on a large number video sequences. We present the results for two scenes here². The first is a repetitively textured synthetic scene containing two walls and a moving teapot. The textures and extremely jerky camera motion were designed to provoke extreme perceptual aliasing. The second sequence was a less cluttered office workspace with fast sweeping camera motion. Various stages of tracking and the consensus test were timed and

²Videos showing our visual tracking results are available at the author’s website <http://www.rffx.net/2010/08/jcpl.html>

recorded. Timings in table 1 are averages taken from between frames 10-200.

We encourage the reader to view the video files accompanying this paper. They show the two-stage visual tracker and JCPL handling each of the test sequences correctly. JCPL identifies the best set of matches in each frame, on average 3.4 to 4.6 times faster than the optimised JCBB^T implementation. JCPL tests only 0.5% to 2.0% of the total solution space while JCBB^T tests an order of magnitude more. Compared to a visual tracker measuring full IC regions for all features, our two-stage tracking process tests 4.9 to 9.7 times less image pixels. The $O((n - p)^2)$ cost incurred in conditioning the secondary features is present, however it and the primary selection heuristic are still minimal compared the cost of a full IC image search. In the first sequence the moving teapot is robustly ignored as it moves after frame 200. We note the IDP features are difficult to initialise during large camera motions, an apparent limitation of the underlying EKF and not JCPL or the tracker. Python’s overheads are evident given the lack of time difference between the average primary and secondary feature measurements.

5 Discussion

In our test sequences with large dynamic camera motion and perceptual aliasing JCPL is extremely robust and tracks correctly. Verified with an exhaustive search, JCPL chooses the “best” set of matches in all frames of all test sequences. The current gold-standard approach, JCBB, fails to identify the correct matches in about 10% of frames and tracking usually fails. Our modified non-greedy version of JCBB (JCBB^T) produces the same correct matches as JCPL, however on average JCPL is 3.4 to 4.6 times faster. Further, if JCBB was *not* used within our two-stage tracker, it would be many orders of magnitude slower again. JCPL is also robust to moving objects that cover up to 25% of each frame (non-static scenes).

JCPL requires a slight increase in memory use compared to JCBB^T, where JCBB’s recursive algorithm is memory stack-intensive compared to JCPL storing lists of potential match sequences. The algorithmic complexity of JCPL is higher than JCBB, here the optimised version of JCPL is about 100 lines of Python code, whereas JCBB^T’s recursive design is only about 40 lines of Python.

While JCBB is currently the most popular approach for visual data association, in the future we plan to directly compare JCPL to the recent work

of Handa et al. (2010). Their approach is complex and no reference implementation has been made available. Execution times given in their original AM paper (Chli and Davison, 2009) show their matching approach is 1.8 to 3.4 times faster than JCBB. While our approach shows a larger improvement, 3.4 to 4.6 times, no direct comparisons or conclusions can be made. SubAM is likely to search a similar number of pixels in each frame, however spend more time repeatedly conditioning groups of feature predictions.

Initial direct comparisons with the 1-point RANSAC approach of Civera et al. (2010) have been performed, however conclusions cannot be made here. Civera's approach is not designed for situations where each feature has a large number of potential IC matches. It begins by finding the best single IC match for each feature and in our tests returns only a small fraction of the globally consensual feature matches. Although a modified approach could produce comparable results, searching all of IC regions would require significant additional image search operations.

We plan to integrate JCPL and the two-stage tracker into an optimised, real-time C/C++ system based on libCVD (Klein and Murray, 2007). Given JCBB has been shown to work well in simple video sequences in real-time, we expect that JCPL with the two-stage tracker will be very effective given the observed performance increase. Even in complex and dynamic scenes its bounded computation time should enable hard real-time constraints. Further, JCPL may be useful on low-powered processors, where higher efficiency and ability to search larger IC areas within each frame, would permit processing at lower frame rates (eg. 2Hz). Our current Python-based, non-optimised, system can be run in real-time at about this frame rate. JCPL's efficient and robust consensus approach allows feature matching thresholds to be loosened, possibly improving tracking abilities.

6 Conclusions

We have described the JCPL consensus algorithm which can be used in a variety of tracking scenarios where probabilistic priors are available. We have demonstrated JCPL and a two-stage visual tracker as the front-end for a visual SLAM system. JCPL produces notable speed, accuracy and robustness improvements over JCBB, the most common global consensus approach.

References

- Bar-Shalom, Y., Li, X. R. and Kirubarajan, T. (2001), *Estimation with Applications to Tracking and Navigation*, Wiley-Interscience.
- Chli, M. and Davison, A. J. (2009), 'Active matching for visual tracking', *Robotics and Autonomous Systems* **57**(12), 1173–1187.
- Civera, J., Davison, A. J. and Montiel, J. M. M. (2008), 'Inverse depth parametrization for monocular SLAM', *IEEE Transactions on Robotics*.
- Civera, J., Davison, A. and Montiel, J. (2010), '1-Point RANSAC for EKF filtering. application to Real-Time structure from motion and visual odometry', *Journal of Field Robotics to appear*.
- Clemente, L., Davison, A., Reid, I., Neira, J. and Tardós, J. (2007), Mapping large loops with a single hand-held camera, in 'Robotics: Science and Systems'.
- Davison, A. (2003), Real-time simultaneous localisation and mapping with a single camera, in 'IEEE International Conference on Computer Vision', p. 1403–1410.
- Davison, A. (2005), Active search for real-time vision, in 'Proceedings of the 10th International Conference on Computer Vision, Beijing'.
- Davison, A., Cid, Y. and Kita, N. (2004), Real-Time 3D SLAM with Wide-Angle vision, in 'IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon'.
- Davison, A., Reid, I., Molton, N. and Stasse, O. (2007), 'MonoSLAM: Real-Time single camera SLAM', *IEEE Trans. On Pattern Analysis And Machine Intelligence*.
- Fischler, M. A. and Bolles, R. C. (1981), 'Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography'.
- Fleet, D. J. and Weiss, Y. (2006), 'Optical flow estimation', *Mathematical Models of Computer Vision: The Handbook* p. 239–258.
- Grimson, W. E. L. (1990), *Object recognition by computer: the role of geometric constraints*, MIT Press.
- Handa, A., Chli, M., Strasdat, H. and Davison, A. (2010), Scalable active matching, in 'IEEE Conference on Computer Vision and Pattern Recognition'.
- Klein, G. and Murray, D. (2007), Parallel tracking and mapping for small AR workspaces, in 'IEEE and ACM International Symposium on Mixed and Augmented Reality'.
- Lowe, D. (2004), 'Distinctive image features from Scale-Invariant keypoints', *International Journal of Computer Vision* **60**(2), 91–110.
- Neira, J. and Tardós, J. (2001), 'Data association in stochastic mapping using the joint compatibility test', *IEEE Transactions on Robotics and Automation* **17**(6), 890–897.
- Paz, L., Pinies, P., Tardos, J. and Neira, J. (2008), 'Large-Scale 6-DOF SLAM with Stereo-in-Hand', *IEEE Transactions on Robotics* **24**(5), 946–957.
- Paz, L., Tardos, J. and Neira, J. (2008), 'Divide and conquer: EKF SLAM in $\mathcal{O}(n)$ ', *IEEE Transactions on Robotics* **24**(5), 1107–1120.
- Pietzsch, T. (2008), Efficient feature parameterisation for visual slam using inverse depth bundles, in 'Proc. British Machine Vision Conf'.
- Shi, J. and Tomasi, C. (1994), Good features to track, in 'IEEE Conference on Computer Vision and Pattern Recognition', p. 593–600.
- Sola, J. (2007), Towards Visual Localization, Mapping and Moving Objects Tracking by a Mobile Robot: a Geometric and Probabilistic Approach, PhD thesis.
- Tordoff, B. J. and Murray, D. W. (2005), 'Guided-MLESAC: faster image transform estimation by using matching priors', *IEEE Trans. PAMI* **27**(10), 1523–1535.
- Vedaldi, A., Jin, H., Favaro, P. and Soatto, S. (2005), KALMANSAC: robust filtering by consensus, in 'Proceedings ICCV', Vol. 1, p. 633–640.
- Williams, B., Klein, G. and Reid, I. (2007), Real-time SLAM relocation, in 'IEEE International Conference on Computer Vision', p. 1–8.

Hybrid wrapper-filter approaches for input feature selection using Maximum relevance-Minimum redundancy and Artificial Neural Network Input Gain Measurement Approximation (ANNIGMA)

Shamsul Huda, John Yearwood, Andrew Stranieri

CIAO, GSITMS, University of Ballarat, Victoria, Australia

s.huda@ballarat.edu.au, j.yearwood@ballarat.edu.au, a.stranieri@ballarat.edu.au

Abstract

Feature selection processes improve the accuracy, computational efficiency and scalability of classification process in data mining applications. This paper proposes two filter and wrapper hybrid approaches for feature selection techniques by combining the filter's feature ranking score in the wrapper stage. The first approach hybridizes a Mutual Information (MI) based Maximum Relevance (MR) filter ranking heuristic with an Artificial Neural Network (ANN) based wrapper approach where Artificial Neural Network Input Gain Measurement Approximation (ANNIGMA) has been combined with MR (MR-ANNIGMA) to guide the search process in the wrapper. The second hybrid combines an improved version of MI based (Maximum Relevance and Minimum Redundancy; MaxRel-MinRed) filter ranking heuristic with the wrapper heuristic ANNIGMA (MaxRel-MinRed-ANNIGMA). The novelty of our approach is that we integrate the capability of wrapper approach to find better feature subset by combining filter's ranking score with the wrapper-heuristic's score that take advantages of both filter and wrapper heuristics. The performances of the hybrid approaches have been verified using synthetic, bench mark data sets and real life data set and compared to both independent filter and wrapper based approaches. Experimental results show that hybrid approaches (MR-ANNIGMA and MaxRel-MinRed-ANNIGMA) achieve more compact feature sets and higher accuracies than filter and wrapper approaches alone.

Keywords: Hybrid Feature Selection, Wrapper, Filter Maximum-Relevance, Maximum-Relevance and Minimum Redundancy, ANNIGMA wrapper,

1 Introduction

Feature selection is an important and frequently used data pre-processing technique in machine learning (Blum, Langely 1997), (John, Kohavi 1994), data mining (Dash, Liu 1997), medical data processing (Puronen et al 2000) and statistical pattern recognition areas (Bne-Bassat 1982), (Mitra et al. 2002). Due to rapid advances of computational technologies and internet, datasets are getting larger and larger. To use datasets with thousands of features for decision making, prediction or classification purposes by using data mining techniques is

a challenge for researchers and practitioners because the performance of data mining methodologies degrades with huge volumes of training data (Blum, Langely 1997), (John, Kohavi 1994), (Dash, Liu 1997). Therefore, feature selection from datasets by removing irrelevant, redundant or noisy features is a primary task for machine learning researchers. Given an m -dimensional dataset, a feature selection algorithm needs to find optimal feature subset from the 2^m subsets of the feature space. Therefore finding an optimal feature subset is computationally expensive (Kohavi et al. 1997). The performance of a feature selection algorithm depends on its evaluation criterion and search strategies.

Significant research works have appeared in the literature on feature selection. These can be grouped broadly into three main categories based on the evaluation criteria: I) the filter model (Dash, Liu 1997), (Kwak et al. 2002), (Wang et al. 1999), (Hall 2000) II) the wrapper model (Kwak et al. 2002) (J.G. Dy et al. 2000) (Hsu et al. 2002) (Dash, Liu 1997) and III) hybrid models (Zhu et al. 2002). The filter models are based on the intrinsic characteristics of the data and do not involve the application of an induction algorithm. Filter models are computationally cheap due to its evaluation criteria. However, feature subsets selected by filter may result in poor prediction accuracies, since they are independent from the induction algorithm. In contrast, the wrapper model (Kwak et al. 2002) (J.G. Dy et al. 2000) (Hsu et al. 2002) uses a predetermined induction algorithm and uses predictive accuracy as the evaluation criteria for the feature selection. However, wrapper models face huge computational overhead due to the use of the induction algorithm's performance criteria as its evaluation criteria. In (Hsu et al. 2002), (Kohavi et al. 1997), a hybrid of genetic algorithm and filter heuristic were proposed where GA framework works as subset generation process and filter heuristic improves local search. Despite significant researches on evaluation criteria and search strategies, current generation feature selection literature lacks the work that can combine the merit of wrapper and filter approaches.

In this paper, we propose a hybrid wrapper and filter approach by using the filter's feature ranking score with the wrapper heuristics in the wrapper stage to speed up the search process and find optimal feature subset in the wrapper stage. In our approach, we hybridize two novel filter heuristics with Artificial Neural Network Input Gain Measurement Approximation (ANNIGMA) wrapper heuristic. The first proposed hybrid feature selection approach uses Mutual Information (MI) based Maximum Relevance (MR) filter ranking heuristics with

ANNIGMA. The second hybrid approach uses an improved version of MI-based filter heuristic Maximum Relevance and Minimum Redundancy; MaxRel-MinRed) with the ANNIGMA (MaxRel-MinRed-ANNIGMA). The novelty of our approach is that we use a wrapper and filter hybrid that combines the filter’s ranking score with the wrapper-heuristic’s score to guide the search process in the wrapper stage. The proposed approaches avoid the computational overhead of hybrid GA-based approaches (Hsu et al. 2002), (Kohavi et al. 1997) and takes advantage of both filter and wrapper heuristics which are absent in the traditional GA-based hybrid approaches (Hsu et al. 2002), (Kohavi et al. 1997). This type of hybrid approach is a new concept and has not been explored yet in the literature.

The rest of the paper is organized as follows. The next section introduces some related literature. The proposed hybrid of wrapper-filter feature selection algorithm using the combination of filter heuristic Maximum-Relevance-Minimum-Redundancy (MaxRel-MinRed) and Artificial Neural Network Input Gain Measurement Approximation (ANNIGMA) is described in Section 3. It also discusses the hybrid approach using Maximum-Relevance (MR) and ANNIGMA. Section 4 presents experimental results and discussion. Conclusions of this study are presented in the last section.

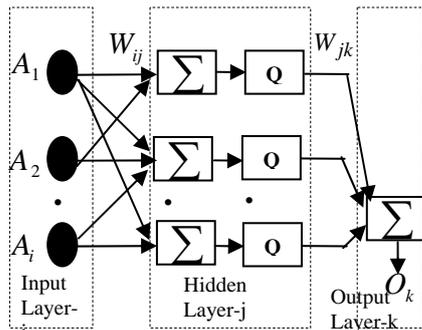


Figure 1. A single hidden layer Multi layer Perceptron (MLP) neural network in wrapper approach

2 Related Work.

2.1 Generalized Filter and wrapper approaches

Filter approaches start from an initial subset either empty or full and generate a new subset each step by following a search strategy (either forward or backward or bi-directional) through the feature space. Each generated subset is evaluated using filter heuristics. If a current subset has a higher evaluation score than previous, it is assigned as the current best subset. The search process stops on a user defined stopping criteria based on the score and number of optimal feature set. The final subset can be justified further using an induction algorithm. In the wrapper approach (Kwak et al. 2002) (J.G. Dy et al. 2000) (Hsu et al. 2002) generated subsets are evaluated using a predetermined induction algorithm. However, subsets in the wrapper approach are evaluated by the predictive accuracies of a trained classifier, therefore are more significant than those in the filter approach. In the Neural network based wrapper literature, search process in the wrapper can also guided by a wrapper heuristic such as Artificial Neural Network Input Gain

Measurement Approximation (ANNIGMA) (Hsu et al. 2002) which shows significant improvement over the wrapper alone.

2.2 Artificial Neural Network Input Gain Measurement Approximation (ANNIGMA)

Artificial Neural Network Input Gain Measurement Approximation (ANNIGMA)(Hsu et al. 2002) is a weight analysis based wrapper heuristic that ranks features by relevance based on the weight associated with feature in a Neural Network based wrapper approach. Features that are irrelevant or redundant will produce more error than relevant features. Therefore, during training, weights of noisy features are controlled in such a way that they contribute to the output as least as possible. ANNIGMA (Hsu et al. 2002) is based on the above strategy of the training algorithm. For a two layer Neural Network, (Figure 1) if i, j, k are the input, hidden and output layer and Q is a logistic activation function (1) of the first layer and second layer has a linear function, then output of the network is as (2).

$$Q(x) = (1/(1 + \exp(-x))) \tag{1}$$

$$O_k = \sum_j Q\left(\sum_i A_i \times W_{ij}\right) \times W_{jk} \tag{2}$$

Then local gain is defined as (3)

$$LG_{ik} = \frac{\Delta O_k}{\Delta A_i} \tag{3}$$

According to C.N. Hsu and H.J. Huang et.al. (Hsu et al. 2002), the local gain can be written in terms of network weight as (4):

$$LG_{ik} = \sum_j |W_{ij} \times W_{jk}| \tag{4}$$

Then ANNIGMA score for feature-i (F_i) is the local gain (LG) normalized based on a unity scale as (5)

$$ANNIGMA(F_i) = \frac{LG_{ik}}{\max_{(i)} LG_{ik}} \tag{5}$$

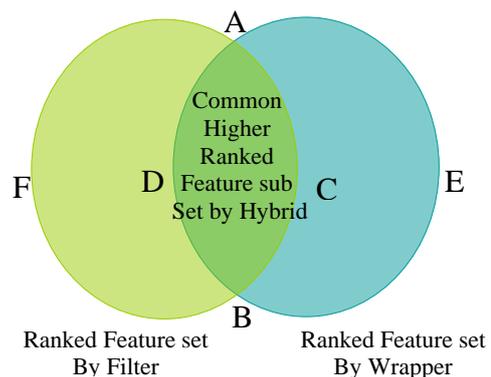


Figure 1.1 Venn diagram for combined heuristics

3 Hybrid feature selection algorithms using Maximum Relevance and Minimum Redundancy Filter Heuristic and Artificial Neural Network Input Gain Measurement Approximation Wrapper Heuristics

Standard filter approaches can extract knowledge of the intrinsic characteristics from real data. However filter

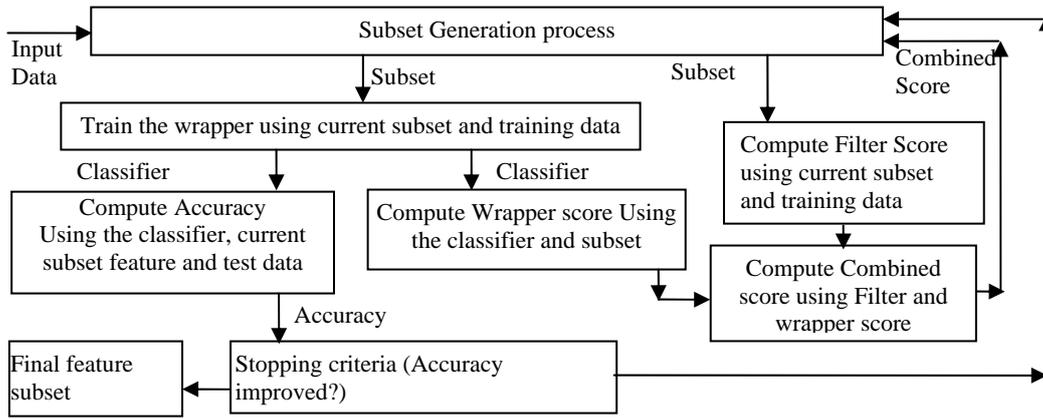


Figure 2. General Framework for proposed hybrid of wrapper and Filter feature selection approaches

approaches do not use any performance criteria based on predictive accuracies. This does not guarantee that final feature subset will do better in classification/prediction tasks. In contrast, the wrapper approaches (Kwak et al. 2002), (J.G. Dy et al. 2000), (Hsu et al. 2002) use a predetermined induction algorithm and different search strategies (Jain et al. 1997), (Ferri et al. 1994) to find the best feature subset. Use of predictive-accuracy based evaluation criteria in the wrapper ensures good performance from the selected feature subset. However repeated execution of the induction algorithm (in the worst case exponential search space) in the search process incurs a high computational cost in the wrapper approach.

In this paper, proposed hybrid approaches introduce the filter heuristic in the wrapper stage and take advantages of both approaches which is able to find more significant features than either wrapper and filter alone. The idea behind this approach can be explained by the Venn-diagram (in figure 1.1.). If the two feature subsets (ACBF and ADBE figure 1.1) are separately ordered/ranked according to their score, then common higher ranked feature subset (ACBD) is the strongly recommended most significant feature subset by the both feature selection algorithms. If the scores of both algorithms are normalized on the same scale and combined (summed), then feature subsets with higher combined scores provide the common higher ranked feature subset from both algorithms. A Backward Elimination (BE) search strategies based on the combined score along with the wrapper evaluation criteria can find the most significant features. Performance of the combined score may be affected due to performance of the incorporated filter for a particular wrapper approach in the hybrid. However, different filter approaches can be combined to find a suitable hybrid for a particular wrapper heuristic and vice-versa. In this paper, we have combined two filter heuristics: mutual information based Maximum Relevance (MR), Maximum Relevance-Minimum Redundancy (MaxRel-MinRed) with Artificial Neural Network Input Gain Measurement Approximation (ANNIGMA) based wrapper. Here, we have focused on a Neural network based wrapper and different filter heuristics. We will use other wrapper approaches in a future work. The following sub-sections describe different heuristics and steps of the proposed hybrid algorithms.

3.1 Maximum Relevance (MR)

Relevant features provide more information about the class variable than irrelevant features. Therefore mutual information based maximum relevance (Wang et al. 1999) is a good heuristic to select salient features in data mining area. If S is a set of features F_i and class variable is c , the maximum relevance (Wang et al. 1999) can be defined as (6).

$$\max_{\text{imum}} \text{Relevance}(S, c) = \frac{1}{|S|} \sum_{f_i \in S} I(F_i; c) \quad (6)$$

$I(F_i; c)$ is the mutual information between F_i and class c which is defined as (7).

$$I(F_i; c) = H(F_i) - H(F_i | c) \quad (7)$$

$H(F_i)$ is the entropy of F_i with the probability density function $p(f_i)$ where F_i takes discrete values from the set $F = \{f_1, f_2, \dots, f_i\}$, then $H(F_i)$ is defined as (8)

$$H(F_i) = - \sum_{f_i \in F} p(f_i) \log p(f_i) \quad (8)$$

$H(F_i | c)$ in (2) is the conditional entropy between F_i and c and is defined as (9)

$$H(F_i | c) = - \sum_{f_i \in F} \sum_{c_i \in C} p(f_i, c_i) \log p(c_i | f_i) \quad (9)$$

where class variable c takes the discrete values from the set $C = \{c_1, c_2, \dots, c_i\}$.

3.2 Maximum Relevance and Minimum Redundancy

Maximum relevance (MR) (Wang et al. 1999) can select features that are highly relevant to class. However MR may contribute to redundancy. When two features are highly dependent on each other, the corresponding class discriminative ability of the two features would not be affected much if one of them were removed. Therefore, to avoid the redundancy in MR, a redundancy function is incorporated with maximum relevance as (10)

MR - Minimum Redundancy =

$$\frac{1}{\max_s |S|} \sum_{f_i \in S} I(F_i; c) - \text{Minimum Redundancy} \quad (10)$$

Minimum Redundancy is defined as (11)

$$(\text{Red}, c) = \frac{1}{|S|^2} \sum_{i, j \in S} I(F_i; F_j) \quad (11)$$

Where $I(F_i; F_j)$ is the mutual information between the features F_i and F_j .

3.3 Computation of combined score in proposed hybrid algorithm-1: Hybrid of Maximum Relevance and ANNIGMA (MR-ANNIGMA)

The proposed MR-ANNIGMA uses Artificial Neural Network as the classification algorithm in the wrapper stage. An n-fold cross-validation approach has been used in MR-ANNIGMA to train the wrapper. In each fold we compute the ANNIGMA score for every feature. Then after training of all folds, the ANNIGMA score is averaged as (12):

$$ANNIGMA(F_i)_{average} = \left(\frac{1}{n}\right)(ANNIGMA(F_i)_1 + .. + ANNIGMA(F_i)_n) \quad (12)$$

While computing the combined score in the proposed ANNIGMA, the relevance of a feature in the current subset is computed from the individual score which is scaled to the maximum individual relevance of the subset. Thus relevance of a feature in a subset in the hybrid approach is as (13)

$$Relevance(F_i) = \frac{I(F_i; c)}{\max_{f_j \in S} I(F_j; c)} \quad (13)$$

The combined score of filter’s heuristic and wrapper’s heuristic in the proposed MR-ANNIGMA is computed as (14).

$$Combined\ Score(MR - ANNIGMA) = \frac{I(F_i; c)}{\max_{f_j \in S} I(F_j; c)} + ANNIGMA(F_i)_{average} \quad (14)$$

3.4 Computation of combined score in proposed hybrid algorithm-2: Hybrid of Maximum Relevance-Minimum-Redundancy and ANNIGMA (MaxRel-MinRed-ANNIGMA).

As the MR-ANNIGMA, the proposed (MaxRel-MinRed –ANNIGMA) uses ANN as the wrapper. The ANNIGMA score is computed as (12). An incremental search method (Peng et al. 2005) is used to compute the Maximum Relevance and Minimum Redundancy score as (15). Maximum Relevance and Minimum Redundancy (MaxRel-MinRed) score is the difference of maximum relevance score of a candidate features in the candidate set and redundancy score between the corresponding feature with a feature in the goal set.

$$MaxRel_MinRed\ Score = \max_{F_i \in F - F_{l-1}} \left(\frac{1}{|S|} \sum_{f_j \in S} I(F_i; c) - \frac{1}{l-1} \sum_{F_j \in F_{l-1}} I(F_i; F_j) \right) \quad (15)$$

Since the MaxRel-MinRed score is a difference of feature score which is relative to the search iteration, while computing the combined score in the hybrid, an equivalent weighted score of MaxRel-MinRed score is computed for each feature. First the features are ordered according to their ranks in the MaxRel-MinRed incremental search method (Peng et al. 2005). Then equivalent weighted score is computed from their ranking on a unity scale. Orders of the feature ranking are

incremental integers starting from one to total number of features in the data set where top-ranked has a maximum score of one. Therefore, equivalent weighted MaxRel-MinRed score is as (16)

$$Weighted\ MaxRel_Min\ Red\ Score(F_i) = 1 - (Rank\ feature\ (F_i)\ in\ MaxRel_Min\ Red / |F|) \quad (16)$$

The combined score of filter’s heuristic and wrapper’s heuristic in the proposed (MaxRel-MinRed –ANNIGMA) is computed as (17).

$$Combined\ Score(MaxRel_MinRed\ ANNIGMA; F_i) = Weighted\ MaxRel_MinRed\ Score(F_i) + ANNIGMA(F_i)_{average} \quad (17)$$

Algorithm-1 and 2: Procedure (Hybrid Wrapper-Filter approach)

Input: $D(F_1, F_2, \dots, F_m)$ // Training data with m features

Output: S_{BEST} //an optimal subset of features

Begin

1. Let S=whole set of m features F_1, F_2, \dots, F_m
2. S_0 =Initial set of feature which records all generated subsets with accuracy
3. for N = 1 to m-1
4. Current set of feature $S_{current} = S$
5. Compute Filter score by (6) and (10)
6. for fold=1 to n
7. Train the network with $S_{current}$
8. Compute ANNIGMA of all features
9. Compute Accuracy
10. endfor
11. Compute average accuracy of all folds for $S_{current}$
12. Compute average ANNIGMA of $S_{current}$ by (12)
13. Compute combined score for every feature in $S_{current}$ by (14 to 19) for hybrids
14. Rank the features in $S_{current}$ using the combined score in descending order
15. $S_0 = S_0 \cup S_{current}$
16. Update the current feature set $S_{current}$ by removing the feature with lowest score
17. endfor
18. $S_{BEST} =$ Find the subset form S_0 with the highest accuracy.
19. return S_{BEST}

End

3.5 Detail steps of Hybrid algorithms (MaxRel-MinRed-ANNIGMA and MR-ANNIGMA)

The detail algorithm of hybrid approaches is described in algorithm-1 and 2 and Figure 2.

3.5.1 Search strategies and subset generation in MaxRel-MinRed-ANNIGMA and MR-ANNIGMA

Both of the hybrid approaches use a Backward Elimination (BE) search strategy to generate a subset of features. Initially hybrid starts with the full feature set. Subset generation in BE is guided by the wrapper-filter hybrid heuristic score. The combined score computation follows the steps of sub-sections (3.1 to 3.4). When the number of features in BE process is significantly reduced compared to total feature, the filter score component is weighted less than the wrapper score as (18) and (19)

$$\begin{aligned}
 \text{Combined Score}(\text{MaxRel-MinRed ANNIGMA}; F_i) = & \\
 (u * \text{Weighted MaxRel_MinRed Score}(F_i)) & \\
 + (v * \text{ANNIGMA}(F_i)_{\text{average}}) & \\
 \text{-----} & \text{-----} (18)
 \end{aligned}$$

$$\begin{aligned}
 \text{Combined Score}(\text{MR-ANNIGMA}) = & \\
 (u * \frac{I(F_i; c)}{\max_{(f_j \in S)} I(F_j; c)}) + (v * \text{ANNIGMA}(F_i)_{\text{average}}) & \\
 \text{where } 1 \leq u, v \geq 0 \text{ -----} & \text{-----} (19)
 \end{aligned}$$

3.5.2 Wrapper step in MR-ANNIGMA/MaxRel-MinRed-ANNIGMA

Both the proposed MR-ANNIGMA and MaxRel-MinRed-ANNIGMA, use a single hidden layer Multi Layer Perceptron (MLP) Network (Figure-1) in the wrapper stage. An n-fold cross validation approach has been applied in the training of the network. The evaluation criterion of feature subset is based on the average prediction accuracy over n-fold of the wrapper (MLP network). In Algorithm-1 and 2, steps-1 to 11 computes the average accuracy over n-folds for the current subset of features. Step-12 to step-14 computes the hybrid scores and ranks the features based on their combined score. Step-15 to step-16 generates new subset based on the feature ranking and keep records of evaluated feature subsets with their accuracy. The BE processes in MR-ANNIGMA and MaxRel-MinRed-ANNIGMA update MR, MaxRel-MinRed and ANNIGMA and the combined score in every iteration. The combined score guides the subset generation. The BE continues until a single feature is remaining in the current subset. The subset with highest accuracies or close to the highest accuracies with fewer features than it is chosen as the final feature subset.

4 Experimental Results and discussion

The proposed hybrids (MR-ANNIGMA and MaxRel-MinRed-ANNIGMA) have been tested on both synthetic and UCI Machine learning repository data sets (Asuncion et al.) and Tobacco control policy evaluation dataset (Thompson et al. 2006) in Table-1. For each data set in Table-1, the data is normalized in the range [-1, 1]. CAIM (Kurgan et al. 2004) discretization technique has been used for continuous attributes while computing filter score. A single hidden layer neural network with the different network configuration for each data set (Table-1) is used.

The results of the hybrids (MR-ANNIGMA and MaxRel-MinRed-ANNIGMA) have been compared to filter approaches MR, MaxRel-MinRed and the wrapper ANNIGMA. Each of the above five algorithms were tested using 10-fold cross validation and executed for 10-

trials. In the BE process 2/3 iterations use (u=v=1) and last 1/3 iterations uses (u=0.3, v=0.7). The average accuracies from 10 trials were considered for final accuracies and described in Table 2 to Table 8.

Data set	Hidden nodes	Hidden Layer Transfer function	Output Transfer function	Max. epoc
Synthetic	5	tansig	purelin	250
Wine	6	tansig	purelin	150
Ionosphere	22	tansig	purelin	300
Cancer (Diagnostic)	12	tansig	logsig	400
Sonar	24	tansig	purelin	200
Pima	6	tansig	purelin	200
Tobacco	22	tansig	purelin	350

Table 1: Network construction data for different data sets.

4.1 Evaluation and experimental analysis of the search process in the hybrids using combined score on a Synthetic Data set

This synthetic data set has been constructed with 6 features as Table-II where W, X, Y and Z are uniformly distributed on [-0.5, +0.5].

Features	Description
Featrure-1 (F_1):	X
Featrure-2 (F_2):	3X+1
Featrure-3 (F_3):	W
Featrure-4 (F_4):	W-Y
Featrure-5 (F_5):	Z
Featrure-6 (F_6):	2Z + 1

Table 2 : Description of Input Features of synthetic data

Index of BE Iteration	Total Features in BE iterations	MR	MaxRel-MinRed	ANNI-GMA	MR-ANNIGMA	MaxRel-MinRed-ANNIGMA
i)	6	6.917	77.167	7.167	76.883	77.267
ii)	5	7.467	77.683	7.117	77.700	77.717
iii)	4	7.783	77.700	7.900	77.850	77.900
iv)	3	6.500	75.650	6.767	76.583	77.967
v)	2	6.217	76.100	6.217	76.550	76.250
vi)	1	7.250	67.283	67.600	67.617	67.317

Table 3: Accuracies for five algorithms at different iterations of BE for synthetic data set

$$C = \begin{cases} 0 & \text{if } (2X - W) < 0 \\ 1 & \text{if } (2X - W) \geq 0 \end{cases} \quad (20)$$

$$C = \begin{cases} 0 & \text{if } (Y + W) > 0 \\ 1 & \text{if } (Y + W) \leq 0 \end{cases} \quad (21)$$

The class variable C is binary type and has been generated by using two rules (20) and (21). 600 samples have been generated in which 300 samples' class values have been computed using (20) and rest 300 samples' class values have been computed using (21). It is seen from Table-2 that class C can be computed using Features-(F_1 or F_2) and (F_3 and F_4). Features (F_5 and F_6) are irrelevant to class variable C and F_2 is redundant with F_1 . Therefore, the actual salient feature sets are

(F_1, F_3, F_4) or (F_2, F_3, F_4). All five algorithms have been executed on this synthetic data set. The different iterative accuracies of BE process in five algorithms have been given in Table-3.

First Iteration (Total 6 Features)	Accuracies (77.167%)	Second Iteration (5 Features)	Accuracies (77.117%)	Third Iteration (Total 4 Features)	Accuracies (77.900%)
Feature Index	ANNIGMA Score	Feature Index	ANNIGMA Score	Feature Index	ANNIGMA Score
3	0.974419	3	0.999538	3	0.997381
4	0.548960	1	0.799970	1	0.531413
1	0.547774	2	0.783565	2	0.473748
2	0.516855	4	0.283010	4	0.421396
5	0.105015	5	0.127403	5	Removed
6	0.099459	6	Removed	6	Removed

Table 4: Accuracies and features' score at different iterations of BE in ANNIGMA for synthetic data set. Removed means corresponding feature has been removed in this iteration

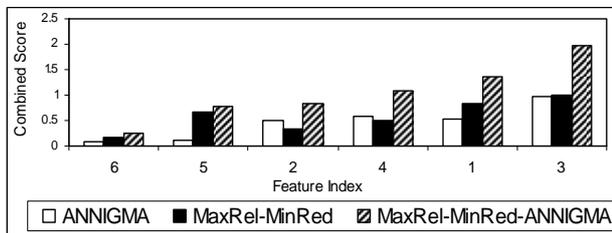


Figure 3. Synthetic Data: The combined score in the first iteration of BE process in the hybrid (MaxRel-MinRed -ANNIGMA) when total features is 6. Y-axis gives the combined score and X-axis gives the feature's serial no.

The ranking order of features in MR is (3, 1, 2, 4, 5, 6) in synthetic data. The BE process for MR finds best accuracy with four features (3, 1, 2, 4) as in Table-3 (iteration-iii) where MR incorporates the redundant feature-2. The ANNIGMA score of features in different iterations of BE for ANNIGMA heuristic is given in Table-4. It is seen in Table-4 that order of features changes (according to the score) in different iterations of BE in ANNIGMA process and corresponding accuracies also changes. In 3rd iteration (iteration-iii, Table-3) of BE, ANNIGMA finds best accuracies 77.9% with total four features (3, 1, 2, 4) which includes redundant feature-2. The ranking order of features in MaxRel-MinRed is (3, 1, 5, 4, 2, 6). The BE process accuracies is given in Table-III for MaxRel-MinRed which finds best accuracy with four features (3, 1, 5, 4) as in Table-3 (iteration-iii) but this final feature set is different from both ANNIGMA and MR.

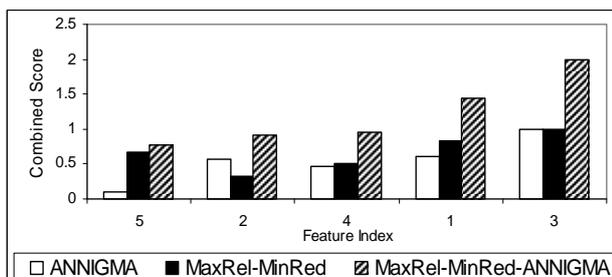


Figure 4. Synthetic Data: The combined score in the 2nd iteration of BE in the hybrid (MaxRel-MinRed -ANNIGMA) when total features is 5. Y-axis gives the combined score and X-axis gives the feature's serial no.

The ranking of features and their score in BE process for (MaxRel-MinRed-ANNIGMA) is given in (Figure 3, 4, 5). In Figure 3, feature-6 has the lowest combined score at first iteration and has been removed after first iteration. In Figure-4, in second iteration, ANNIGMA finds feature-5 as the lowest score, MaxRel-MinRed finds feature-2 as the lowest, however (MaxRel-MinRed – ANNIGMA) finds feature-5 as the lowest. Therefore, feature-5 has been removed after second iteration. In Figure-5, feature-2 is removed after 3rd iteration since the lowest combined score. (MaxRel-MinRed –ANNIGMA) finds the highest accuracy 77.967% (Table-3) with only three features (3, 1, 4) and this final feature set has no (irrelevant or redundant) component and is the correct salient features of the synthetic data set. This shows the significance of hybridization of wrapper and filter approaches in MaxRel-MinRed-ANNIGMA.

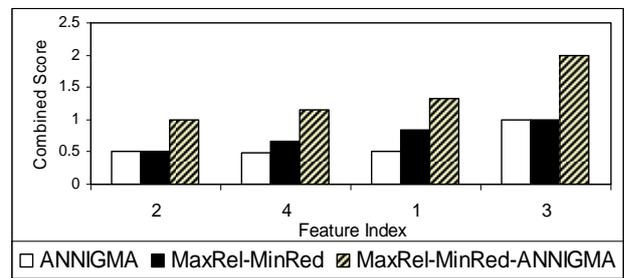


Figure 5. Synthetic Data: The combined score in the third iteration of BE process in the hybrid (MaxRel-MinRed -ANNIGMA) when total features is 4. Y-axis gives the combined score and X-axis gives the feature's serial no.

Total Features	ANNIGMA (%)	MR (%)	MR-ANNIGMA (%)	MaxRel-MinRed (%)	MaxRel-MinRed-ANNIGMA(%)
13	97.528	97.697	97.416	97.578	97.472
12	96.910	96.966	97.921	97.079	97.809
11	97.753	96.517	96.966	96.124	97.079
10	97.022	96.798	96.854	97.360	97.640
9	97.079	97.022	97.472	96.966	97.921
8	97.191	97.640	97.360	97.865	96.517
7	97.697	96.742	96.966	97.360	97.697
6	97.921	97.416	98.034	96.798	97.865
5	96.124	96.348	96.124	97.360	98.315
4	95.225	95.337	95.618	96.517	96.124
3	94.438	94.719	94.888	93.876	93.539
2	90.000	90.618	90.169	91.910	90.393
1	78.933	78.989	78.944	79.270	79.270

Table 5: Accuracies for five algorithms at different iterations of BE process for wine data set.

4.2 Evaluation and experimental analysis of the search process in the hybrids using combined score on Wine data set (Asuncion et al.)

This data set has total 13 real/integer valued attributes with no missing values. The detailed accuracies in different iterations of BE process for wine data set is given in Table-5. The wrapper approach (ANNIGMA) achieves an accuracy of (97.921%) for 6 attributes (7,10,12,13,2,1). The filter-MR achieves accuracy (97.640%) for 8 attributes (7,10,13,12,1,11,6,2) and the filter-MaxRel-MinRed achieves accuracy 97.865% for 8 attributes (7,1, 10,13,11,12,6,5) which is different from final set of MR.

The hybrid process (MR-ANNIGMA) starts with 13 attributes where attribute-8 has the lowest score for ANNIGMA, attribute-3 has the lowest MR score and the hybrid finds attribute-8 as the lowest (Figure 6). Therefore the hybrid eliminates attribute-8 after the first cycle. In the next cycle of BE (Figure 7), the hybrid re-computes all feature's score resulting in attribute-3 attaining the lowest combined score. Therefore it eliminates attribute-3. The hybrid (MR-ANNIGMA) continues BE process and achieves the highest accuracy (98.034%) for six attributes (10,7,1,13,12,11) which is different from final feature set of ANNIGMA. The hybrid MaxRel-MinRed-ANNIGMA achieves the highest accuracy 98.315% for 5 attributes (7,10,1,13,11). Therefore MaxRel-MinRed-ANNIGMA obtains the smallest feature set in all algorithms with the highest accuracy.

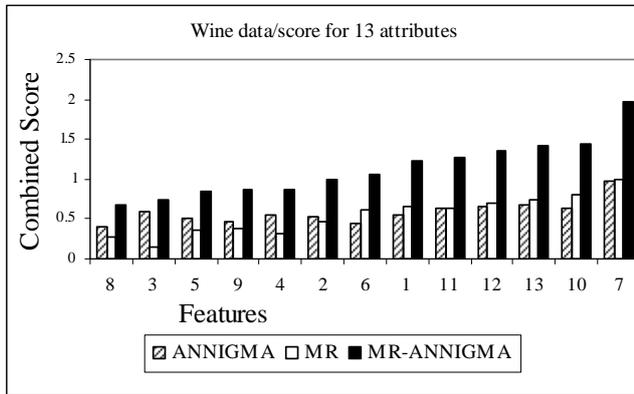


Figure 6. The combined score in the BE search process in the hybrid (MR-ANNIGMA) when total features is 13. Y-axis gives the combined score and X-axis gives the feature's serial no.

4.3 Evaluation and experimental analysis of the search process in the hybrids using combined score on a real life data set: Tobacco Control Policy Evaluation data set (Thompson et al. 2006)

Tobacco Control Policy Evaluation data set: The proposed algorithm has also been tested on a real life data set - "Tobacco Control Policy Evaluation data set". This data set is constructed through International Tobacco Control Policy Evaluation Project (ITC Project) of World Health

Organization (WHO) (ITCEP). ITC completed a four country survey (ITC-4 tobacco data) (ITCEP), (Fong et al. 2005) with a target of estimating the impact of psychological and behavioural impact of the key policies of Framework Convention on Tobacco Control (FCTC) (ITCEP), (Fong et al. 2005), (Thompson et al. 2006), (Heyland et al. 2006) organized by the World Health Organization (WHO). The Four-Country Survey was made among randomly selected smokers in four English-speaking countries: Canada, the United States, the United Kingdom, and Australia. ITC-4 participant smokers are adult who have smoked more than 100 cigarettes in their lifetimes and have smoked at least once in the past 30 days.

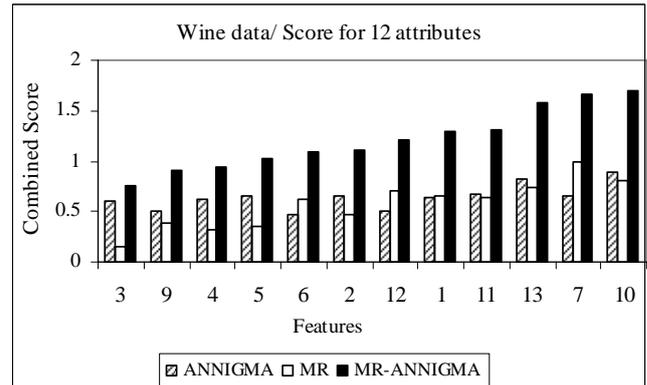


Figure 7. The combined score in the BE search process in the hybrid (MR-ANNIGMA) when total features is 12. Y-axis gives the combined score and X-axis gives the feature's serial no.

The survey consists of four waves. More than seventy five questions have been considered to evaluate the impact of tobacco control policy measures among smoking population. Survey question are mainly based on psychosocial – beliefs about smoking, beliefs about quitting, psychosocial questions such as perceived risk and health worry, smoking behaviour such as total minutes to first cigarette, addictedness to cigarettes), knowledge of health effects/tobacco constituents, socio-demographic questions such income, smokers' reaction and outcome on cessation advice and services, smokers' reactions on warning labels, advertising, monitoring of anti-tobacco campaigns, price/taxation and sources of tobacco, smokers' reactions and effect on smoking restrictions.

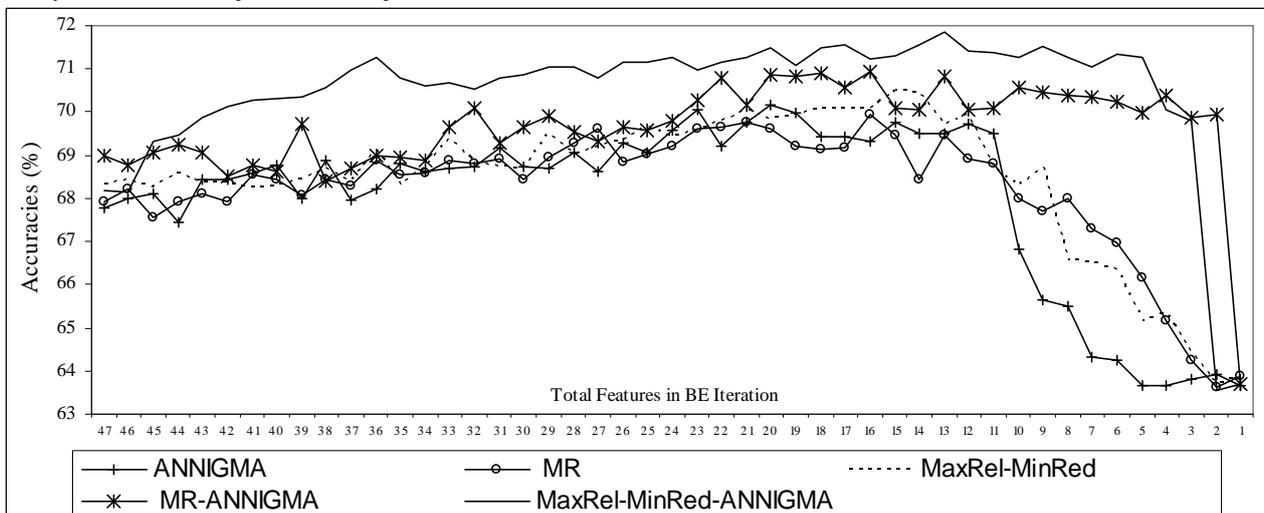


Figure 8. Accuracies in different iterations of BE process in five algorithms for Tobacco Control Policy Evaluation data. Y-axis gives the accuracies and X-axis gives the total number of feature in different iterations of BE process for five algorithms.

The main outcome questions is whether the smokers' have made any attempt to stop smoking since they were interviewed last or they have stopped smoking for about 6 months. We have used here Wave-1 data set of tobacco data. There are 77 attributes in the Wave-1 data set (Integer and Real) with 6682 examples. All five algorithms have been executed on tobacco data. Initially for all algorithms, tobacco data has been pre-processed and the attributes are ordered based on their corresponding heuristic score in descending. Then last 30 attributes with lowest rank are discarded to reduce computational overhead and top-ranked 47 attributes have been used for evaluation of each algorithm. 10-fold cross validation with 10 trials is applied for each of the five algorithms (MR, MaxRel-MinRed, ANNIGMA, MR-ANNIGMA and MaxRel-MinRed-ANNIGMA). The average accuracies over 10 trials for five algorithms for different iterations of the BE process is presented in Figure-8. ANNIGMA achieves 67.793% at 47 attributes. Accuracies over iterations continue to increase up to 70.048% at 23 attributes which is the highest in all iterations. Therefore 23 attributes is the final feature set from ANNIGMA. The filter MaxRel-MinRed achieves accuracy 70.485% for 15 features. It also achieves an acceptable accuracy 70.443% for 14 features. (MR) achieves highest accuracies 69.94% for 16 features. It also achieves an acceptable accuracy 69.45% for 15 features. The hybrid MR-ANNIGMA approach obtains the highest accuracy 70.922% with 16 features. The MR-ANNIGMA also finds a second highest 70.835% for 13 features which is also acceptable. MaxRel-MinRed-ANNIGMA approach obtains the highest accuracy 71.869% with 13 features. The MaxRel-MinRed-ANNIGMA also finds 71.536% accuracy for 9 features which is also acceptable. The results show that hybrid approaches (MR-ANNIGMA, MaxRel-MinRed-ANNIGMA) achieve better accuracies using fewer features than filter or wrapper alone. This demonstrates the significance of the hybridization in the proposed approaches in searching for the most important feature set.

4.4 Evaluation and experimental analysis of the search process in the hybrids using combined score on Ionosphere data set (Asuncion et al.)

This data set has total 34 real valued attributes with no missing values. The detailed accuracies in BE process for the ionosphere data set for all algorithms is described in Table-6. The filter approach (MR) achieves highest accuracy of (91.311%) with 15 attributes and (91.083%) with 11 attributes which is closer to the highest and considered as final feature set of it. The ANNIGMA achieves (90.057%) with four attributes. The filter MaxRel-MinRed achieves the highest accuracy of (91.339%) with 13 attributes. Hybrid MR-ANNIGMA achieves (92.137%) with four attributes. However these final four attributes (5,21,3,6) are different from final feature set (6,24,15,14) of ANNIGMA. The hybrid MaxRel-MinRed-ANNIGMA achieves the highest accuracy of (92.792%) with three attributes (5,6,3) (Table 6 and 7). It is seen that our hybrid approaches achieve the

highest accuracy with very compact feature set (less than five features). However, MaxRel-MinRed-ANNIGMA performs best among all algorithms with highest accuracy (92.792%) and fewest features (3). This proves the significance of the hybrid approaches to select the most salient feature set.

Total Features in BE Iterations	MR	ANNIGMA	MaxRel-MinRed	MR-ANNIGMA	MaxRel-MinRed-ANNIGMA
34	84.188	86.439	84.672	86.011	84.615
33	85.299	87.322	87.094	82.222	83.618
32	84.131	84.615	85.499	84.929	82.279
31	84.986	86.268	86.467	87.607	86.382
30	87.066	85.328	90.256	86.467	89.060
29	88.234	85.613	89.886	86.724	88.091
28	87.521	84.843	88.775	84.501	88.917
27	84.615	86.752	87.464	85.157	85.755
26	87.179	83.789	88.519	85.271	85.726
25	87.350	88.120	88.575	86.724	87.920
24	85.271	83.504	87.578	84.046	86.923
23	86.524	85.556	87.009	83.875	88.661
22	88.946	85.527	87.749	87.436	88.262
21	86.752	86.724	87.578	85.755	88.034
20	85.442	86.524	87.607	84.387	87.151
19	89.687	88.547	88.262	89.829	87.892
18	89.487	89.687	90.228	88.946	88.860
17	90.456	88.063	89.145	87.464	90.969
16	88.405	87.208	89.430	88.718	89.886
15	91.311	87.464	90.798	88.063	91.567
14	89.373	89.744	89.459	89.630	89.772
13	90.684	86.325	91.339	88.917	91.368
12	90.513	88.262	90.085	89.715	90.997
11	91.083	86.524	89.886	89.402	91.823
10	90.627	88.376	89.060	91.197	92.023
9	89.829	88.177	88.860	88.433	91.994
8	89.174	86.980	88.632	89.402	93.020
7	87.037	88.632	90.855	90.712	90.969
6	86.040	90.057	90.256	91.738	93.048
5	88.405	89.829	88.860	91.595	94.330
4	87.407	90.057	90.883	92.137	91.880
3	88.746	89.886	89.715	86.752	92.792
2	87.236	87.977	89.630	87.037	90.197

Table 6: Accuracies for five algorithms at different iterations of BE process for Ionosphere data set (Asuncion et al.).

Table-7 summarizes the final accuracies and number of optimal feature selected for all data sets (described in Table-1) for all algorithms. It shows that the proposed hybrid approaches achieves very compact feature sets in all data sets trialled with higher accuracies than both filter and wrapper alone. This demonstrates that the hybridization of filter and wrapper in the MR-ANNIGMA and MaxRel-MinRed-ANNIGMA lead to improved predictive accuracy with fewer features. However MaxRel-MinRed-ANNIGMA performs better than MR-ANNIGMA and finds smallest feature set with highest accuracies.

Data set		MR (%)	ANNIGMA (%)	MaxRel-MinRed (%)	MR-ANNIGMA (%)	MaxRel-MinRed ANNIGMA (%)	Other (%)
Wine	Accuracy	97.640	97.921	97.865	98.034	98.315	98.2 (Huang et al. 2008)
	Total Features	8	6	8	6	5	
Ionosphere	Accuracy	91.311	90.057	91.339	92.137	92.792	92.51 (Huang et al. 2008)
	Total Features	15	6	13	4	3	
Cancer (Diagnostic)	Accuracy	96.148	96.287	96.92	97.065	97.71	94.9 (II-Seok et al. 2004)
	Total Features	21	14	17	16	15	
Sonar	Accuracy	83.506	83.606	83.073	84.236	84.594	83.4 (Optiz et al. 1999)
	Total Features	17	40	12	16	15	
Pima	Accuracy	76.95	76.71	77.04	77.17	77.173	77.0 (Hsu et al. 2002)
	Total Features	{8,2,6}=3	{2,6,7,1,5}=5	6	{7,6,2}=3	{7,6,2}=3	
Tobacco	Accuracy	69.45	70.048	70.443	70.835	71.536	
	Total Features	15	23	14	13	9	

Table 7 Detailed accuracies for five algorithms and number of features in final feature sets for all data sets.

4.5 Computational Performance

The hybrid algorithms runs a backward elimination (BE) process where each iteration involves computational time in training the network, the computation of MR, MaxRel-MinRed score, ANNIGMA and hybrid score. Computation of MR score and ANNIGMA has linear time complexity in terms of feature dimensionality. At the beginning when all features are used, the time for training and computing scores (MR, ANNIGMA, and hybrid) would be the highest. Subsequent computation will take less time. Computational performance has been described in Table-8. The experimental platform was 3.2-GHz Pentium-4 CPU with 1GB of RAM. Table-8 shows that MaxRel-MinRed-ANNIGMA takes more time than MR-ANNIGMA.

Data Set	MR (Hrs)	ANNIGMA (Hrs)	MaxRel-MinRed (Hrs)	MR-ANNIGMA (Hrs)	MaxRel-MinRed-ANNIGMA (Hrs)
Synthetic	0.1219	0.1208	0.1225	0.1239	0.1246
Wine	0.1021	0.1304	0.1114	0.1428	0.1479
Ionosphere	0.8223	0.840	0.8261	0.8545	0.9081
Cancer	0.8543	0.8012	0.8745	0.8731	0.8834
Sonar	0.9014	0.9415	0.9124	0.9512	1.0972
Pima	0.2214	0.214	0.2573	0.2421	0.2588
Tobacco	4.241	4.520	4.631	5.342	5.459

Table 8: Computational time (Hours) for five algorithms for all data sets.

5 Conclusions

This paper proposes two novel hybrids of wrapper and filter approaches for input feature selection problem. The novelty of our approaches is that these integrate knowledge (from the intrinsic characteristics of data) obtained by the filter approach into the wrapper approach and combines the wrapper’s heuristic score with the filter’s ranking score in the wrapper stage of the hybrid.

To the best of our knowledge, the idea of our approach is new and has not been explored yet in the literature. The first proposed hybrid combines a mutual information (MI) based Maximum Relevance (MR) filter ranking heuristic with an Artificial Neural Network (ANN) based wrapper approach where Artificial Neural Network Input Gain Measurement Approximation (ANNIGMA) has been combined with MR (MR-ANNIGMA). The proposed second hybrid algorithm combines an improved version of MR (Maximum Relevance and Minimum Redundancy; MaxRel-MinRed) filter ranking heuristic with the ANNIGMA (MaxRel-MinRed-ANNIGMA). The combined heuristics in the hybrids: (MR-ANNIGMA and MaxRel-MinRed-ANNIGMA) take the advantages of the complementary properties of the both filter and wrapper heuristics and guide the wrapper to find optimal and compact feature subsets in the wrapper. The approaches have been tested using synthetic data, bench mark machine learning data sets and real life-Tobacco Control Policy Evaluation data sets with varying number of features and sample size. Our experiments show that hybrid algorithms (MR-ANNIGMA and MaxRel-MinRed-ANNIGMA) ranks the features in such a way that the internal BE process of the wrapper step generates better subsets of features than both filter and wrapper approaches in terms of wrapper evaluation criteria and achieves higher accuracies and smaller feature sets than both filter and wrapper approaches. However, MaxRel-MinRed-ANNIGMA outperforms all other algorithms. In the future we will use other search strategies such as bidirectional search with the proposed approaches and then will evaluate the approaches on the rest of the Waves’ data of tobacco control data as well as other bench mark data sets.

6 References

Blum, A.L, and Langely,P, “Selection of relevant features and examples in Machine Learning”, Artificial Intelligence, Vol 69, pp 245-271,1997

- John, G.H., Kohavi, R. and Pfleger, K, "Irrelevant Feature and the subset selection problem", Proc. Of 11th Int. conference on Machine Learning, pp 121-129, 1994
- Dash, M. and H. Liu, "Feature selection for classification", Intelligent data analysis: An International Journal, vol-1, no-3, pp 131-156, 1997
- Puronnen,S., Tsymbal,A. and I. Skrypnik, "Advanced local feature selection in Medical Diagnostics", Proc. 13th IEEE symp. computer-based medical diagnostics, 2000.
- Bne-Bassat,M., "Pattern recognition and Reduction of dimensionality", Handbook of Statistics-II, P.R. Krisnaiah and L.N. Kanal eds. Pp 773-791, 1982
- Mitra, P., Murthy, C.A., and S.K. Pal, "Unsupervised Feature selection using Feature similarity", IEEE Transaction on Pattern Analysis and Machine Intelligence, vol 24 pp 301-312, March 2002.
- Kwak,N., and C. Choi, Member, "Input Feature Selection by Mutual Information Based on Parzen Window", IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 24, No. 12, December 2002
- Wang,H., Bell,D., and F. Murtagh, "Axiomatic Approach to Feature Subset Selection Based on Relevance", IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 21, No. 3, March 1999
- Hall,M.A., "Correlation-Based Feature Selection for Discrete and Numeric Class Machine Learning," Proc. 17th Int'l Conf. Machine Learning, pp. 359-366, 2000.
- Kwak,N., and C. Choi, "Input Feature Selection for Classification Problems", IEEE Transaction on Neural Networks, Vol. 13, No. 1, January 2002
- J.G. Dy and C.E. Brodley, "Feature Subset Selection and Order Identification for Unsupervised Learning," Proc. 17th Int'l Conf. Machine Learning, pp. 247-254, 2000.
- Hsu,C.N., H.J. Huang, and D. Schuschel, "The ANNIGMA-Wrapper Approach to Fast Feature Selection for Neural Nets", IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics, Vol. 32, No. 2, April 2002
- Zhu,Z., Y. S Ong, M. Dash, "Wrapper-Filter feature selection algorithm using a memetic framework", IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics, Vol. 32, No. 2, April 2002
- Kohavi,R., and G.H. John, "Wrappers for Feature Subset Selection," Artificial Intelligence, vol. 97, nos. 1-2, pp. 273-324, 1997
- II-Seok Oh, Ji-Seon Lee and Byung-Ro Moon, "Hybrid genetic algorithms for Feature selection", IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 26,2004.
- Jain,A., and D. Zongker, "Feature selection: Evaluation, Application and small sample performance", IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 19, No. 2, pp 153-158, Feb 1997.
- Ferri,F.J, P. Pudil, M. hatef and J.Kittler, "Comparative study of techniques for large-scale feature selection", Pattern recognition in practice IV, E.S. Gelsema et.al. eds pp 403-413, 1994
- Asuncion, A. and Newman, D. J.. UCI Machine Learning Repository Irvine, CA: University of California, School of Information and Computer Science. [<http://www.ics.uci.edu/~mllearn/MLRepository.html>].
- Optiz,D., and R. Maclin, "Popular Ensemble methods: An Empirical study," Journal of Artificial Intelligence Research, vol 11, pp 169-198,199
- Huang,J.J., Y.Z Cai and X.M. Xu, "A parameter less feature ranking algorithm based on MI", Neurocomputing, (Vol-71), 2008,pp 1656-1668.
- Kurgan,LA., et al., "CAIM discretization algorithm", IEEE Trans. on Knowledge and Data Engineering, 26, 2004, pp 145-153.
- Peng,H., C. Ding and F. Long, "Minimum Redundancy-Maximum Relevance Feature selection", IEEE intelligent Systems, Nov 2005, pp 70-71.
- ITCEP,International Tobacco Control policy Evaluation Project (ITCEP), <http://www.itcproject.org/>. WHO, 2008, WHO REPORT on the global TOBA CCO epidemic, 2008, The MPOWER package
- Fong,G.T, K. M. Cummings, R. Borland, G. B. Hastings, P. Hyland, G. A. Giovino, D. Hammond, and M. E. Thompson, "The conceptual framework of the International Tobacco Control (ITC), Policy Evaluation Project," Tobacco Control, vol. 15, Suppl. 3, pp. 3-11, 2005.
- Thompson,M.E., G. T. Fong, D. Hammond, C. Boudreau, P. Driezen, P. Hyland, R. Borland, K. M. Cummings, G. B. Hastings, M. Siahpush, A. M. Machintosh, and F. L. Laux, "Methods of the International Tobacco Control (ITC) Four Country Survey", Tobacco Control, vol. 15, Suppl. 3, pp. 12-18, 2006.
- Hyland,A., R. Borland, Q. Li, H-H. Yong, A. McNeill, G. T. Fong, R. J. O'Connor, and K. M. Cummings, "Individual- level predictors of cessation behaviours among participants in the International Tobacco Control (ITC) Four Country Survey", Tobacco Control, vol. 15, Suppl. 3, pp. 83-94, 2006.

Web-Based Visualisations Supporting Rehabilitation of Heart Failure Patients by Promoting Behavioural Change

Susanne Fischer^{1,2} Burkhard C. Wünsche¹ Linda Cameron³
 Eva Rose Morunga³ Umang Parikh¹ Lana Jago³ Stefan Müller²

¹ Graphics Group, Department of Computer Science
 University of Auckland, New Zealand,
 Email: burkhard@cs.auckland.ac.nz, upar006@aucklanduni.ac.nz

² Institute für Computervisualistik, Department of Computer Science
 University of Koblenz-Landau, Germany
 Email: sanne@uni-koblenz.de, stefanm@uni-koblenz.de

³ Department of Psychology
 University of Auckland, New Zealand,
 Email: l.cameron@auckland.ac.nz, emor029@aucklanduni.ac.nz, l.jago@auckland.ac.nz

Abstract

Heart failure is a major cause of death in the Western world and has a severe impact on the functioning status of individuals suffering from it. Self care management involves behaviour and lifestyle changes that reduce the negative effects of this chronic illness. We have developed web-based visualisations for educating patients and promoting behavioural change. The tool uses interactive web graphics to visualise relationships between lifestyle, symptoms, patient parameters and the disease. The goal is to empower and motivate patients to take more control of the disease management process. In contrast to many educational websites with interactive content our application utilises real patient parameters and it has been evaluated from a health psychology perspective.

Initial usability studies revealed difficulties with the platform independent design, but that overall the tool was perceived as educational and easy to use. A more detailed study explored the responses of Māori individuals with heart failure and their whanau to a patient education intervention using the tool. Semi-structured interviews, guided by the Common Sense Model of illness representations, demonstrate that the programme promotes knowledge and understanding of the illness and its associated symptoms and promotes protective behaviour.

Keywords: visualisation, web graphics, heart failure, chronic disease management, telehealth, behavioural change

1 Introduction

Cardiovascular diseases are a leading cause of death in New Zealand (Hay 2004) and the Western world (Lopez et al. 2006). Heart failure is a common chronic form of it. Previous research indicates that many heart failure patients have a poor understanding how their lifestyle influences their cardiac health and rehabilitation (Horowitz et al. 2004). Medication and changing lifestyle can help the patient to live a normal life, but many patients do not have a good

understanding how they can control their disease due to the complex relationship between disease, medication, symptoms and lifestyle choices. As a result many patients do not adhere to rehabilitation procedures. Examples are reduced adherence in medication usage and adverse lifestyle choices such as smoking, lack of exercises, and a high-salt diet. This lengthens the rehabilitation process, increases the risk of another cardiac event, adds demands on health care staff, and overall increases health care costs (Horowitz et al. 2004).

In this paper we present and evaluate the use of visualisations of patient parameters for improving patients' understanding of their disease and increasing their level of control over the rehabilitation process.

Section 2 summarises important facts about heart failure impacting the design of our solution. Section 3 reviews the literature in this field and discusses related applications. Section 4 analyses the problem and derives design requirements. The design of our solution and implementation details are explained in sections 5 and 6. Section 7 presents a usability study and a detailed patient study demonstrating the effectiveness of our application. We conclude this paper in section 8 and give a short overview of future work.

2 Heart Failure

Heart failure is a cardiovascular disease characterised by the heart being unable to pump sufficient blood to meet the needs of the body (Fuster et al. 2008). The disease can result from any structural or functional cardiac disorder, which impairs the ability of the ventricle to fill with or eject blood (American College of Cardiology/American Heart Association Task Force on Practice Guidelines 2005). The term "congestive heart failure" is frequently used instead of "heart failure" because it often leads to fluid retention. But because not all individuals with heart failure complain of oedema at the time of initial or subsequent evaluation, the term "heart failure" is preferred (American College of Cardiology/American Heart Association Task Force on Practice Guidelines 2005). Almost any form of cardiac disease can produce heart failure, but the most common cause in Western society is coronary artery disease (Timmis & McCormick 2003).

The impact of heart failure, due to its increasing prevalence, is viewed as a significant economic and societal burden (Masoudi et al. 2002). In 2006, the

Copyright ©2011, Australian Computer Society, Inc. This paper appeared at the 34th Australasian Computer Science Conference (ACSC 2011), Perth, Australia, January 2011. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 113, Mark Reynolds, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

cost of heart failure in the USA was estimated at approximately \$29.6 billion (Doughty & White 2008).

2.1 Symptoms

Most symptoms of heart failure are caused by fluid retention and reduced oxygen supply. Symptoms caused by fluid retention, such as shortness of breath, weight gain and swelling in the feet or ankles, can be regulated by medication and reduced salt and fluid intake. Symptoms caused by reduced oxygen supply are confusion and fatigue and weakness, the latter two are especially prevalent during physical activity because of inadequate oxygen delivery to the muscles (Timmis & McCormick 2003).

Symptoms can vary widely and the “Rotterdam study” showed that sixty percent of individuals with left ventricular systolic dysfunction did not have symptoms or signs of heart failure (Mosterd et al. 1999). Furthermore attempts to diagnose heart failure in the elderly population may be difficult as typical signs and symptoms may be masked (Masoudi et al. 2002).

2.2 Heart Failure Self Care

Heart failure self care refers to specific behaviours that individuals initiate and perform on their own behalf, with the intention of improving health, preventing disease or maintaining their well-being. Individuals with heart failure can assist the treatment process by taking medication, adhering to dietary requirements such as low sodium intake, exercise, restricting smoking and the intake of alcohol, and monitoring for signs and symptoms of heart failure (Lopez et al. 2006, Lee et al. 2009).

Guidelines on warning signs and management of heart failure have been published for patients and health care professionals (National Heart Foundation of New Zealand 2010). Significant parameters influencing heart conditions are age and gender, weight, blood pressure, pulse rate, exercise, alcohol, smoking and salt intake. All of these parameters can be recognised or measured by the individuals with heart failure and are hence useful for patient education and disease monitoring and management.

Four key barriers to heart failure self care have been identified: medication adherence, dietary adherence, symptom monitoring and decision making (Riegal et al. 2009). Comorbidities, such as diabetes, can make it difficult to follow dietary guidelines and monitor symptoms. Additional important factors are poor health literacy and problems with the healthcare system, which often neither encourages nor supports self care by individuals.

3 Literature Review

A growing body of evidence supports the illness cognition and behaviour processes delineated by the Common-Sense Model of self-regulation (Cameron & Leventhal 2003, Hagger & Orbell 2003). In particular, research supports the crucial roles of illness representations, or mental schema that include images as well as abstract beliefs about symptoms, consequences, and actions for managing the illness conditions. Mental images within an illness representation play a powerful role in shaping beliefs; enhancing a sense of coherence or understanding of one’s condition; forming “IF-THEN” connections among symptom experiences, representational beliefs, and recommended actions; and motivating adherence to treatment (Cameron & Chan 2008). Using visual images in educational programmes can thus have a

powerful impact on representational beliefs, coherence or understanding, and behavior. Pictures have been found to have longer lasting effects on memory than words (Gardner & Houston 1986) and their effects can be positive, such as encouraging the pursuit of goals, or negative, such as motivating avoidance behaviours (Bradley & Lang 1999). Embedding visual images within an appropriate narrative structure is therefore critical for ensuring that the images motivate behaviours as intended. A well known example of using images in the medical field to effectively influence behaviour is the use of graphic warning labels on tobacco packs (Hammond et al. 2003). The Common-Sense Model served as the theoretical framework guiding the development of the HEART MENTOR Programme.

3.1 Web-based Educational Tools for Individuals with heart failure

A large variety of web-based resources exist for educating and supporting individuals with heart failure. The American Heart Association website uses videos to explain symptoms of heart failure and what to do when they get worse (American Heart Association Inc. 2010b). The associated “Heart Failure Media Library” contains images and Adobe Flash animations that explain heart failure related topics. The “Heart360™” site, powered by Microsoft’s Health Vault, allows patients to manage health information, evaluate health parameters, and to obtain recommendations and reports for visiting their healthcare provider (American Heart Association Inc. 2010a).

The National Heart Foundation of New Zealand website delivers comprehensive information about the heart’s function, how to prevent heart disease, how symptoms of heart failure are caused and how to do lifestyle changes (National Heart Foundation of New Zealand 2010). It uses predominantly text and 2D graphics. Brochures can be downloaded and targeted information for cardiac rehabilitation is available to enable individuals with heart failure to return to the most normal life possible after a cardiac event. This includes helping them to take control of their life and improving their quality of life by means of education, information, physical activities and social support.

While many educational websites have interactive content, the visualisations do not reflect patient parameters and hence there is little identification of patients with the visual content they perceive. Furthermore no evaluations of these sites’ effectiveness exist.

3.2 Patient Specific Visualisations

To improve the rehabilitation of individuals with heart failure, Rubin et al. developed a tool with an embedded animated 3D heart model (Rubin et al. 2005). The 3D heart model is an adaptation of a 3D finite element model of the mechanical and electrical behaviour of a porcine heart developed by the Bioengineering Institute of the University of Auckland (Hunter & Pullan 2002). The animated model is integrated into a web site using the ZINC plug-in and adapts to user input such as heart rate, state of the arteries (e.g., no plaque), and structure and number of capillaries. Changes to the heart, e.g., due to exercises and a healthy diet are illustrated by changes to the model, images, pop-up windows with explanations and videos. A disadvantage of the application is that it only runs in the Mozilla Firefox browser, because of the need of the ZINC plug-in.

Lee et al. used this model for studying various forms of heart disease education. The authors found that imagery content is superior to text-based content

and leads to increased representational beliefs and mental imagery relating to heart disease, worry, and intentions at post-intervention (Lee et al. 2010). Increases in sense of coherence (understanding of heart disease) and worry were sustained after 1 month. The imagery contents also increased healthy diet efforts after 2 weeks.

This research builds on top of the above work and replaces the Finite Element heart model with a simplified polygonal model suitable for popular web graphics technologies. We also integrate a 2D body model and provide comprehensive inter- and intra-element communication to improve understandings of the complex causal relationships between symptoms, disease, medication and lifestyle.

4 Problem Analysis and Design Requirements

4.1 Requirement Analysis

The main purpose of our application is to educate and motivate the patient using visual content conveying the effect of lifestyle choices and the relationship between them and medication, symptoms and disease. The requirements for the application are divided into visualisation requirements, web-based requirements, interface requirements and structural requirements.

4.1.1 Visualisation Requirements

The visualisations must be easy to perceive, informative, and convey relationships between disease specific parameters, e.g., “If you take this medication, then these symptoms will be relieved”. This might necessitate abstract and symbolic visualisations since many symptoms and medication effects do not have obvious physical manifestations. The patients acceptability of the animation plays a significant role (Bradlyn et al. 2003). Adding new associations can enhance message effectiveness (Lee et al. 2010). Most individuals with heart failure do not have a vivid or detailed picture of their hearts (Broadbent et al. 2004). To improve the perception the visualisation should be as simple as possible, while still looking professional, and retaining enough details to display functional and anatomical changes.

Most available medical brochures contain non-photorealistic images, which omit non-essential information and reduce the complexity of the content. Non-photorealistic visualisations can be more effective for communicating specific information than photographs or photorealistic visualisations (Strothotte & Schlechtweg 2002). The freedom to modify other parameters such as colour, texture, and highlighting details, allows us to visually convey health or illnesses. The heart model should be animated in order to emphasise its crucial pumping action. A 3D model is preferred because this improves patient identification with the model and makes perception of the pumping behaviour easier compared to using a 4-vessel cross section (ventricles and atria).

Section 2 demonstrated that many symptoms and medications affect other organs than the heart, and hence a whole body visualisation is necessary. In order to simplify perception of parameters, and because of users’ familiarity with them, we prefer a schematic 2D frontal graphic of the body similar to illustrations in medical brochures or biology school books. Organs that play a decisive role in heart failure must be shown within the body visualisation and must be identifiable by the user.

4.1.2 Web-Based Requirements

To make the application easily accessible it must be web-based. The web application should provide a consistent experience with immediate responses and real-time interactions. In order to avoid problems due to latency and bandwidth (many users will have dial-up connections) client side computations should be used where possible. Other desirable features are scalability (for running on a PDA) and data base support.

Of particular importance is the choice of 2D and 3D graphics technologies. We used and extended an evaluation framework by Holmberg et al. (Holmberg 2006, Holmberg et al. 2006) and identified the following main requirements:

Technical Capabilities

Communication with servers is necessary for loading visualisations and accessing databases. Compressed communication is of low importance since the amount of transferred data is likely to be small. Encrypted communication is relevant for future work when real patient data will be collected and stored. For the 2D visualisation animations of geometry and visual attributes are essential. For the 3D visualisation we want to have texture mapping and real-time animation of heart geometry in order to display patient parameters and increase patients’ identification with the model.

Community Support

The selected technologies should be platform independent, standardised and widely supported. Downloads and installation of plug-ins should be avoided and if necessary made as easy as possible. The application should be compatible across major browsers and operating systems to reach a wide range of audience. The cost and availability of development tools is important to ensure fast and cost-efficient development.

Interactivity

Interaction is important to emphasise the relationships between disease, symptoms and lifestyle, e.g., to demonstrate changes in patient conditions for different lifestyle choices. The 3D heart model should be rotatable in order to appreciate its function and the effects of stress. The 2D body visualisation should support selection of organs and display more information about them and their role in heart failure.

Script support is essential to work predominantly on the client computer, and support should be provided for creating new content and inserting it into the existing scene. Inter-element communication is essential when the web application uses different technologies.

Application Specific

Native graphical structures are good but not essential. It must be possible for the 2D visualisation technology to create a “complex” graphics like a body shape. Hence, the option to use parametric curves is beneficial. The chosen 3D visualisation technology should support loading and animating 3D models.

4.1.3 Structural Requirements

In order to communicate logical “IF-THEN” statements that connect concepts, experiences or beliefs it is helpful to represent them as “narrative” (Meadows 2006). The narrative provides a structure,

helps retaining attention, enhances or recalls memories (Simpson & Barker March 2007), and enables the user to identify with story characters (Kolko 2010). Furthermore, the narrative can project into the future and emphasise options and choices.

To create such a narrative the key relationships between components have to be identified by capturing dependencies. It is important to consider appropriate start and end points and to display data in the most relevant context. The connections between in- and output must be apparent so that users easily understand the meaning of the content and where to navigate from there.

4.1.4 Interface Requirements

The user interface should be designed like an interactive storyboard in order to support the narrative, facilitate navigation, and increase impact on patient behaviour. Various other supporting media can be considered to convey messages. For example, we want to use videos to demonstrate positive lifestyle choices such as patient specific exercises. The media integration should improve usability and intensify conveyed messages. Explanatory text is essential for parameters which can not be visualised (e.g., likelihood of disease) or to prevent ambiguities (Lee et al. 2010). Sound can also convey health or illness (e.g. heart rate, breathing, coughing). The tone or volume can be changed to reflect patient conditions.

4.1.5 Functional Requirements

The application should, inform, motivate and empower the patient. “Inform” means answering common questions such as

- What is heart failure?
- What are the Symptoms of heart failure?
- What are the causes of these symptoms?
- How can lifestyle influence the condition?

“Motivate” means increasing adherence, e.g., using medication, and encouraging users to follow rehabilitation procedures such as diets and exercises. “Empower” means that patients recognise their ability to influence their condition and improve symptoms. This is possible with lifestyle changes such as reduced salt and alcohol intake, more exercise and quitting smoking. Furthermore patients can assist doctors by being proactive and looking out for warning signs such as swellings and sudden weight increase.

From consultations with cardiologists and guidelines for patients and health care professionals (National Heart Foundation of New Zealand 2010) we identified the following parameters relevant for heart failure diagnosis, monitoring and rehabilitation.

- Age and Gender influence the risk of developing heart failure. These risk factors can not be controlled by the patient and hence are not significant for the intended goals of the visualisation.
- Rapid weight gain can be a sign of a worsening heart condition. Since weight is also an important measure of diet and lifestyle, this input parameter is essential.
- Blood pressure data is of limited use for our application since few patients in New Zealand have the necessary monitoring equipment.

Patient parameter	Visual representation
weight	body contour
fluid retention	enlarged body contour
fluid in the lung	blue lung
swollen feet	enlarged feet contour
reduced blood circulation	transparency of arteries increases farther from heart
confusion	thought balloon with symbols
fatigue	thought balloon with “zzz”
weak posture	hanging shoulder and head

Table 1: Visualisation of patient parameters.

- The pulse is easy to measure and visualise, but care must be taken that no wrong conclusions are drawn by pulse rate changes due to other factors such as stress and physical activity.
- In order to improve patient outcomes we want to promote physical activities. Hence, exercise is an essential parameter.
- Alcohol intake is an important component of patient behaviour and requires monitoring and control.
- Heart failure patients are strongly advised to quit smoking and hence this parameter needs monitoring.
- Patients’ salt intake is a significant parameter in heart failure (high blood pressure), but is difficult and inconvenient to measure quantitatively. Preliminary interviews suggested that recording the number of fast food meals and snacks consumed would be unreliable due to forgetfulness and different interpretations (what is a “snack”?). We hence use a qualitative measure by recording whether patients add extra salt to their food.

5 Design

5.1 Visualisations

Based on the identified symptoms and affected organs, the most relevant components of the 2D body visualisation are the heart, the lung, the arteries and the body contour. Table 1 explains how different patient parameters are displayed. When displaying conditions it is important to first show a healthy body or the current situation for comparison. Changes to the 3D heart model include changes in the heart beat rate and enlargement to indicate cardiac remodelling when load is increased (high blood pressure).

5.2 Structure and Content

A major design aspect of the application is to visually represent key relationships between symptoms, patient behaviour and the disease. Based on the previously discussed information the following tables were developed. Table 2 presents symptoms and the corresponding patient parameters represented in the body visualisation. Table 3 presents patient behaviours (lifestyle choices) and the affected patient parameters.

As discussed in subsection 4.1.3 a clearly structured narrative is crucial for the effectiveness of the application. We choose a linear structure which is centered around the 2D body and 3D heart visualisation. Text is predominantly displayed in the text section of the web page as indicated in figure 6. More detailed explanations are displayed in pop-up windows. This simplifies the linear structure and gives the user the option to skip detailed explanations.

Symptom	Patient parameter
breathlessness	lung, circulation
weight gain	weight, circulation
swollen feet	swollen feet, circulation
fatigue	weak posture, fatigue, circulation
confusion	confusion, circulation
weakness	weak posture, circulation

Table 2: Mapping of symptoms to patient parameters.

Behaviour	Patient parameter
high calorie diet	weight
exercise	weight, circulation
salt intake	swollen feet, weight, circulation
smoking	lungs, circulation
alcohol intake	lungs, weight, circulation

Table 3: Mapping of patient behaviours (lifestyle choices) to patient parameters.

The narrative starts with an explanation of symptoms of heart failure. This is followed by information about weight monitoring since this is one of the most essential measures for patients to observe. The patients should understand that rapid weight gain in a couple of days has to be reported to their healthcare provider. Explanations to weight and diet related items, such as exercise and salt intake, are followed by information about smoking and alcohol intake. The narrative ends with information about cardiac rehabilitation.

6 Implementation

6.1 Technology Choices

Web graphics technologies were evaluated using the framework presented in (Holmberg 2006) and the requirements summarised in section 4.1.2. For the 2D visualisation both Adobe Flash and SVG were identified as possible solutions. At the start of this research Adobe Flash needed a plug-in, whereas SVG only needed one for the Internet Explorer and all other major browser were natively supported. Further advantages are the text based nature of SVG and free tools for graphical editing, such as Inkscape (Inkscape 2010). We hence decided to use SVG for the 2D body graphics.

For the 3D content JOGL was chosen and embedded using a Java applet. The technology is a Java binding of the popular OpenGL API which facilitates the integration and animation of different 3D model formats. It allows the use of vertex and fragment shaders for rendering and animation and Java applets are supported by all major browsers. JOGL does not need to be installed on the client computer such as Java3D. Eclipse is used as development tool.

Besides the visualisation technologies, HTML, CSS and PHP are used to create the user interface. JavaScript is applied for the client side scripting, because both technologies have the ability to work with it. The Document Object Model (DOM) offers the access to elements of the SVG and the Java applet, so that user input through the user interface can influence the visualisations.

6.2 2D Visualisation

Figure 1 (a) shows the healthy body contour with the arteries, the lung and the heart. Each element is described in SVG with a <path>-tag which characterises a Bézier curve and its style (colour, opacity,

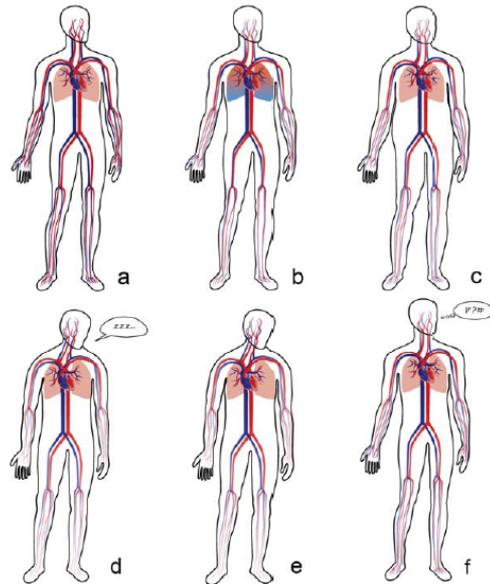


Figure 1: Body visualisations. a: healthy, b: breathlessness, c: weight gain, d: fatigue, e: weakness, f: confusion.

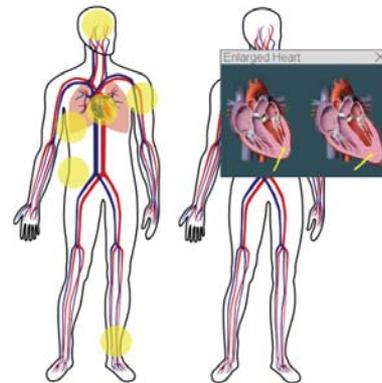


Figure 2: Highlighted body regions (left) and SVG pop-up window within the body graphic (right).

etc.). The arteries and veins are represented in red and blue to indicate oxygen rich and poor blood, respectively. The lung is pinkish in the normal state and becomes blueish when fluid build-up is represented. Insufficient blood supply to the body is indicated by increasing the transparency of arteries with increasing distance from the heart.

6.2.1 Intra-Element Interaction

SVG supports four animation elements which are defined in the SMIL Animation specification (W3C 2010):

- <animate> allows scalar attribute animation over time
- <set> sets a value of an attribute for a specified duration
- <animateMotion> moves an element along a motion path
- <animateColor> modifies the colour value of particular attributes or properties over time

The animation can be activated through a mouse click, mouse motion or keyboard input. Since browser support of these functionalities is limited we use JavaScript to animate and interact with the graphics.

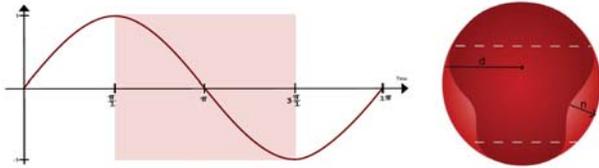


Figure 3: Section of a sine curve used to animate the model in order to simulate non linear deformation during a heart beat (left) and a schematic representation of the resulting deformation (right).

Intra-element interaction is used to highlight specific areas of the body (yellow dots in figure 2) when the mouse moves over them. A click opens a small window using an SVG Window Object (Andreas Neumann and Andre M. Winter 2010) and a JPEG image is loaded by creating a new SVG element.

6.2.2 Inter-Element Communication

Inter-element interaction describes the communication between SVG and other elements of the application. The user input is integrated via a HTML form. With every input a JavaScript function is called to change the SVG graphics using access via the Document Object Model. The body contour, for example, is modified through

```
svgdoc.getElementById(body).setAttribute(d,x)
```

The SVG document is searched for the element “body” and the path attribute ‘d’ is changed into the new value ‘x’. The modified contour is illustrated in parts (c)-(e) of figure 1. In a similar way it is possible to edit the colour of the SVG element. For example, for the lung in figure 1 (b) a colour gradient is used when the symptom “breathlessness” is chosen.

6.3 3D Heart Model

JOGL is supported since Java Runtime Environment version 1.4.2. The JNLPAppletLauncher downloads native code on the fly and allows the use of JOGL via an applet without the need to be signed or an additional installation of software on the client computers. The user will receive a security dialog to accept the certificate for the JLNPApplletLauncher.

The applet reacts to user input through a JavaScript function call. The JavaScript function can call functions in the Java application via DOM:

```
document.getElementById("HeartApplet").  
getSubApplet().function();
```

This ability to use JavaScript and to call functions enables loading new content in response to user input. MAYSCRIPT has to be named in the <applet>-tag to give access to the JavaScript of the web page. JOGL does not natively support loading of 3D models, but there are some implementations of OBJ file loader for Java. We modified one from (Andrew Davison 2010) to work within applets.

6.3.1 Heart Animation

In order to simulate the heart deformation we analysed a heart model obtained from computed tomography images (Wünsche & Young 2003). The hearts motion is best described as wringing motion. The heart surface seems to have the strongest perceived deformation around the center section and towards the apex. We approximate this motion by displacing

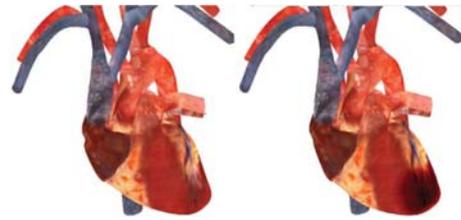


Figure 4: Heart model without (left) and with (right) multitexturing to indicate an infarcted region.

the surface of a polygonal 3D heart model using the function

$$\hat{v} = v + n \sin(\alpha) a |\sin(\text{time})|$$

where n is the surface normal at the vertex v and a is a scale factor. The sine function with the time argument varies between 0 and 1, which corresponds to the maximum contraction and expansion of the heart. The second sine function with the parameter α has the effect that the displacement depends on the position with respect to the centroid of the heart. The displacement is zero for vertices above and below the white lines in figure 3 (right) and is maximal at the centre section of the model. The vertex displacement formula is efficiently implemented as a vertex shader using GLSL (OpenGL Shading Language) and has no measurable effect on rendering speed.

6.3.2 Texture Mapping

Multitexturing is used to create an infarct region as illustrated in figure 4. The shader computes new texels by subtracting the infarction texture from the heart texture (see figure 4). Note that in reality infarcted and healthy tissue are visually indistinguishable. The visualisation symbolises heart damage caused by bad life style choices.

6.4 User Interface

The user interface consists of three panels: one for visualising the body, one for the heart, and one for user input and navigation. The user input consists of HTML form elements such as text entry and selection lists. Input values change the body and heart visualisation via JavaScript functions.

Before the application starts a page with an SVG and a Java applet is shown to test whether appropriate plug-ins are installed. Links to the appropriate downloads are provided (see figure 5). The first page of the application shows visualisations of a healthy heart and body. The drop-down box for selecting symptoms is initially set to “healthy”.

7 Results

7.1 Usability Study

The usability of the application was tested using a web-based survey of technical and user interface aspects (Fischer 2009). A total of 23 participants from Germany and New Zealand took part in the survey. 16 participants (nine females and seven males) completed the survey and 7 did not because of technical difficulties. Twelve participants were between 20 and 30 years old, three participants between 30 and 40 years old, and one was in the 40-60 age group. Half of the participants were native English speakers. Only one person was not a university student, the remaining participants studied psychology (8), computer science (4), medicine (1), and other fields (2).

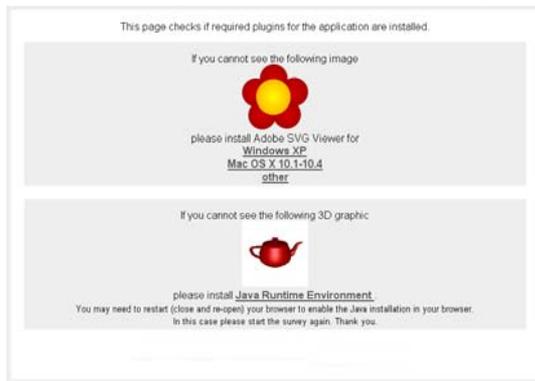


Figure 5: Test and download page for required plugins.

7.1.1 Technical Aspect

Technical problems were predominantly dependent on the Operating System. The 12 participants, who used Windows XP finished the whole survey. Half of them used Mozilla Firefox 3 and did not need a plug-in for SVG. The other half used Internet Explorer 7 and had to install the plug-in. Five of the 12 participants could not see the Java applet, but it worked after installing the Java Runtime Environment. “Not my computer” was the only reason mentioned for not install the SVG Viewer or the Java Runtime Environment.

During the development we became aware that Windows Vista and Internet Explorer did not support the Adobe SVG Viewer. As a result 5 of 8 Windows Vista users did not complete the survey since they could not see the body graphics.

One participant used Linux and Firefox 3. The SVG visualisation worked fine, but the Java Runtime Environment could not be installed because of lacking administrator privileges. Two participants using Mac OS X and Safari did not complete the survey, because the Java applet was not working after installation of the Java Runtime Environment.

7.1.2 User Interface Aspect

All but two of the participants found interaction with the 2D body graphics “easy” or “very easy”. The two participants who disagreed did not recognise that the graphics was interactive. All but one participant were satisfied with the heart and body model and had no difficulties understanding them. Most users were satisfied with the written messages in the text boxes, but 3 participants would have preferred a more “entertaining” text. Further suggestions were to replace/extend information in the visualisation with a quiz to make it more interesting, to visualise smoking and alcohol consumption more explicitly, and to make navigation options visually more obvious. Participants who could see the visualisations perceived the application as moderately informative, and those without problems using it enjoyed the experience (Fischer 2009).

7.2 Clinical Study

In order to evaluate the effectiveness of the application a study was conducted with Māori heart failure patients. Heart failure is a major health issue for Māori (Ministry of Health, 2010; Riddell, 2005; Robson & Harris, 2007) since the heart failure mortality rate is approximately three times that of non-Māori (Māori Health (2010)). In addition, Māori are five and a half times more likely to be hospitalised with heart failure than non-Māori (Māori Health, 2010).

However, Māori have been resolute in resolving their state of health, incorporated key legislation, and developed models of health that resonate with their cultural values. The negative state of Māori health may be offset by the majority of health conditions (chronic illnesses and risky lifestyles) that are preventable and manageable. As such, this allows an opportunity for patient education intervention that acknowledges Māori cultural values and concepts, which must be taken into account when examining the effects of our application.

7.2.1 Study Design

The study was conducted using semi-structured interviews. Participants were Māori individuals diagnosed with heart failure who attended the Manukau Super Clinic, Manukau or whānau of the Māori individuals with heart failure. Exclusion criteria to participation were cognitive impairment, age of less than 18 years and a non-Māori ethnic identity. Te Reo Māori (the Māori language) was spoken as appropriate with the participants. Eighteen potential participants who met the exclusion and inclusion criteria were invited to participate in the study. Four potential participants declined due to failing health or other commitments. Eight Māori individuals with heart failure and six of their whānau members of the Māori individuals with heart failure participated in the study.

The use of semi-structured interviews has been suggested as the appropriate technique for health research among different cultural groups (Bowling 1997). An advantage of this approach is that more complex issues can be probed and answers clarified in a more relaxed environment. It is therefore more successful in enhancing in-depth discussion. Face to face interaction also supports ‘he kanohi kitea’ people meeting personally, so that trust in the relationship between the researcher and participants can be built.

Since many participants did not have a computer and/or Internet access the interviewer conducted the studies with a laptop containing an installation of the HEART MENTOR application (see figure 6).

A brief introduction informed the participant that the HEART MENTOR Programme was a ten minute programme with images of the heart and body, and information about heart failure. The programme was accessed as directed by the text and audio media to the completion of the programme. Information details requested by the programme were fictional or supplied information from either the interviewer or participant. The option was offered to the participant to operate the programme, if not the interviewer would take direction from the participant, for example, to view information again or to make changes to informational details. Participants then viewed the heart imagery programme. During the presentation of the programme, dialogue was encouraged from the participant by natural verbal prompts from the interviewer, such as, “let’s go back to that, shall we?” or “shall we put that in?”. After the completion of the programme, questions explored the participant’s responses to the imagery programme. At the end of the session all participants received a koha (gift) in appreciation of their participation.

Ethics approval was obtained from the Northern Y Regional Ethics Committee for a period of 1 year, from 31 July 2009 until 31 July 2010 (Ref: NTY/09/05/045). Further approval was from the Māori Research Review Committee Counties Manukau District Health Board (Ref: July_ap_03) and Counties Manukau District Health Board (Ref: 750) as the study was situated within the Counties Manukau District Health Board authority.

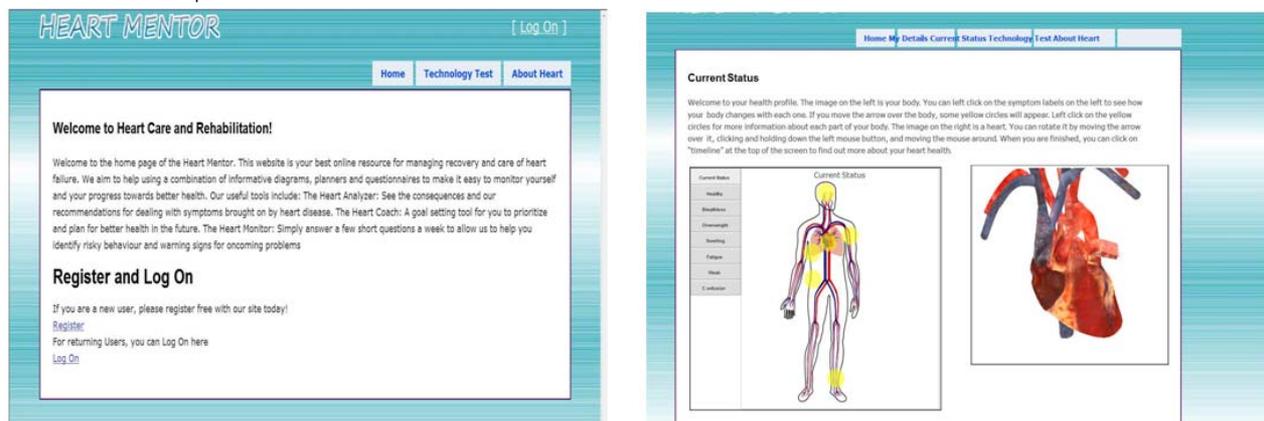


Figure 6: The HEART MENTOR “Welcome” page (left) and the interactive “Current Status” page (right).

7.2.2 Heart Imagery Programme Questions

The key themes to the heart imagery programme questions were, whether and how the programme added to knowledge about the heart condition and enhanced coherence in heart failure representations. These questions addressed two areas: The quality of the programme itself and understanding of the role of adherence to treatments in managing heart failure. Questions about the programme centred on whether information increased understanding of the heart condition, associated symptoms and its perceived benefits. One question about the programme was technical, in that the participant was asked about the ease and simplicity in operating the programme. Changes in understanding about treatment and symptom management were assessed by the adherence questions. An additional question, exploring possible gains in knowledge and understanding of the participants’ heart condition, probed for willingness to recommend the heart imagery programme to others, “Do you think this programme should be easily available to yourself and your whānau?”. More detailed information is found in (Morunga 2010).

7.2.3 Evaluation of Participant Responses

The HEART MENTOR programme depicts graphic images of the body and the heart. For that reason, prior to the demonstration of the heart imagery programme, the interviewer verified with the Māori participants that this was acceptable to them. As all participants agreed the demonstration of the heart imagery programme proceeded. Thirteen of the participants viewed and commented on the heart imagery programme.

Gaining *matauranga* (knowledge) and understanding information from the heart imagery programme was of great benefit to the participants. Most participants relished the opportunity to read and discuss sections of the programme.

Of the HEART MENTOR programme featured pages, the “Current Status” page in figure 6 evoked the most comments due to the graphic sections on ‘symptoms’ and ‘image of the heart’. The program text “The image on the left is your body”, was quickly responded to by the participants as in, “is that really my body?”. The interviewer explained that recording in the “My Symptoms” section tick boxes of the “My Details” page would provide the information for the image of the body. Some participants asked that the programme go back to the “My Symptoms” section to tick other boxes to verify the body image changes.

All the participants found the beating heart and the ability to rotate it very engaging. The inter-

viewer was continuously instructed to “move the heart around . . . that way . . . the other way”. Most questions were about “is that my heart beating?”, “why is it red and blue?”, and “is it (the heart) really that size?” The interviewer answered some of the questions and again advised the participants to talk with their doctors. One participant stated she was “fascinated by the beating and can’t take my eyes off the heart”. The images of the heart, body, circles and labels, all the information, had helped them to understand the symptoms of their heart condition better, or that they now recognised the symptoms of their whānau members. One individual with heart failure had gained some new understanding about blood flow in relation to the body’s ability to function.

“those parts in yellow. That was interesting, that. I don’t know if I can explain it, yeah, all I can say was it was interesting to see why certain parts of the body doesn’t function properly because of the blood flow, not getting much blood flow into it. That’s about all I picked up on that, the reason why they can become dysfunctional.”

However, another individual with heart failure was distressed by the symptom information but expressed the intention to be more vigilant in the future to his symptoms.

“That’s scary eh? Yeah, I know what to look for, you know the weakness, all the weakness, like the breathing, and your body blowing up, coming up to a heart attack.”

With added knowledge and understanding there can be improved management of the individual with heart failure’s illness. The following responses typify comments from the participants.

A whānau participant expresses her relief at gaining further understanding about the individual with heart failure’s illness to manage a specific area of concern.

“We truly know how to, what is happening inside his heart, and why he’s getting all these symptoms. In the 2 years that we’ve been dealing with this illness, it’s so good to have it summarised up so that we know how to care for ourselves better. You know the next thing I think we need to do is have a strict heart diet.”

Whānau members were pleased about identifying the individual with heart failure’s symptoms and understanding more about the illness.

“Yeah, yes, I understand it a little bit more. Because he used to be breathless too and he used to tell me that his doctor said that he has an enlarged heart.”

“Yeah I think, like I don’t think, you know having seen it, it makes me understand his illness better.... His puku, his swelling and stuff like that, that’s made that a bit clearer. And we identified with all those symptoms eh?”

As the following whānau participant implies the information added to their existing knowledge and, also, corrected a misconception.

"I mean we know that she's got a heart condition, but we don't know what it affects and what it entails. But I think to see why she actually has to do that and why the body's doing that, yeah, that's interesting. Because I didn't know that involved too much of the heart. I thought that was because of the smoking over all these years. So, yeah, I didn't realise it was the heart condition, why the lungs were filling. We knew that the fluid on the lungs wasn't good for her heart though. We understood that. Yeah, so that's good."

An individual with heart failure was enthusiastic about his improved understanding and possible strategies to manage the illness.

"it's new information and I just got to watch my eating, watch my drinking, listen to my body and, yeah, learn to eat more healthier, healthier food."

An unexpected benefit can be recognition of symptoms that were undiagnosed. One whānau participant recognised a symptom that was troubling her. To clarify, following is selected text from the heart imagery programme that preceded the following comment, "You may notice some swelling in your abdomen, legs and feet. You will see that these areas of the body are now bigger. Swelling occurs, because blood doesn't flow as quickly around your body, so fluid can build up in tissues and veins."

"Now I understand why my legs always swelled up. But I always had that for, even though I was skinny, you know, I still get swelling in the ankles. Well not now, I'm not skinny anymore. But I thought it was just the hereditary from my dad."

The complete interview findings are presented in (Morunga 2010). In summary the heart imagery computer-based programme is placed as the mechanism through which knowledge and understanding of the illness and its symptoms are enhanced. This can alleviate emotional distress that the participant may be experiencing. For example, one individual with heart failure was anxious as to why he had to take a specific medicine and undertook behaviours to avoid taking this medicine. Knowledge and understanding about the illness and more specifically, the medicine's purpose to treat a symptom can alleviate the distress and influence self-management of the illness and the symptoms.

The HEART MENTOR's graphics encouraged a greater sense of coherence between the symptoms and the condition, and thus enabled a strengthening of knowledge and understanding of the illness and the symptoms. This encouraged the endorsement by the participants to make available and accessible the programme to whānau and other individuals with heart failure so as to increase their knowledge and understanding of the illness and its symptoms. Thus, enhanced knowledge and understanding of the illness and its symptoms can motivate protective action, such as, for the individual with heart failure to improve self-management of the illness and the symptoms, and for the whānau participant to be supportive of the individual with heart failure's self-management of the illness and the associated symptoms.

8 Conclusion and Future Work

The HEART MENTOR is a novel application supporting the heart rehabilitation process through behavioural change. This is achieved by combining educational content with an interactive 2D body and 3D heart model reflecting patient parameters.

The usability study proved that the lack of web graphic standards and the rapid changes in software

and hardware made platform independence difficult to achieve. In particular SVG proved to be an unfortunate design decision due to Adobe support ceasing since Windows Vista. We are currently testing Adobe Flash and Silverlight for 2D graphics and Web3D and Silverlight for 3D graphics.

Several users did not notice that the 3D heart model is interactive and only started to rotate it after they were prompted to do so. We hence believe that more audio support is necessary to guide users through the application and to demonstrate disease and patient parameters (e.g., a dry cough versus a "normal" cough).

The heart imagery programme promotes knowledge and understanding of the illness and its associated symptoms. The sub-themes were (1) Emotional distress is alleviated by knowledge and understanding of the illness and its symptoms; (2) Text contents and imagery contents of graphics encourage knowledge and understanding of the illness and its symptoms and (3) Availability and accessibility of the heart imagery programme fosters shared knowledge and understanding of the illness and its symptoms. Overall, the study revealed that the patient education programme enhanced representational coherence of heart failure and its management held by Māori individuals with heart failure and their whānau, therefore, motivating engagement in protective behaviours.

We are currently extending the HEART MENTOR application to further empower patients to take charge of their own health. This will be achieved by adding social networking components for providing social support and opportunities for sharing with peers (patient groups). We also are developing interactive tools for creating *action plans* which allow patients to define and monitor goals.

Results so far have been encouraging and we strive to make our application accessible to a wider population group in the future.

9 Acknowledgements

We would like to thank staff from the National Heart Foundation, Māori health and cardiology health professionals of Counties Manukau District Health Board, and the Māori Research Review Committee (CMDHB) for their help in preparing the patient study. We are grateful to SIMTICS Limited for providing us with a polygonal 3D heart model. We would also like to thank Andreas von Arb for assisting us with technical support.

References

- American College of Cardiology/American Heart Association Task Force on Practice Guidelines (2005), 'ACC/AHA 2005 Guideline Update for the Diagnosis and Management of Chronic Heart Failure in the Adult'. URL: <http://circ.ahajournals.org/cgi/content/full/112/12/e154>.
- American Heart Association Inc. (2010a), 'Heart360™'. URL: <https://www.heart360.org>.
- American Heart Association Inc. (2010b), 'Homepage'. URL: <http://www.americanheart.org>.
- Andreas Neumann and Andre M. Winter (2010), 'SVG Window Object'. URL: <http://www.carto.net/papers/svg/gui/Window>.
- Andrew Davison (2010), 'Pro Java 6 3D Game Development'. URL: <http://fivedots.coe.psu.ac.th/~ad/jg2/>.

- Bowling, A., ed. (1997), *Research Methods in Health: Investigating Health and Health Services*, Open University Press.
- Bradley, M. M. & Lang, P. J. (1999), 'Fearfulness and affective evaluations of pictures', *Motivation and Emotion* **23**(1), 1–13.
- Bradlyn, A. S., Beale, I. L. & Kato, P. M. (2003), 'Psychoeducational interventions with pediatric cancer patients: Part I. patient information and knowledge', *Journal of Child and Family Studies* **12**, 257–277.
- Broadbent, E., Petrie, K. J., Ellis, C. J., Ying, J. & Gamble, G. (2004), 'A picture of healthmyocardial infarction patients' drawings of their hearts and subsequent disability: A longitudinal study', *Journal of Psychosomatic Research* **57**, 583–587.
- Cameron, L. D. & Chan, C. K. Y. (2008), 'Designing health communications: Harnessing the power of affect, imagery, and self-regulation', *Social and Personality Psychology Compass* **2**, 262–283.
- Cameron, L. D. & Leventhal, H., eds (2003), *The Self-Regulation of Health and Illness Behaviour*, Routledge.
- Doughty, R. N. & White, H. D. (2008), Epidemiology of heart failure, in R. R. Baliga, B. Pitt & M. M. Givertz, eds, 'Management of Heart Failure', Springer, London, pp. 1–11.
- Fischer, S. (2009), Web-based simulation for heart failure management and care, Master's thesis, University of Koblenz-Landau, Germany.
- Fuster, V., O'Rourke, R. A., Walsh, R. A. & Poole-Wilson, P., eds (2008), *Hurst's The Heart*, 12th edn, McGraw-Hill Companies, Inc.
- Gardner, M. P. & Houston, M. J. (1986), 'The effects of verbal and visual components of retail communications', *Journal of Retailing* **62**(1), 64–78.
- Hagger, M. S. & Orbell, S. (2003), 'A meta-analytical review of the common-sense model of illness representations', *Psychology & Health* **18**(2), 141–184.
- Hammond, D., Fong, G. T., McDonald, P. W., Cameron, R. & Brown, K. S. (2003), 'Impact of the graphic canadian warning labels on adult smoking behaviour', *Tobacco Control* **12**(4), 391–395.
- Hay, D. R. (2004), Cardiovascular disease in New Zealand, NHF report no. 82, National Heart Foundation of New Zealand. URL: <http://www.heartfoundation.org.nz/files/NHF6949%20TechReport82.pdf>.
- Holmberg, N. (2006), A framework for interactive web-based visualization, Master's thesis, Dept. of Computer Science, University of Auckland.
- Holmberg, N., Wünsche, B. C. & Tempero, E. (2006), A framework for interactive web-based visualization, in 'Proc. of Seventh Australasian User Interface Conference (AUIC 2006)', pp. 137–144.
- Horowitz, C. R., Rein, S. B. & Leventhal, H. (2004), 'A story of maladies, misconceptions and mishaps: effective management of heart failure', *Social Science & Medicine* **58**(3), 631–643.
- Hunter, P. J. & Pullan, A. J. (2002), 'New developments in the Auckland heart model', *International Journal of Bioelectromagnetism - Special Issue on Electrocardiology and Neurophysiology* **4**(2), 47–50.
- Inkscape (2010), 'Homepage'. URL: <http://www.inkscape.org/>.
- Kolko, J., ed. (2010), *Pause & Effect: The Art of Interactive Narrative*, Morgan Kaufmann.
- Lee, C. S., Tkacs, N. C. & Riegel, B. (2009), 'The influence of heart failure self-care on health outcomes: hypothetical cardioprotective mechanisms', *Journal of Cardiovascular Nursing* **24**(3), 179–187.
- Lee, T., Cameron, L., Wünsche, B. & Stevens, C. (2010), 'A randomized trial of computer-based communications using imagery and text information to alter representations of heart disease risk and motivate protective behaviour', *British Journal of Health Psychology*.
- Lopez, A. D., Mathers, C. D., Ezzati, M., Jamison, D. T. & Murray, C. J. L., eds (2006), *Global Burden of Disease and Risk Factors*, Oxford Press and The World Bank. URL: <http://files.dcp2.org/pdf/GBD/GBD.pdf>.
- Masoudi, F. A., Havranek, E. P. & Krumholz, H. M. (2002), 'The burden of chronic congestive heart failure in older persons: magnitude and implications for policy and research', *Heart Failure Reviews* **7**, 9–16.
- Meadows, M. S., ed. (2006), *Pause & Effect: The Art of Interactive Narrative*, Indiana, USA, New Riders.
- Morunga, E. R. (2010), Māori, heart failure and a computer-based heart imagery programme, Master's thesis, University of Auckland, New Zealand.
- Mosterd, A., Hoes, A. W., Bruyne, M. C., Deckers, J. W., Linker, D. T., Hofman, A. et al. (1999), 'Prevalence of heart failure and left ventricular dysfunction in the general population - the Rotterdam study', *European Heart Journal* **20**, 447–455.
- National Heart Foundation of New Zealand (2010), 'Homepage'. URL: <http://www.heartfoundation.org.nz>.
- Riegel, B., Moser, D. K., Anker, S. D., Appel, L. J., Dunbar, S. B., Grady, K. L. et al. (2009), 'State of the science: Promoting self-care in persons with heart failure: A scientific statement from the american heart association', *Circulation - Journal of the American Heart Association* **120**, 1141–1163.
- Rubin, J., Wünsche, B. C., Cameron, L. & Stevens, C. (2005), Animation and modelling of cardiac performance for patient monitoring, in 'Proceedings of IVCNZ '05', pp. 476–481.
- Simpson, A. & Barker, P. (March 2007), 'The persistence of memory: using narrative picturing to co-operatively explore life stories in qualitative inquiry', *Nursing Inquiry* **14**, 35–41.
- Strothotte, T. & Schlechtweg, S. (2002), *Non-Photorealistic Computer Graphics: Modeling, Rendering and Animation*, Morgan Kaufmann.
- Timmis, A. & McCormick, T., eds (2003), *Heart Failure*, Churchill Livingstone.
- W3C (2010), 'SMIL Animation - W3C Recommendation 04-September-2001'. URL: <http://www.w3.org/TR/smil-animation/>.
- Wünsche, B. C. & Young, A. A. (2003), 'The visualization and measurement of left ventricular deformation using finite element models', *Journal of Visual Languages and Computing - Special Issue on Biomedical Visualization for Bioinformatics* **14**(4), 299–326.

Visualizing the Refactoring of Classes via Clustering

Keith Cassell

Craig Anslow

Lindsay Groves

Peter Andreae

School of Engineering and Computer Science
 Victoria University of Wellington,
 PO Box 600, Wellington 6140, New Zealand
 Email: {kcassell,craig,lindsay,pondy}@ecs.vuw.ac.nz

Abstract

When developing object-oriented classes, it is difficult to determine how to best reallocate the members of large, complex classes to create smaller, more cohesive ones. Clustering techniques can provide guidance on how to solve this allocation problem; however, inappropriate use of clustering can result in a class structure that is less maintainable than the original. The ExtC Visualizer helps the programmer understand the class structure by visually emphasizing important features of the class's members and their inter-relationships. More importantly, it helps users see how various clustering algorithms group the class's members. These insights help a programmer choose appropriate techniques for refactoring large classes.

Keywords: Software visualization, clustering, refactoring, graph, maintainability

1 Introduction

Code maintenance is expensive. Some studies (Yip & Lam 1994) indicate that over 65% of the cost of software is maintenance. We address a common maintenance problem in object-oriented systems - the presence of large, complex classes with many methods and attributes (a.k.a. *members*). This paper describes our research in visualizing how the members of large classes can be re-organized using clustering techniques. Using the outputs of the clustering process, programmers can refactor their large classes and improve their software.

Fowler (Fowler et al. 1999) defines *refactoring* as “a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior”, and he identifies a large class as being one of the “bad smells” in software that indicate likely problems. Fowler recommends using the *Extract Class* refactoring to distribute the methods and attributes from the large class into appropriate new classes.

Our research is primarily concerned with determining how methods and attributes can be reallocated, so that class-oriented refactorings like *Extract Class* can be applied. As part of this, we want to help the programmer see the important characteristics of these class members and their inter-relationships, and

based on these, how they can be recombined to form more smaller, more cohesive classes.

Many tools, including our ExtC¹ tool (*Extract Class*), help programmers see potentially important characteristics of an object-oriented class through the use of color, shape, and size, as well as through the relationships between the members as depicted in graphs. While such displays are helpful, large classes tend to produce crowded displays that obscure the underlying structure. Moreover, some intraclass relationships are complex and involve many methods and attributes. What is needed is a display that emphasizes the most important relationships within the class, including complex ones.

Clustering algorithms can help deal with the complexity. Clustering algorithms provide the ability for grouping things based on their characteristics, so we can use them to discover the underlying class structure that is critical for refactoring. However, there are many potentially useful clustering algorithms, and these can produce widely varying results depending on the algorithm chosen, how the algorithms are parameterized, and on the characteristics of the underlying data upon which the algorithms operate.

Some programmers may be content to see the outcomes of the clusterings and then use these as the bases of new classes via an *Extract Class* refactoring. Other programmers will prefer to see the clustering algorithm in action, either to see how closely the algorithm's functioning matches their intuition, or to see whether some intermediate results might suit them better as the basis for new classes.

Our ExtC tool can show particular incremental clustering algorithms in action (Figure 1). The user can control when these algorithms combine (or separate) members. This provides the ability to see the groups produced by the clustering algorithm. Moreover, because the user sees when members are combined (and to a limited extent, why they are combined), he can determine how much this matches his intuition.

The remainder of this paper is structured as follows. Section 2 provides some background. Section 3 covers the visualization features of the ExtC tool, while Section 4 discusses observations we made while using ExtC to analyze a number of open source software projects and the insights we gathered based on the visualizations. Section 5 discusses related work. The final section contains our conclusions and discusses potential future work.

Copyright ©2011, Australian Computer Society, Inc. This paper appeared at the 34th Australasian Computer Science Conference (ACSC 2011), Perth, Australia, January 2011. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 113, Mark Reynolds, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.

¹available from <http://code.google.com/p/ext-c/>

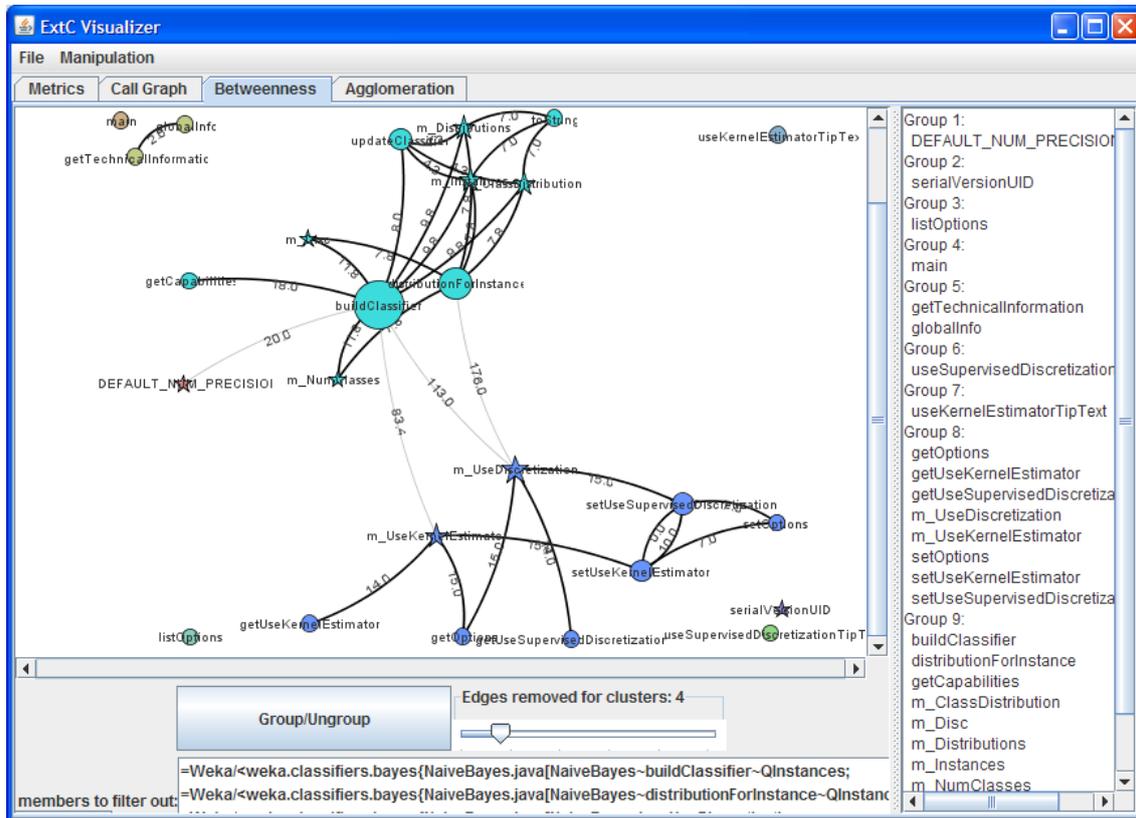


Figure 1: The ExtC GUI

2 Background

This section gives a brief background of some software metrics that are relevant to clustering. It then discusses clustering and how it is relevant to refactoring. We discuss related work in visualization in Section 5.

2.1 Metrics

Software metrics are used to measure various aspects of software. We are particularly interested in measuring cohesion (how well software elements fit together), which has proven useful for identifying problematic software (Lanza & Marinescu 2006, Simon et al. 2001, Wettel & Lanza 2008, Tsantalis & Chatzigeorgiou 2009, Cassell et al. 2009) and for evaluating a modified software system relative to the original one. Furthermore, because one of our goals is improved cohesion for the new classes, an understanding of how these metrics work can help us devise algorithms to improve cohesion.

There are many such metrics described in the literature (Briand et al. 1997, Chae et al. 2000, Zhou et al. 2002, 2004). While many just consider the number of interactions between methods and attributes, some also consider the pattern of method and attribute interactions within a class. For example, the Cohesion Based on Member Connectivity (CBMC) metric (Chae et al. 2000) represents the access patterns between methods and attributes as a graph, and calculates the cohesiveness based on the number of nodes that need to be removed to fragment the graph. Closely related to CBMC is ICBMC (Improved CBMC) (Zhou et al. 2002), which fragments the graph by removing edges rather than nodes. Metrics such as these offer insights into the underlying

structure of classes which can be exploited for determining how to extract new classes via clustering.

2.2 Clustering

Unsupervised clustering (Jain et al. 1999, Witten & Eibe 2001) is useful for identifying subsets of data that may represent coherent concepts. In the context of correcting poorly designed classes, the data to be clustered consists of the classes' members. The clustering algorithms use information about the affinity of the classes' members to group them. These groups can be used as the basis for forming cohesive classes, either in a large-scale class reorganization (Anquetil et al. 1999, Mitchell & Mancoridis 2006, Serban & Cibula 2008) or on a smaller scale, using the Extract Class refactoring (Cassell et al. 2009).

There are many different kinds of clustering techniques described in the literature. Berkhin (Berkhin 2002), for example, lists over 20 categories and sub-categories of clustering algorithms. Each algorithm has its own strengths, and because they have distinct ways of operating, different algorithms often produce different results with the same data set. (In fact, some algorithms are heuristic-based, so they may produce different results on different runs over the same data.) This section briefly reviews two categories of algorithms that we are investigating for use in refactoring - agglomerative and divisive clustering. Agglomerative clustering is a "bottom up" approach to clustering, while divisive clustering is "top down".

2.2.1 Agglomerative Clustering

Agglomerative clustering starts with seed entities and adds closely related entities to them until some stopping criterion is reached. The algorithms typically

determine what constitutes a closely related entity using a distance function (or similarity function). Entities that are closest (most similar) are combined, distances are recalculated, and the process repeats.

The effective use of agglomerative clustering depends on important choices regarding the parameters to these algorithms:

1. Feature set - the characteristics of the entities to be evaluated
2. Distance function - a function that measures the distance between the entities based on their feature set

Part of this parameterization involves the representation of the clusters, i.e., how one defines the feature set of the cluster and how the distance function takes those into account when computing the distance between groups or between groups and individual entities.

For the purpose of restructuring classes, the entities to be clustered are generally attributes and methods, which are themselves dissimilar. This raises the issue of how one calculates the distance between an attribute and a method or between groups that are combinations of attributes and methods.

Several researchers have used a Jaccard similarity metric (Simon et al. 2001, Serban & Czibula 2008) for this. A Jaccard similarity metric calculates similarity by dividing the number of features two entities have in common by the number of features total. As an example, one can assign slightly different feature sets to methods and attributes. The feature set for an attribute might include the attribute itself and the methods that access it, whereas the feature set for a method might include the method itself and the attributes it accesses. When a cluster is formed, its feature set becomes the merged features of its components.

We discuss how some other researchers have used this approach in Section 5, while Section 3.2.2 discusses our visualization of agglomerative clustering.

2.2.2 Divisive Clustering

Divisive clusterers work by splitting large groups into smaller ones. There are many divisive clustering techniques; this paper discusses *betweenness clustering*, which is a graph-based technique that has been applied to many domains (Girvan & Newman 2002), including object-oriented software (Dietrich et al. 2008, Cassell et al. 2009). One major difference between agglomerative and betweenness clustering is that the latter does not rely on similarity or distance functions that operate on the data.

Instead, betweenness clustering separates a connected graph into disconnected subgraphs by removing edges based on mathematical characteristics of the original graph. The subgraphs produced constitute the clusters. In a previous paper (Cassell et al. 2009), we discuss how we used betweenness clustering on the intraclass dependency graphs of some open source projects to recommend refactorings. We discuss betweenness clustering in the context of our visualizations in Section 3.2.2.

It is worth noting that betweenness clustering is similar in spirit to the ICBM technique (Zhou et al. 2002) for measuring cohesion, which relies on determining the cut sets for a graph. In fact, the creators of ICBM mention that it could be used as a basis for class restructuring.

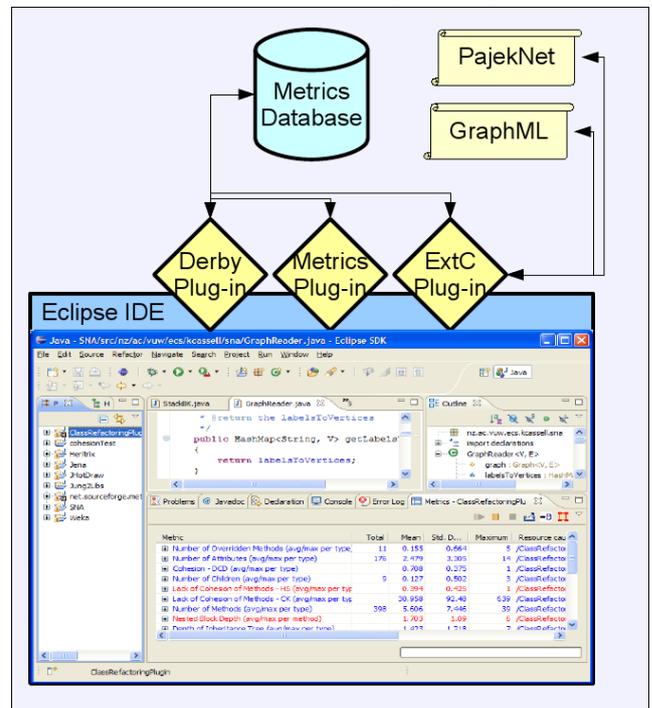


Figure 2: ExtC Architecture

3 ExtC Visualizer

3.1 Architecture

ExtC is designed to work as a plug-in in the Eclipse development environment (Shavor et al. 2003) as shown in Figure 2. In addition to providing an overall architecture via its plug-in oriented development framework, Eclipse provides extensive capabilities for code navigation and for programmatically processing Java code.

The Eclipse plug-in framework allows us to make use of third-party plug-ins. For example, we have enhanced the open-source *Eclipse Metrics2* plug-in (Sauer & Boissier 2010) to gather additional metrics and to store those metrics in a database. The Derby plug-in (Schaub 2008) provides ExtC access to that database.

ExtC uses the JUNG graph framework (O'Madadhain et al. 2003) for many graph-related tasks, including graph processing algorithms, layout, and manipulation. It further provides the capability of reading and writing graphs in either PajekNet or GraphML (Brandes & Pich 2005) format, so they can be read by other graphing packages.

Classes selected in the ExtC user interface have their intraclass dependency graphs shown in an ExtC graph display (Figure 3) while the corresponding code is loaded into Eclipse. In addition, the output of the clustering can be used as an input to an Extract Class refactoring tool, although this is currently a manual step.

3.2 Graphical User Interface

The ExtC GUI provides several views of the classes. The metrics view provides a tabular display of metric data pertaining to the classes of interest, while the dependency graph view (Figure 4) allows one to explore the relationships between a class's members. The agglomerative clustering view (Figure 5) and the betweenness clustering view (Figure 6) help the user

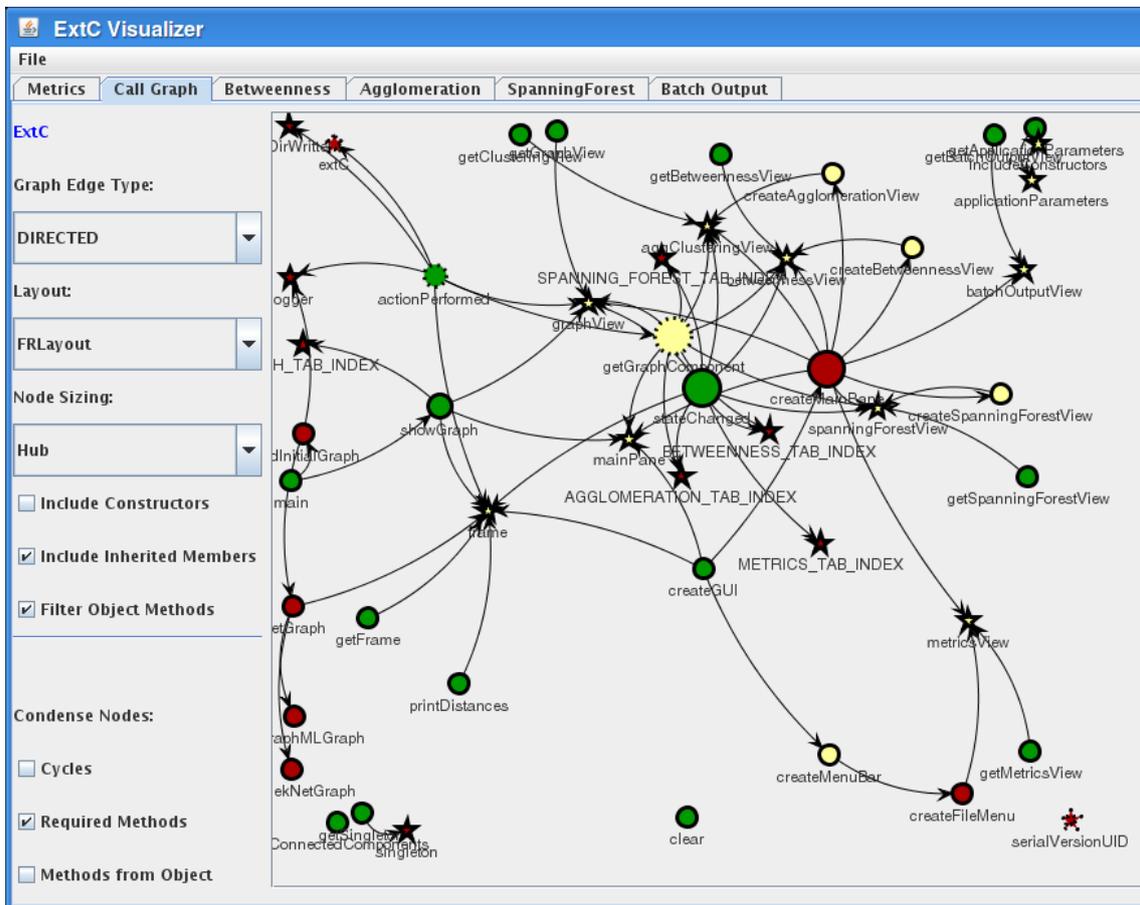


Figure 3: The ExtC Graph View

see how those clustering algorithms work on the underlying member data.

Its interactive graph display makes use of 2D graphics and color where the user can perform various manipulations (panning, scrolling, resizing, node movement, etc.). The clustering views have much the same capabilities, but also provide animations showing how the clustering algorithms work on the class's members. The animation capabilities will be discussed more thoroughly in Section 3.2.2.

3.2.1 Graph Display

The ExtC graph display (Figure 3) helps the user visualize the intraclass relationships between methods and attributes based on a static analysis of the code in a Java file. Each node represents either a method (circle), an attribute (star), or a group of members (triangle). Edges between nodes indicate either a method directly calling a method, or a method directly accessing an attribute. Colors indicate the allowable access to a member. For this figure, green indicates public, yellow indicates protected, and red indicates private; these colors are user configurable. The borders of the nodes indicate where the member is defined. A heavy border indicates that the member was defined in the class being examined, a lighter border indicates an inherited member, and a dashed border indicates a member defined in an inner class.

The user can alter the display of the graph as a whole by choosing any of several graph layout algorithms. Different layouts highlight different aspects of the graph structure. For example, Figure 4(a) shows how a shallow structure displayed within a directed

acyclic graph (DAG) layout makes it easy to identify “data classes” that provide access to attributes but that have little logic. After the initial layout, the user can move nodes using the mouse or choose another layout to redisplay.

Our tool provides an interface where the user can choose to size nodes by various criteria, for example, by their out degree. (This can be useful in helping to identify large “brain methods” (Lanza & Marinescu 2006) that contain too much functionality.) Current bases for sizing include in-degree, out-degree, and hub and authority scores (Kleinberg 1999).

In addition to the options on how to display the nodes representing the class's members, the user also has options regarding whether to display certain members. For example, users may decide that methods inherited from Java's *Object* class (*toString*, *equals*, etc.) are not relevant to their analysis of the behavior of the class and may choose to exclude them. This reduces “noise” in the graph and makes it easier to see the relationships between the class members that provide the main functionality.

ExtC also provides an interface where the user can choose to “condense” nodes that should be considered as a group. Currently, there are two supported condensations (1) nodes representing methods involved in recursive cycles, and (2) nodes representing methods required by interfaces or superclasses.

Interfaces and superclasses impose constraints on clustering, e.g. a single class must implement all of the methods specified in an interface. Consider Figure 4(b). This dependency graph looks like it could be split into two meaningful groups by eliminating three edges near the middle. However, if one condenses all

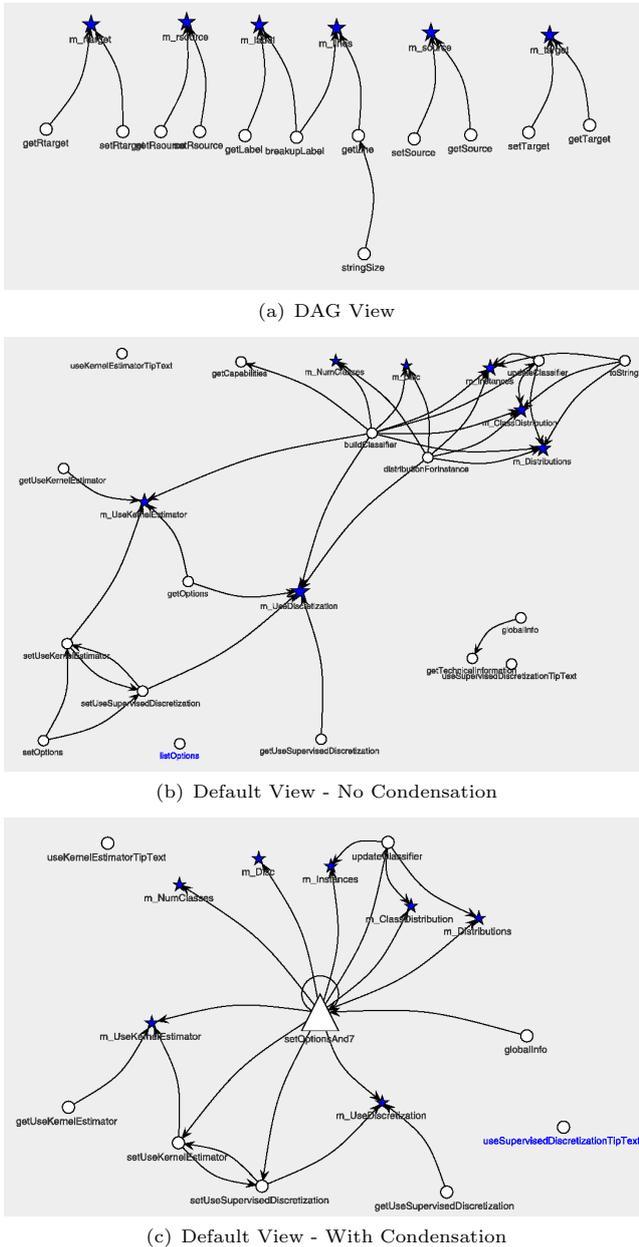


Figure 4: Dependency Graph Displays

methods imposed by interfaces and superclasses into a single triangular node, one gets the graph in Figure 4(c). This “hub and spoke” arrangement is less amenable to splitting via edge removal.

3.2.2 Clustering

The clustering views (Figures 5 and 6) have many of the same features as the graph view, including showing the dependency graph of a class. The primary difference between the graph view and the clustering views is the capability for animation. By manipulating a slider at the bottom of the screen, the user indicates the number of iterations of the clustering algorithm that he wants to execute. This may cause clusters of nodes to form on the screen. The exact graphical effect of moving the slider will depend on the clustering algorithm being used.

The visualizations for agglomerative and betweenness clustering are independent, but complementary.

Because the algorithms operate in a different manner and can produce different results, seeing the algorithms in action helps the programmer choose those clustering results that best match his intuition and use these as the basis for forming new classes.

Agglomerative Clustering

Agglomerative clustering starts with individuals and merges them together into groups. Many clustering systems (Jain et al. 1999) show the results of agglomeration as dendrograms, which are tree-based structures. Each level in the tree indicates the merger of existing clusters of one or more elements. ExtC includes such a tree-based display. It also provides a display that shows the agglomeration algorithm acting on the software dependency graph. The graph is customized for our software task in that it shows the underlying dependencies between the class’s members, but only shows the distances between linked nodes, rather than for all node pairs.

The display for the agglomerative clustering is similar to that of the graph view, with the following differences. Circular nodes represent unclustered members and are labeled with the member name. Clustered nodes are polygons. Clusters of only two members are represented as triangles while larger clusters are represented by polygons where the number of sides is equal to the number of members in the cluster. Cluster nodes are labeled with the names of one of their members followed by one or more additional special characters. The edges in the graph are labeled with the distances between the nodes.

The user controls the clustering via a slider. Moving the slider to the right causes more more clusters to form, and the distance values on the edges to change.

Figure 5 shows agglomerative clustering in action. For this example, we have created a distance function that produces a small distance for the nodes that have few links except to each other. At each iteration, the two nearest nodes are merged. The node that is farthest from the center is removed, while the more central one “absorbs” it and changes shape. After the merge step is completed, the edge weights (distances) are recalculated. While this is happening, a separate pane is simultaneously displaying a dendrogram-like tree that shows the hierarchical structure of the clusters formed thus far.

Figure 5(a) shows the dependency graph before the first aggregation step. The first seven iterations of aggregation are not shown. They are relatively uninteresting as the nodes on the outskirts of the graph are being clustered with their neighbors, and no cluster has more than three members. Figure 5(b) shows the graph after the seventh iteration. Figure 5(c) shows the graph after eight clustering iterations, when the first group of four is formed. Figure 5(d) shows the formation of a new group of two, and 5(e) shows the merger of that group with the group of four.

Betweenness Clustering

Betweenness clustering starts with a graph and removes edges to break the graph into disconnected parts, which are the clusters. The edges removed are those with the highest betweenness, where the betweenness value is the number of shortest paths between pairs of nodes that pass through that edge. If one considers a graph to represent information flow, where information passes through the edges, the high betweenness edges indicate where a graph can be cut to maximally disrupt information flow (or equiva-

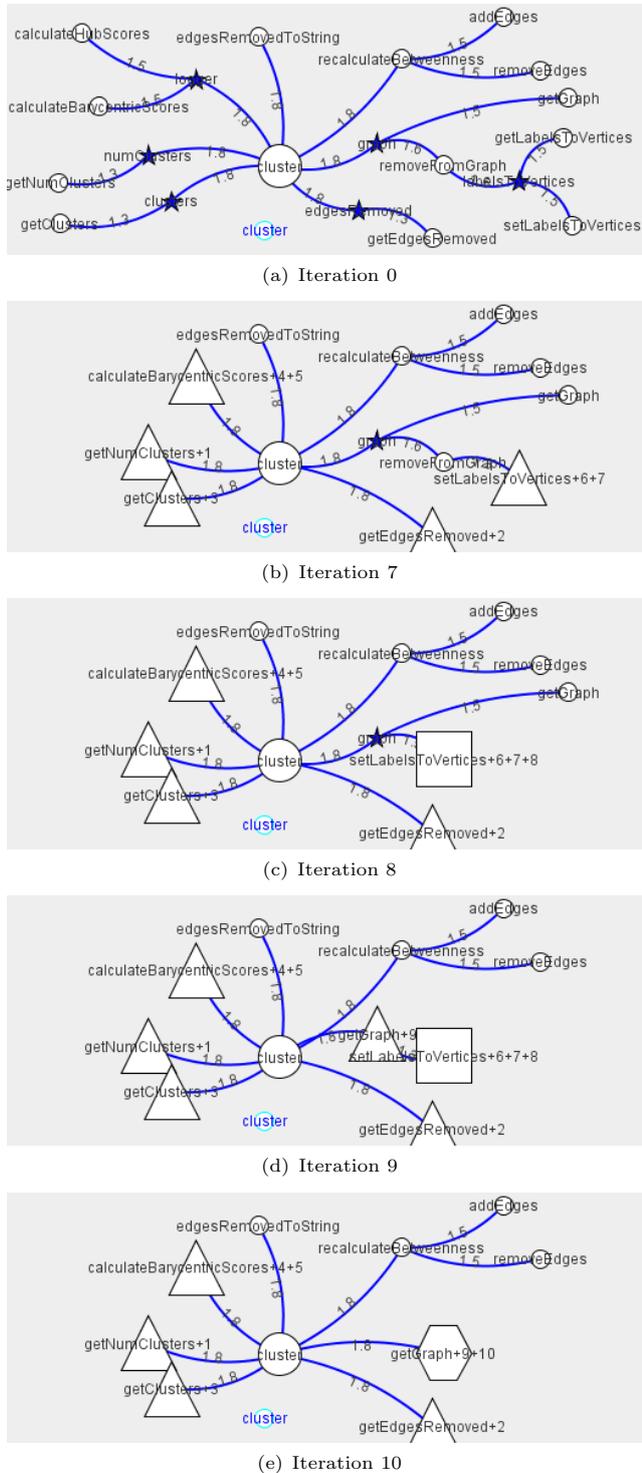


Figure 5: Agglomerative Clustering

lently, group together those nodes with highly shared information).

The display for the betweenness clustering is also similar to that of the graph view, and is based on a demo in JUNG (O'Madadhain et al. 2003). Circular nodes represent a class's members and are labeled with the member name. Clusters are represented by multiple nodes having the same color. Edges are labeled with their betweenness values.

The user controls edge removal via a slider. Moving the slider to the right causes more edges to be removed from the graph, more clusters to form, and

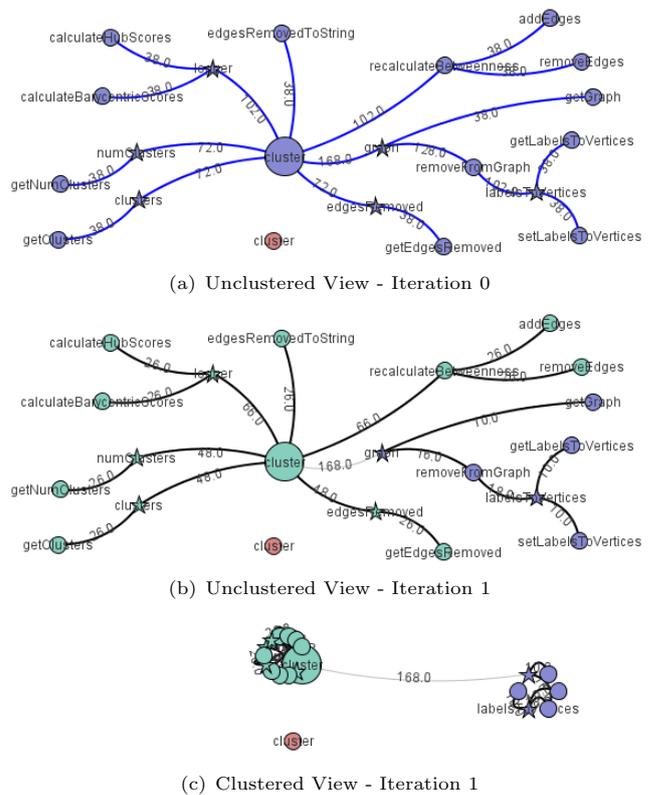


Figure 6: Betweenness Clustering

the betweenness values on the edges to change.

Figure 6(a) shows a dependency graph before the first edge has been removed by betweenness clustering for the same class as in the prior example. In this example, the removal of a single edge (Figure 6(b)) is sufficient to form a new cluster of five nodes towards the bottom right of the graph. Each cluster has a different color.

The basic graph display can be fairly cluttered for large classes. However, by toggling the grouping feature, the user can cause each cluster of nodes to be put close together (Figure 6(c)). Each cluster still has distinct colors, but now its members are arranged in a tight circular layout. This focuses the user's attention on the groups and their interactions.

The most visible links now convey a different meaning. The links between the nodes of a group are generally obscured due to the tight packing, so the most visible links are the grey links between clusters. These indicate edges that have been removed to separate the groups and indicate locations where the classes to be created will be coupled.

4 Discussion

We developed ExtC primarily to help us better understand the characteristics of large Java classes and how they might be refactored into smaller, more cohesive classes. The classes we examined came primarily from four open source projects collected within the Qualitas Corpus (Hertrix, Jena, JHotDraw, and Weka) (Qualitas Research Group 2008), as well as our own ExtC project. Over 100 classes with at least 20 methods were examined using ExtC.

This section discusses observations we made while using the visualizations to analyze these classes. Based on these observations, we reached conclusions

encompassing several broad areas:

1. Domain knowledge that should be considered by clustering algorithms to improve their ability to aid in refactoring large classes.
2. Insights into how the clustering algorithms worked
3. Possible improvements to the visualizations

The following subsections discuss these observations and conclusions, organized according to the view in which the observations were made. The subsection on the graph view incorporates most of the observations and conclusions about (1), while the subsections on agglomerative and betweenness clustering mostly cover (2) and (3). In general, for each observation we will name a single example of a class that showed the discussed behavior, although the behavior was typically shown by multiple classes.

4.1 Graph View

4.1.1 Cohesion and Clustering

Observation: In a DAG layout, many of the “leaves” of the graph were the expected star-shaped attributes; however, there were also fairly many circle-shaped methods. Upon investigating the corresponding code, some of these methods were shown to be “no-ops” or simple descriptors required by interfaces, while others made heavy use of other classes. Example: `weka.gui.visualize.SVGTextAreaFigure` (Weka)

Conclusion: Because most of the popular cohesion metrics concentrate on relationships between methods and attributes within a class, these leaf methods, and the associated calling methods, will cause misleadingly low cohesion scores. This may cause sub-optimal results when clustering using cohesion measurements as part of the distance function or stopping criteria.

4.1.2 Data Classes

Observation: Some large, noncohesive classes are largely “data classes” composed of attributes and their accessors. Example: `nz.ac.vuw.ecs.kcassell.callgraph.CallGraphNode` (ExtC)

Conclusion: Because we are primarily interested in splitting up a class’s logic, data classes should be not be considered. These should be detectable programmatically.

4.1.3 “Special” Members

Observation: Some of the nodes with the most links corresponded to methods that were not really part of the “business logic” of the class. These include nodes that represented “informational” methods (calls to loggers, `toString`, ...) and nodes that represented “generic” methods that consider most if not all of the fields (e.g. `clone`, `equals`, initialization methods), and probably cross-cutting concerns. Example: `net.n3.nanoxml.XMLElement` (JHotDraw)

Conclusion: To help clarify the fundamental logic of the class to the user, it is highly beneficial to remove nodes that are not part of the “business logic”, but are connected to many other nodes. Such nodes may

not have a large effect on the ultimate results, but they do decrease efficiency and introduce noise.

It would be nice to be able to automatically dispose of informational nodes, but this is difficult. For example, one could automatically eliminate `clone` from consideration. However, some of the code we have analyzed does most of its cloning work in a related method that `clone` calls. To adjust for the vagaries of highly connected nodes, we added a filter to the UI that enables the user to indicate nodes that should not be displayed.

Furthermore, many of these special members should not be moved exclusively to one of the split classes, but should themselves be split between the new classes. For example, each new class will likely want its own logging class, `toString` methods, etc.

4.1.4 Method Chains and Clustering Criteria

Observation: Many call graphs have “chains” of method calls, where each method in the chain calls only one other method from the class, terminating with a method accessing a single variable. Example: `com.hp.hpl.jena.graph.impl.GraphBase` (Jena)

Conclusion: If there is a member that is only connected to a single other member, those members should be clustered. An agglomerative clusterer’s distance function should capture this idea.

4.2 Agglomerative Clustering

Based on our tentative conclusion that chains of methods should be clustered, we decided to run some agglomerative clustering experiments where the distance function was primarily based on the number of edges on the shortest path between nodes, with a fractional secondary distance being added based on the number of edges each node had. (The secondary distance was to ensure that nodes with many connections would be joined after those that were more exclusively linked.)

4.2.1 “Fast-forwarding” Needed

Observation: Many of the initial groupings are somewhat obvious, as the nodes with only one connection are combined with their neighbors. For large classes, this makes the beginning of the animation fairly uninteresting. Example: `com.hp.hpl.jena.rdf.model.impl.StatementBase` (Jena)

Conclusion: We should provide a user option for “fast-forwarding” past specified kinds of agglomerations. It may be useful to provide options for fast forwarding, e.g. until a group of a specified size is reached.

4.2.2 Distance Functions

Observation: Squares (indicating three clustering steps) were appearing on the graph while there were still singly connected circles (indicating an unclustered member). This indicates that when there are multiple chains of nodes in the original graph, a single chain might be involved in multiple clusterings before another chain is involved in any. Example: `weka.gui.beans.CostBenefitAnalysis` (Weka)

Conclusion: A distance function that only looks at the current state of the graph and ignores the original state may give counterintuitive results.

4.2.3 Handling of Deprecated Methods

Observation: Some of the first nodes to be agglomerated involved deprecated methods.

Conclusion: Deprecated methods need to be a special case for class refactoring. Presumably, they were deprecated, because they could not be safely removed from the class. Visually, these should be grouped with the other condensed nodes. Example: `org.apache.commons.httpclient.HttpConnection` (Heritrix)

4.2.4 Variable Results

Observation: Running the same algorithm on the same data multiple times can give varying results. This occurs when multiple edges each have the same (smallest) distance. Example: `weka.classifiers.functions.RBFNetwork` (Weka)

Conclusion: Nondeterminism is troubling. This could be eliminated in many cases by having a more precise distance function. On the other hand, when two distances are the same, or nearly so, the user might prefer seeing the alternative clusterings. This warrants further study.

4.3 Betweenness Clustering

4.3.1 Graph Density

Observation: It is difficult to see the relative densities of different areas of graphs that represent classes with hundreds of members. Example: `com.hp.hpl.jena.rdf.model.impl ModelCom` (Jena)

Conclusion: Betweenness clustering helps by highlighting the less dense areas of a graph when the high betweenness edges are removed and the edge colors are changed.

4.3.2 Edge Weight Recalculation

Observation: When one removes the edge with the highest betweenness value, there can be a drastic shift in edge weights when they are recalculated. This occurs when the removal of the edge disconnects two subgraphs. The nodes that were connected by the high betweenness edge tended to be central to the pre-split graph. After the graph is disconnected, they tend to be peripheral to the new subgraphs. Example: `org.archive.crawler.datamodel.CrawlURI` (Heritrix)

Conclusion: Attempts to be efficient by cutting down on betweenness recalculation may give faulty results.

4.3.3 Node Weighting Based on Method Size

Observation: The betweenness clustering algorithm is sensitive to long chains of links. Consider the call graph shown in Figure 7(a). Suppose the method in the lower right is a large method that should be broken up into smaller, more maintainable pieces. After performing several Extract Method refactorings, one has a functionally equivalent class with the call graph in Figure 7(b). However, betweenness clustering now produces different groups.

Conclusion: This weakness might be addressed by node weighting, e.g. giving nodes representing large methods more weight than nodes representing small methods.

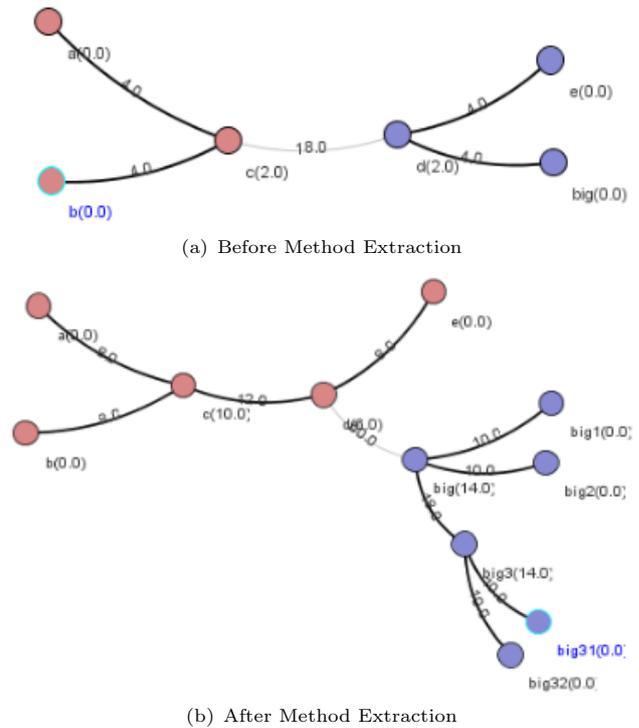


Figure 7: Betweenness Clustering and Method Extraction

4.3.4 “Short cut” Edges

Observation: Some high betweenness edges are just “short cuts” between some nodes that may also be linked indirectly, therefore betweenness clustering may remove edges that were not part of the cut set. Example: `org.jhotdraw.samples.svg.io.SVGOutputFormat` (JHotDraw)

Conclusion: We had specified several a priori conditions meant to ensure the quality of the proposed new class structure. One of the heuristic criteria specified for being an acceptable group was that the number of nodes in the new group should be at least four times the number of edges removed. This criterion was intended to limit the amount of coupling introduced by extracting a new class. Unfortunately, it can result in false negatives when edges are removed from the graph that are not part of the cut set, because these edges do not indicate coupling between the newly created classes.

Our original criteria should be modified as follows. Instead of a simple 4x multiplier based on edges removed, the density of the proposed clusters should be compared with that of the subgraph from which it came. Alternatively, only the edges removed that were part of the cut set should be considered in the 4x calculation.

4.3.5 Betweenness and Directed Graphs

Observation: When the graph is directed, the highly weighted edges tend to indicate those method calls that have the largest call tree. Removing edges just serves to isolate the busiest methods. Example: `com.hp.hpl.jena.n3.N3JenaWriterCommon` (Jena)

Conclusion: Betweenness clustering on directed call

graphs is not helpful for determining how to extract classes.

5 Related Work

There has been significant work regarding the identification and visualization of object-oriented software that needs refactoring (Lanza & Marinescu 2006, Wetzel & Lanza 2008), but relatively little work on visualizing how these problematic classes could be improved. Contributions have come from several areas. In addition to work in visualization, there have been contributions from the fields of refactoring, machine learning and clustering, graph theory, metrics, and network analysis.

The work on visualizing how object-oriented software can be improved by refactoring has shown a steady progression over the years. A Virtual Reality Modelling Language (VRML) visualization (Simon et al. 2001) can be used to present distance information between class members in three dimensions, so programmers might see opportunities to move methods or attributes between classes or to perform Extract Class or Inline Class refactorings. This visualization shows the proximity of the classes' members based on a Jaccard similarity metric, but does not explicitly show the relationships between them.

Churcher and his colleagues (Churcher et al. 2003) capture those relationships. Their 3D graphs of classes include relationship information as links. They point out how various graph shapes show different degrees of cohesion. They further note the relationship between graphs indicating low cohesion and the possibilities for splitting a class, but they do not provide further guidance about how that split might be accomplished.

Noack's thesis (Noack 2007) does describe how a split could be done. While his primary emphasis is on determining how to lay out call graphs to emphasize the structure of software using clustering techniques, he also provides several techniques for splitting the graph to separate dense areas of the graph (similar to betweenness clustering).

In an approach somewhat similar to ours, a group of researchers at Massey University (Dietrich et al. 2008) use a variation of betweenness clustering to help identify opportunities for reorganizing software modules. Based on a specification of how many edges are to be removed, their tool suggests groupings of software components.

None of the previously mentioned tools showed clustering in action; however, there has been some activity in the network analysis community. For example, the JUNG graph framework (O'Madadhain et al. 2003) provides a clustering demo graphically illustrating how betweenness clustering works on social network data, and we used this as the basis of our betweenness clustering visualization of object-oriented software.

Most of the work above has involved some kind of graph-based visualization; however, it is worth mentioning some clustering work for the purposes of refactoring classes that is not graph-based and does not have a significant visual component. Several researchers have applied clustering techniques to the problem of remodularizing software systems (Anquetil et al. 1999, Mitchell & Mancoridis 2006), but only recently have these techniques been applied to refactoring classes. A Romanian team (Serban & Czibula 2008) created a system to enable experimentation with various ways of recombining attributes and methods into classes. In most of their experiments

with agglomerative clustering, they used a Jaccard similarity metric. Unfortunately, it can be difficult to understand how these work without corresponding visualizations.

6 Conclusions

Our goal is to make object-oriented software easier to maintain by breaking large, noncohesive classes into smaller, more cohesive ones. Because programmers will make the final judgment about how classes will be organized, it is important that they see how a particular recommendation came about. ExtC helps by showing how clustering algorithms group (reallocate) methods and attributes for the formation of new classes.

We have provided visualizations for two major categories of clustering algorithms - agglomerative and divisive. Our experiences using ExtC thus far have inclined us to favor divisive clustering techniques, like betweenness clustering, over agglomerative techniques for two main reasons. First, divisive clustering matches up better with our mental model of the task of splitting classes. When refactoring, one wants to maintain the existing interface, so to be conservative, one generally wants to extract one new class at a time. This is consistent with making a single division of the graph. Secondly, divisive clustering generally requires fewer steps than agglomerative clustering for extracting classes. For a large class, watching the many steps required to reach two clusters can be extremely tedious.

ExtC has provided us some useful insights into our refactoring tasks, so we intend to enhance it. Our main effort over the next months is to expand the scope of ExtC beyond working on the dependency graph of a single class. Some of the information that can be useful for clustering involves how the studied class is used by other classes, and how it uses other classes. Our tool should display these relationships and distinguish them visually from the intraclass relationships.

We also want to expand our tool to be applicable for other class refactorings besides Extract Class. *Move Method*, *Extract Subclass*, and other refactorings have similar requirements to Extract Class in that they look for tight relationships (clusters) between certain members that are not well captured in the current class structure.

In addition to expanding the scope of what ExtC can do, we would like to make it more flexible. For example, it would be nice if a user could add some domain knowledge to help the clustering algorithms generate better results. Right now, the clustering algorithms work with little or no domain knowledge, i.e. the clustering doesn't know anything about software and how it should be structured. As software engineers we have clandestinely added some knowledge. For instance, the call graphs themselves represent certain relationships within the software. We have also added knowledge via our hard-coded distance functions; however, an arbitrarily complex user-defined (and domain aware) distance function could be used to give more precise results. The challenge is to determine how to let the user enter such knowledge for a distance function. Perhaps a software domain specific rule language can be constructed.

While there are enhancements to make, we feel that ExtC already provides capabilities that are useful to object-oriented programmers. There has been little prior work on applying clustering algorithms to refactoring large classes and, as far as we know, ExtC

is the first tool that suggests how to refactor software by showing clustering in action.

7 Acknowledgments

We thank James Noble for his helpful review comments and Stuart Marshall for his assistance.

References

- Anquetil, N., Fourrier, C. & Lethbridge, T. C. (1999), Experiments with clustering as a software modularization method, in 'Proceedings of the Sixth Working Conference on Reverse Engineering', IEEE Computer Society, p. 235.
- Berkhin, P. (2002), Survey of clustering data mining techniques, Technical report, Accrue Software.
- Brandes, U. & Pich, C. (2005), GraphML transformation, in 'Graph Drawing', pp. 89–99.
- Briand, L., Daly, J. & Wust, J. (1997), A unified framework for cohesion measurement in object-oriented systems, in 'Proceedings of the Fourth International Software Metrics Symposium, 1997.', pp. 43–53.
- Cassell, K., Andreae, P., Groves, L. & Noble, J. (2009), Towards automating class-splitting using betweenness clustering, in '24th IEEE/ACM International Conference on Automated Software Engineering', Auckland, New Zealand, pp. 595–599.
- Chae, H. S., Kwon, Y. R. & Bae, D.-H. (2000), 'A cohesion measure for object-oriented classes', *Software Practice and Experience* **30**(12), 1405–1431.
- Churcher, N., Irwin, W. & Kriz, R. (2003), Visualising class cohesion with virtual worlds, in 'Proceedings of the Asia-Pacific Symposium on Information Visualisation (APVIS)', Australian Computer Society, Inc, pp. 89–97.
- Dietrich, J., Yakovlev, V., McCartin, C., Jenson, G. & Duchrow, M. (2008), Cluster analysis of java dependency graphs, in 'Proceedings of the 4th ACM Symposium on Software Visualization', ACM, Ammersee, Germany, pp. 91–94.
- Fowler, M., Beck, K., Brant, J., Opdyke, W. & Roberts, D. (1999), *Refactoring: Improving the Design of Existing Code*, Addison-Wesley, Boston.
- Girvan, M. & Newman, M. (2002), 'Community structure in social and biological networks.', *Proc Natl Acad Sci U S A* **99**(12), 7826, 7821.
- Jain, A., Murty, M. & Flynn, P. (1999), 'Data clustering: a review', *ACM Computing Surveys* **31**(3), 264–323.
- Kleinberg, J. M. (1999), 'Authoritative sources in a hyperlinked environment', *J. ACM* **46**(5), 604–632.
- Lanza, M. & Marinescu, R. (2006), *Object-Oriented Metrics in Practice*, Springer-Verlag New York, Inc.
- Mitchell, B. & Mancoridis, S. (2006), 'On the automatic modularization of software systems using the bunch tool', *IEEE Transactions on Software Engineering* **32**(3), 193–208.
- Noack, A. (2007), Unified quality measures for clusterings, layouts, and orderings of graphs, and their application as software design criteria, PhD thesis, Brandenburg University of Technology, Cottbus, Germany.
- O'Madadhain, J., Fisher, D., White, S. & Boey, Y. (2003), The JUNG (Java universal Network/Graph) framework, Technical Report UCI-ICS 03-17, School of Information and Computer Science, University of California, Irvine.
- Qualitas Research Group (2008), 'Qualitas corpus version 20080208'.
URL: <http://www.cs.auckland.ac.nz/~ewan/corpus>
- Sauer, F. & Boissier, G. (2010), 'Eclipse metrics plugin continued', <http://metrics2.sourceforge.net/>.
- Schaub, S. (2008), 'Eclipse corner article: Creating database web applications with eclipse', <http://www.eclipse.org/articles/article.php?file=Article-EclipseDbWebapps/index.html>.
- Serban, G. & Czibula, I. (2008), Object-Oriented software systems restructuring through clustering, in 'Artificial Intelligence and Soft Computing - ICAISC 2008', Springer-Verlag, Berlin / Heidelberg, pp. 693–704.
- Shavor, S., D'Anjou, J., Fairbrother, S., Kehn, D., Kellerman, J. & McCarthy, P. (2003), *The Java(TM) Developer's Guide to Eclipse*, Addison-Wesley Professional.
- Simon, F., Steinbrckner, F. & Lewerentz, C. (2001), Metrics based refactoring, in 'Proceedings of the Fifth European Conference on Software Maintenance and Reengineering', IEEE Computer Society, p. 30.
- Tsantalis, N. & Chatzigeorgiou, A. (2009), 'Identification of move method refactoring opportunities', *IEEE Transactions on Software Engineering* **35**(3), 347–367.
- Wettel, R. & Lanza, M. (2008), Visually localizing design problems with disharmony maps, in 'Proceedings of the 4th ACM Symposium on Software Visualization', ACM, Ammersee, Germany, pp. 155–164.
- Witten, I. H. & Eibe, F. (2001), *Data Mining.*, Hanser Fachbuch.
- Yip, S. & Lam, T. (1994), A software maintenance survey, in 'Proceedings First Asia-Pacific Software Engineering Conference', pp. 70–79.
- Zhou, Y., Lu, J. & Xu, H. L. B. (2004), 'A comparative study of graph theory-based class cohesion measures', *SIGSOFT Softw. Eng. Notes* **29**(2), 13–13.
- Zhou, Y., Xu, B., Zhao, J. & Yang, H. (2002), ICBCMC: an improved cohesion measure for classes, in 'Proceedings of the International Conference on Software Maintenance (ICSM'02)', IEEE Computer Society, pp. 44–53.

Optimistic and Efficient Concurrency Control for Asynchronous Collaborative Systems

Haifeng Shen

Yongyao Yan

School of Computer Science, Engineering and Mathematics
 Flinders University, Adelaide, Australia
 Email: {haifeng.shen, yan0056}@flinders.edu.au

Abstract

Concurrency control is a key issue in distributed systems. A number of techniques have been devised to tackle the issue, but these techniques are generally unsuitable to be used in collaborative systems, which have the special requirements of consistency maintenance, responsiveness, and unconstrained interaction. OT (Operational Transformation) is an optimistic concurrency control technique originally invented for synchronous collaborative systems to meet these requirements. But existing transformation control algorithms are inefficient to be used in asynchronous systems. In this paper, we present an OT-based concurrency control solution for asynchronous collaborative systems, including an efficient contextualization-based transformation control algorithm underpinned by operation propagation and replaying protocols to achieve contextualization. The solution has been formally verified in terms of consistency maintenance and demonstrated by a variety of prototype collaborative applications.

Keywords: collaborative systems, concurrency control, consistency maintenance, operational transformation, contextualization.

1 Introduction

Interactive applications have been around for decades and are still among the most commonly-used ones in our work and daily lives. People use them to do various tasks such as coding, word processing, designing, modeling, and entertaining. Collaboration, which is naturally needed when multiple people are involved in the same or related tasks, remains cumbersome, tedious, and error-prone because most of these applications are primarily single-user-oriented.

Collaborative systems facilitate and coordinate collaboration among multiple users who are jointly fulfilling common tasks over computer networks, particularly the Internet. They can be brand new multi-user interactive applications built from scratch with collaboration in mind or augmented single-user interactive applications with additional collaborative functionality. Such a system typically consists of three characteristic components: multiple human users, interactive applications that provide the human-computer interfaces, and the high-latency Internet backbone that connects distributed computers.

Copyright ©2011, Australian Computer Society, Inc. This paper appeared at the 34th Australasian Computer Science Conference (ACSC 2011), Perth, Australia, January 2011. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 113, Mark Reynolds, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

Concurrency control is the key to managing the contention for distributed data resources and to ensure the orderly access to shared data. It is a major issue in most distributed systems and collaborative systems are of no exception. Over the past decades, a number of concurrency control techniques, such as floor control (Greenberg & Marwood 1994), locking (Knister & Prakash 1993), transactions (Bernstein et al. 1987), causal ordering (Raynal & Singhal 1996), and serialization (Karsenty & Beaudouin-Lafon 1993), have been devised for a variety of distributed systems. However, these techniques are generally unsuitable to be used in collaborative systems because these characteristic components have special concurrency control requirements (Ellis & Gibbs 1989, Sun et al. 1998).

First, human users, who interact with the interactive applications to manipulate the data sources via the human-computer interfaces, have the special requirements of fast response and unconstrained interaction, i.e., to interact with any data objects at any time freely and responsively. Second, as the communication latency of the Internet is high, collaborative systems usually achieve good responsiveness by means of replication to hide the latency - the shared data source is replicated across multiple collaborating sites and one only interacts with her/his own replica. However, replication creates the special requirement of consistency maintenance because inconsistencies may occur due to the concurrent activities from multiple users and the non-deterministic communication latency.

OT (Operational Transformation) (Ellis & Gibbs 1989, Sun & Ellis 1998) is an optimistic concurrency control technique originally invented for synchronous collaborative systems, where users collaborate at the same time by instantly propagating every operation generated at one site to others. It can achieve consistency maintenance without sacrificing good responsiveness and unconstrained interaction. OT consists of a set of transformation functions that transform one operation against another, and a transformation control algorithm that ensures the invocation of every transformation function satisfies required precondition. Over the past decade or so, quite a few transformation control algorithms have been devised for various synchronous collaborative systems.

For OT to be used for asynchronous collaborative systems, where users collaborate at different times by infrequently propagating batches of operations, an efficient transformation control algorithm is imperative because existing ones are catered for synchronous systems only and are inefficient to be used in asynchronous systems. In this paper, we present an OT-based concurrency control solution for asynchronous collaborative systems, including an efficient contextualization-based transformation control algorithm underpinned by operation propagation and re-

playing protocols to achieve contextualization. The solution has been formally verified in terms of consistency maintenance and demonstrated by a variety of prototype collaborative applications.

The rest of the paper is organized as follows. The next section introduces a consistency model and some background about OT. After that, the contextualization-based transformation control algorithm is presented and verified. The subsequent section presents the operation propagation and replaying protocols to achieve contextualization. Significance and applications of the work are discussed in the following section. Finally, the paper is concluded with a summary of major contributions and future work.

2 A Consistency Model and OT

The consistency model consisting of three consistency properties *Convergence*, *Causality preservation*, and *Intention preservation* (Sun et al. 1998), has been widely used to verify whether a concurrency control technique can maintain consistency in collaborative systems. The *convergence* property ensures the consistency of the final states of the shared data source at the end of a collaborative session. This property can be preserved by known techniques such as serialization. The *causality preservation* property ensures the consistency of the execution orders of causally related operations during a collaborative session. This property can be preserved by known techniques such as causal ordering. OT was particularly invented for *intention preservation* and extended for *Convergence* as well. Alternative solutions for *intention preservation* are exemplified by the *Mark and Retrace* technique (Gu et al. 2005).

Every interactive application provides a set of AOs (Application Operations) for a user or the API (Application Programming Interface) to manipulate data objects in the data source and these application-dependent AOs can be abstracted into the following three application-independent POs (Primitive Operations) (Sun et al. 2006):

- **Insert**(*addr*, *obj*) denotes an *Insert* operation that creates an object *obj* at address *addr* in the data source,
- **Delete**(*addr*) denotes a *Delete* operation that removes the object at address *obj* in the data source, and
- **Update**(*addr*, *key*, *val*) denotes an *Update* operation that changes the attribute *key*, to the value *val*, of the object at address *addr* in the data source.

To use OT for *intention preservation*, the *addr* parameter used to reference a PO's target object must follow a *data addressing model* that complies with *XOTDM* (eXtend Operational Transformation Data Model) consisting of a hierarchy of addressing groups, where each group consists of multiple independent linear addressing domains (Sun et al. 2006). The intention of an operation is the execution effect at the time when it is generated. To preserve an operation's intention, OT ensures the operation's target object is correctly addressed across all collaborating sites.

Consider a collaborative drawing session shown in Figure 1, where the data source consisting of three objects Ob_0 (triangle), Ob_1 (circle), and Ob_2 (square) is replicated at two sites. For the sake of simplicity (without loss of generality), we assume the *data addressing model* consists of a single linearly addressed domain *Drawings*, which initially contains $[Ob_0, Ob_1, Ob_2]$. At *site 1*, operation $Op_1 = \mathbf{Delete}([(Drawings, 0)])$

is executed to remove the object at address $[(Drawings, 0)]$, which is Ob_0 (triangle). After that, *Drawings* contains $[Ob_1, Ob_2]$. At *site 2*, operation $Op_2 = \mathbf{Update}([(Drawings, 1)], fill, b)$ is executed concurrently to fill the object at address $[(Drawings, 1)]$, which is Ob_1 (circle), with the *black* color. After that, *Drawings* remains $[Ob_0, Ob_1, Ob_2]$.

If Op_1 and Op_2 were propagated and replayed as-is for consistency maintenance (as depicted by the dashed lines), inconsistency would occur in the sense that the two replicas of the shared data source become divergent and Op_2 's intention (filling the circle with the *black* color) is violated at *site 1*. The intention violation problem was essentially caused by the inconsistent addressing of the same object Ob_1 (circle). At the time when Op_2 was generated at *site 2*, Ob_1 (circle) was addressed by $[(Drawings, 1)]$, which was also the case at *site 1* before the execution of Op_1 . However, after the execution of Op_1 at *site 1*, Ob_0 was removed, and Ob_1 took up Ob_0 's position (index 0) in *Drawings*. Consequently, Ob_1 should be addressed by the new address $[(Drawings, 0)]$. The wrong object Ob_2 (square) would be referenced if the old address $[(Drawings, 1)]$ were still used.

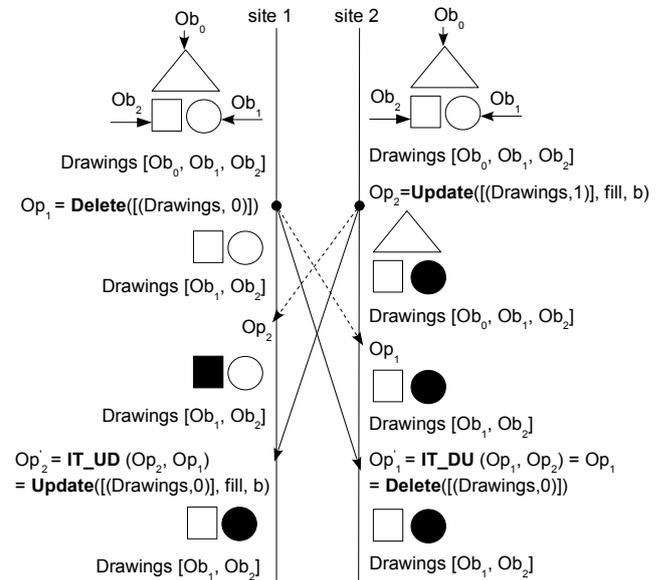


Figure 1: Intention preservation of Op_2 by OT

OT can solve the intention violation problem by compensating for the impact of concurrent operations on addressing an object's replicas. In this example, when Op_2 is propagated to *site 1* (as depicted by the solid line from *site 2* to *site 1*), it has to be transformed against the concurrent operation Op_1 to include its impact on addressing the referenced object Ob_1 by $\mathbf{IT_UD}(Op_2, Op_1)$: $Op'_2 = \mathbf{Update}([(Drawings, 0)], fill, b)$ ($\mathbf{IT_UD}(Op_a, Op_b)$ is a transformation function transforming an *Update* operation Op_a against a *Delete* operation Op_b). The replaying of Op'_2 at *site 1* fills the circle with the *black* color, which has preserved Op_2 's intention.

Transformation function $\mathbf{IT}(Op_a, Op_b)$: Op'_a (Inclusion Transformation) transforms operations Op_a against Op_b in such a way that the impact of Op_b is effectively included in the output operation Op'_a . Depending on Op_a 's and Op_b 's types, there are 9 (3×3) functions ($\mathbf{IT_UD}(Op_a, Op_b)$ is one of them). Details of these functions can be found in (Sun et al. 1998, 2006).

It is noticed that the divergence problem has also been solved by OT in the above example. In fact, transformation functions are defined to take care of both *intention preservation* and *convergence* (there

are conditions and properties to verify them, which will be discussed in later sections).

3 OT-based Concurrency Control in Asynchronous Collaborative Systems

To transform operation Op_a against operation Op_b by the transformation function $\mathbf{IT}(Op_a, Op_b): Op'_a, Op_a$ and Op_b must meet a pre-condition.

Definition 1 (Operation Context)

Given an operation Op , its context, denoted by Υ_{Op} , is the state of the data source based on which Op is defined.

Definition 2 (Operation Context Equivalent Relation)

Given two operations Op_a and Op_b , Op_a is context-equivalent to Op_b , denoted by $Op_a \sqcup Op_b$, iff (if and only if) $\Upsilon_{Op_a} = \Upsilon_{Op_b}$.

Definition 3 (Operation Context Preceding Relation)

Given two operations Op_a and Op_b , Op_a is context-preceding Op_b , denoted by $Op_a \mapsto Op_b$, iff $\Upsilon_{Op_b} = \Upsilon_{Op_a} \vdash Op_a$, where “ \vdash ” is the operation execution operator.

Υ_{Op_a} is the context on which Op_a is defined. After the execution of Op_a on Υ_{Op_a} , the new context can be described by $\Upsilon_{Op_a} \vdash Op_a$. If operation Op_b is defined on this new context, i.e., $\Upsilon_{Op_b} = \Upsilon_{Op_a} \vdash Op_a$, then Op_a is context-preceding Op_b . In general, if the initial context is Φ and n operations Op_1, \dots, Op_n have been executed in sequence, then the new context can be described by $\Phi \vdash Op_1 \vdash \dots \vdash Op_n$. For $\forall i \in \{2, \dots, n\}$, $Op_{i-1} \mapsto Op_i$ because $\Upsilon_{Op_1} = \Phi$ and $\Upsilon_{Op_i} = \Upsilon_{Op_{i-1}} \vdash Op_{i-1}$.

So the pre-condition of the transformation function $\mathbf{IT}(Op_a, Op_b): Op'_a$ is $Op_a \sqcup Op_b$. This pre-condition is to ensure their *addr* parameters are directly comparable since they are defined on the same context. A few TCAs (Transformation Control Algorithms) such as *GOT* (Sun et al. 1998), *GOTO* (Sun & Ellis 1998), *adOPTed* (Ressel et al. 1996), *Jupiter* (Nichols et al. 1995), and *COT* (Sun & Sun 2006), have been devised to ensure the pre-condition of every transformation function invocation is satisfied in synchronous collaborative systems. The transformation function $\mathbf{IT}(Op_a, Op_b): Op'_a$ also has a post-condition, which is $Op_b \mapsto Op'_a$ because Op'_a has included the impact of Op_b in such a way that the context of Op'_a is defined on the one right after the execution of Op_b . Every transformation function must be defined to ensure this post-condition.

In addition, the following *OCRCP* (Operation Context Relation Property) describes the correlation between “ \sqcup ” and “ \mapsto ” relations.

Property 1 Operation Context Relation Property (OCRCP)

Given three operations Op_a, Op_b , and Op_c , if $Op_a \mapsto Op_b$ and $Op_a \mapsto Op_c$, then $Op_b \sqcup Op_c$. **Proof:**

1. $\Upsilon_{Op_b} = \Upsilon_{Op_a} \vdash Op_a$ because $Op_a \mapsto Op_b$,
2. $\Upsilon_{Op_c} = \Upsilon_{Op_a} \vdash Op_a$ because $Op_a \mapsto Op_c$, and
3. $Op_b \sqcup Op_c$ because $\Upsilon_{Op_b} = \Upsilon_{Op_c}$.

3.1 Main Issues in Applying OT to Asynchronous Collaborative Systems

Existing TCAs are catered for synchronous collaborative systems in the sense that every operation generated at one site needs to be individually timestamped and instantly propagated to remote sites, and the replaying of every remote operation at any remote site needs to trigger a separate TCA invocation. However, in asynchronous systems, individual timestamping and instant propagation of every operation is either infeasible or unnecessary. On the one hand, it is infeasible in intermittent networking environments (e.g., wireless) because network connection is not persistently available. On the other hand, it is unnecessary because: 1) users are more sensitive to objects that are of their interests and therefore it makes better sense to batch coherent operations for propagation and replaying; 2) some micro-step operations never need to be propagated at all because they are either not interested by other users, or are overridden by or merged into later operations; 3) asynchronous collaboration does not need instant notification of each other’s micro-step progress; and 4) in the Internet or wireless networking environments, where the network resource is scarce, it is wasteful to propagate every operation separately and instantly.

More importantly, it is rather inefficient to invoke a separate TCA for replaying every remote operation in asynchronous systems. Most existing TCAs used in synchronous systems have a quadratic time complexity or worse. If n remote operations arrive at a site and every operation Op needs to invoke a $\mathbf{TCA}(Op, C): Op'$ (C is the collection of operations that have been executed at that site and are concurrent with Op) to derive its execution form, then the TCA will be invoked n times and the overall time complexity will be cubic or even worse! In synchronous systems, the quadratic execution time of TCA is acceptable as C is normally small (because each operation is instantly garbage-collected after having been transformed against all concurrent operations at every collaborating site (Sun et al. 1998)). In contrast, n and C both tend to be big in asynchronous systems due to the infrequent propagation of operations, which makes the cubic time complexity unacceptable.

To apply OT to distributed asynchronous collaborative systems, we need to devise an efficient TCA that does not rely on either timestamping or instant propagation of individual operations. In the following section, we present a new TCA that does not rely on timestamping or instant propagation of individual operations and can efficiently transform two contextualized lists of operations with a quadratic time complexity. Underpinned by an operation propagation protocol and an operation replaying protocol, the two contextualized lists can be of random length, making the algorithm viable for a wide spectrum of collaboration paradigms and particularly efficient for asynchronous collaboration, where the two lists tend to be long.

3.2 SLOT - A Contextualization-based TCA

Definition 4 (Contextualized List of Operations).

Given a list of operations L , L is said to be *contextualized*, iff $L[i] \mapsto L[i+1]$ ($0 \leq i < |L|-1$), where $L[i]$ is the $(i+1)^{th}$ operation in L and $|L|$ is the length of L .

Operations in a contextualized list may come from the same site or from different sites. Take a list consisting of two operations Op_a and Op_b as an example. First, if Op_a and Op_b are consecutively generated at the same site, then $L = [Op_a, Op_b]$ is naturally contextualized because $Op_a \mapsto Op_b$. Second, if Op_a and

Op_b are concurrently generated at two different sites, and $Op_a \sqcup Op_b$, then $L = [Op_a, Op'_b]$ or $[Op_b, Op'_a]$, where $\mathbf{IT}(Op_b, Op_a): Op'_b$ and $\mathbf{IT}(Op_a, Op_b): Op'_a$, is contextualized because $Op_b \mapsto Op'_a$ and $Op_a \mapsto Op'_b$ (by the post-conditions of the two \mathbf{IT} function invocations).

Definition 5 (*List Context*).

Given a *contextualized* list L , its context, denoted by Ψ_L , is the context of the first operation in L , i.e., $\Psi_L = \Upsilon_{L[0]}$.

Definition 6 (*List Context Equivalent Relation*).

Given two *contextualized* lists L_a and L_b , L_a is context-equivalent to L_b , denoted by $L_a \equiv L_b$, iff $\Psi_{L_a} = \Psi_{L_b}$.

Definition 7 (*List Context Preceding Relation*).

Given two *contextualized* lists L_a and L_b , L_a is context-preceding L_b , denoted by $L_a \dashrightarrow L_b$, iff $\Psi_{L_b} = \Psi_{L_a} \succ L_a$, where “ \succ ” is the list execution operator (operations in L_a are executed in sequence on context Ψ_{L_a}).

If $L_a \dashrightarrow L_b$, then $\Upsilon_{L_b[0]} = \Upsilon_{L_a[0]} \vdash L_a[0] \vdash \dots \vdash L_a[|L_a|-1]$ and $L_a[|L_a|-1] \mapsto L_b[0]$. Therefore, if L is the catenation of L_a and L_b , denoted by $L = L_a \succ L_b$, then L is also contextualized. Furthermore, the following two properties describe the correlations between “ \equiv ” and “ \dashrightarrow ” relations.

Property 2 *List Context Relation Property 1 (LCRP-1)*

Given three contextualized lists L_a , L_b , and L_c , if $L_a \dashrightarrow L_b$ and $L_a \dashrightarrow L_c$, then $L_b \equiv L_c$. **Proof:**

1. $\Psi_{L_b} = \Psi_{L_a} \succ L_a$ because $L_a \dashrightarrow L_b$,
2. $\Psi_{L_c} = \Psi_{L_a} \succ L_a$ because $L_a \dashrightarrow L_c$, and
3. $L_b \equiv L_c$ because $\Psi_{L_b} = \Psi_{L_c}$.

Property 3 *List Context Relation Property 2 (LCRP-2)*

Given three contextualized lists L_a , L_b , and L_c , if $L_a \equiv L_b$ and $L_b \dashrightarrow L_c$, then $L_a \equiv (L_b \succ L_c)$. **Proof:**

1. $L_b \succ L_c$ is contextualized because $L_b \dashrightarrow L_c$,
2. $\Psi_{L_a} = \Psi_{L_b}$ because $L_a \equiv L_b$,
3. $\Psi_{L_b} = \Psi_{L_b \succ L_c} = \Upsilon_{L_b[0]}$, and
4. $L_a \equiv (L_b \succ L_c)$ because $\Psi_{L_a} = \Psi_{L_b \succ L_c}$.

The **SLOT** (Symmetric Linear Operation Transformation) control algorithm, which symmetrically transforms two context-equivalent lists L_a and L_b , and returns transformed ones L'_a and L'_b , is defined as follows.

Algorithm 1 **SLOT**(L_a, L_b): (L'_a, L'_b)

Require: $L_a \equiv L_b$

Ensure: $L_a \dashrightarrow L'_b$ and $L_b \dashrightarrow L'_a$

$L'_a \leftarrow L_a$
 $L'_b \leftarrow L_b$

for ($i = 0$; $i < |L'_a|$; $i++$) **do**

for ($j = 0$; $j < |L'_b|$; $j++$) **do**

$Op_{a,i}^j \leftarrow L'_a[i]$

$Op_{b,j}^i \leftarrow L'_b[j]$

$(Op_{a,i}^{j+1}, Op_{b,j}^{i+1}) \leftarrow \mathbf{SIT}(Op_{a,i}^j, Op_{b,j}^i)$

$L'_a[i] \leftarrow Op_{a,i}^{j+1}$
 $L'_b[j] \leftarrow Op_{b,j}^{i+1}$
 end for
end for

return (L'_a, L'_b)

The **SIT** (Symmetric Inclusion Transformation) function symmetrically transforms two context-equivalent operations. Its pre-condition and post-conditions are directly based on those of **IT** functions.

Function 1 **SIT**(Op_a, Op_b): (Op'_a, Op'_b)

Require: $Op_a \sqcup Op_b$

Ensure: $Op_a \mapsto Op'_b$ and $Op_b \mapsto Op'_a$

$Op'_a \leftarrow \mathbf{IT}(Op_a, Op_b)$

$Op'_b \leftarrow \mathbf{IT}(Op_b, Op_a)$

return (Op'_a, Op'_b)

The **SLOT** control algorithm can be implemented as a function **SLOT**(L_a, L_b): (L'_a, L'_b), where original lists and transformed ones are kept separately, or as a procedure **SLOT**(L_a, L_b), where the original lists are replaced by transformed ones.

If the pre-condition of a **SLOT** algorithm invocation is satisfied, the algorithm can ensure the pre-condition of every **SIT** function invocation is satisfied. If the post-condition of every **IT** function invocation is satisfied by the definitions of **IT** functions, the post-conditions of the **SLOT** algorithm invocation are automatically satisfied. The post-conditions of **SLOT**, $L_a \dashrightarrow L'_b$ and $L_b \dashrightarrow L'_a$, have two implications. One is that $L_a \succ L'_b$ and $L_b \succ L'_a$ are also contextualized. The other is that operations in the list L'_b (or L'_a) are ready to be replayed in sequence after having taking into account the impact of L_a (or L_b) by transformation.

Furthermore, to preserve *convergence*, **IT** functions must meet the following *TP-1* (Transformation Property 1).

Definition 8 (*List Equivalent Relation*).

Given two lists L_a and L_b , where $L_a \equiv L_b$, L_a is equivalent to L_b , denoted by $L_a \equiv L_b$, iff $\Psi_{L_a} \succ L_a = \Psi_{L_b} \succ L_b$.

Property 4 *Transformation Property 1 (TP-1)*

Given two operations Op_a and Op_b , where $Op_a \sqcup Op_b$, if **SIT**(Op_a, Op_b): (Op'_a, Op'_b), then $[Op_a, Op'_b] \equiv [Op_b, Op'_a]$.

Given two operations Op_a and Op_b concurrently generated at two collaborating sites and propagated to each other for consistency maintenance, *TP-1* is to ensure convergence in the sense that $\Upsilon_{Op_a} \vdash Op_a \vdash Op'_b = \Upsilon_{Op_b} \vdash Op_b \vdash Op'_a$. Although the two operations are executed in different orders at the two sites, **IT** functions must ensure an identical context after both operations have been executed at the two sites.

To use **SLOT** to control transformation in asynchronous collaborative systems, each collaborating site needs to maintain two linear buffers: *OB* (Outgoing Buffer) for storing unpropagated local operations and *IB* (Incoming Buffer) for storing unreplayed remote operations. The following example illustrates how the **SLOT** control algorithm preserves convergence and the intentions of operations by means of **OT**. As shown in Figure 2, consider a collaborative drawing session involving two sites, where the shared data source initially consists of three objects: Ob_0 (a triangle), Ob_1 (a circle), and Ob_2 (a square), and the data addressing domain *Drawings* initially contains

$[Ob_0, Ob_1, Ob_2]$. The two buffers at the two sites, namely OB_1 and IB_1 (OB and IB at *site 1*), OB_2 and IB_2 (OB and IB at *site 2*), are all empty initially.

At *site 1*, operation $Op_1 = \mathbf{Delete}([(Drawings, 1)])$ is executed to remove the object at address $[(Drawings, 1)]$, which is Ob_1 (circle). After that, $Drawings$ contains $[Ob_0, Ob_2]$, OB_1 contains $[Op_1]$, and IB_1 remains empty. Concurrently at *site 2*, operations $Op_2 = \mathbf{Delete}([(Drawings, 0)])$ and $Op_3 = \mathbf{Update}([(Drawings, 1)], fill, b)$ are executed in sequence to first remove the object at address $[(Drawings, 0)]$, which is Ob_0 (rectangle) and then fill the object at address $[(Drawings, 1)]$, which is Ob_2 (square), with the *black* color. After that, $Drawings$ contains $[Ob_1, Ob_2]$, OB_2 contains $[Op_2, Op_3]$, and IB_2 remains empty.

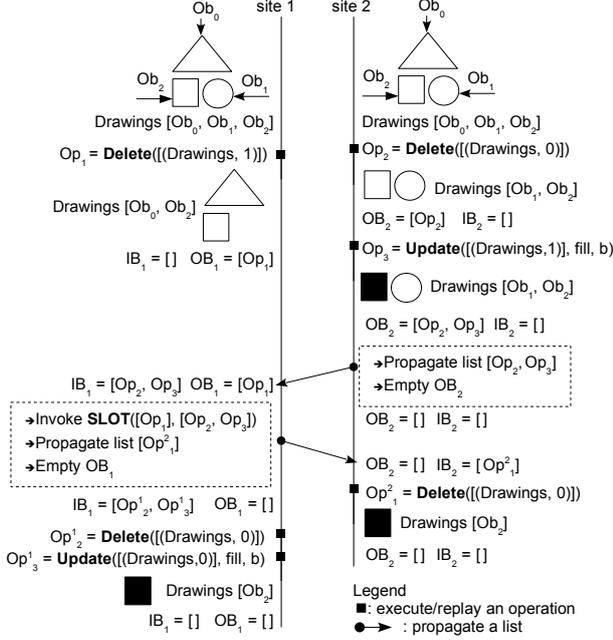


Figure 2: Consistency maintenance by SLOT

At *site 2*, because IB_2 is empty, list $[Op_2, Op_3]$ from OB_2 are propagated as-is to *site 1* and OB_2 is emptied. When the list arrives at *site 1*, it is appended to IB_1 . At *site 1*, because IB_1 is not empty, to propagate list $[Op_1]$ from OB_1 to *site 2*, procedure $\mathbf{SLOT}([Op_1], [Op_2, Op_3])$ needs to be invoked as follows. The pre-condition of this procedure invocation has been satisfied, i.e., $[Op_1] \sqcap [Op_2, Op_3]$, because $\Psi_{[Op_1]} = \Psi_{[Op_2, Op_3]} = Drawings[Ob_0, Ob_1, Ob_2]$.

1. $\mathbf{SIT}(Op_1, Op_2)$: (Op_1^1, Op_2^1) , where $\mathbf{IT_DD}(Op_1, Op_2)$: $Op_1^1 = \mathbf{Delete}([(Drawings, 0)])$ and $\mathbf{IT_DD}(Op_2, Op_1)$: $Op_2^1 = \mathbf{Delete}([(Drawings, 0)])$. The pre-condition is satisfied, i.e., $Op_1 \sqcap Op_2$ because $\Upsilon_{Op_1} = \Upsilon_{Op_2} = Drawings[Ob_0, Ob_1, Ob_2]$. The post-conditions are $Op_1 \mapsto Op_1^1$ and $Op_2 \mapsto Op_1^1$.
2. $\mathbf{SIT}(Op_1^1, Op_3)$: (Op_1^2, Op_3^1) , where $\mathbf{IT_DU}(Op_1^1, Op_3)$: $Op_1^2 = \mathbf{Delete}([(Drawings, 0)])$ and $\mathbf{IT_UD}(Op_3, Op_1^1)$: $Op_3^1 = \mathbf{Update}([(Drawings, 0)], fill, b)$. Because $Op_2 \mapsto Op_1^1$ and $Op_2 \mapsto Op_3$, by Property 1: $OCRP$, the pre-condition is also satisfied, i.e., $Op_1^1 \sqcap Op_3$. The post-conditions are $Op_1^1 \mapsto Op_1^2$ and $Op_3 \mapsto Op_1^2$.

After the \mathbf{SLOT} invocation, $OB_1 = [Op_1^2]$ and $IB_1 = [Op_1^2, Op_3^1]$. The post-conditions of the \mathbf{SLOT} invocation are satisfied, i.e., $[Op_2, Op_3] \dashrightarrow [Op_1^2]$ (or

$Op_2 \mapsto Op_3 \mapsto Op_1^2$) and $[Op_1] \dashrightarrow [Op_1^2, Op_3^1]$ (or $Op_1 \mapsto Op_1^2 \mapsto Op_3^1$), because:

1. $Op_2 \mapsto Op_3$ and $Op_3 \mapsto Op_1^2$ (by the post-conditions of the second \mathbf{SIT} invocation), and
2. $Op_1 \mapsto Op_1^2$ (by the post-conditions of the first \mathbf{SIT} invocation) and $Op_1^2 \mapsto Op_3^1$ (by the following deduction):
 - (a) $\Upsilon_{Op_3^1} = \Upsilon_{Op_1^2} \vdash Op_1^2$ because $Op_1^2 \mapsto Op_3^1$ (by the post-conditions of the second \mathbf{SIT} invocation);
 - (b) $\Upsilon_{Op_1^2} = \Upsilon_{Op_2} \vdash Op_2$ because $Op_2 \mapsto Op_1^2$ (by the post-conditions of the first \mathbf{SIT} invocation);
 - (c) $\Upsilon_{Op_2} \vdash Op_2 \vdash Op_1^1 = \Upsilon_{Op_1} \vdash Op_1 \vdash Op_1^1$ (by TP-1 of the first \mathbf{SIT} invocation); and
 - (d) $Op_1^2 \mapsto Op_3^1$ because $\Upsilon_{Op_3^1} = \Upsilon_{Op_1^2} \vdash Op_1^2$ (by step a) = $\Upsilon_{Op_2} \vdash Op_2 \vdash Op_1^1$ (by step b) = $\Upsilon_{Op_1} \vdash Op_1 \vdash Op_1^1$ (by step c).

Then the list $[Op_1^2]$ is propagated to *site 2* and OB_1 is emptied. When the list arrives at *site 1*, it is appended to IB_2 . At *site 2*, because $[Op_2, Op_3] \dashrightarrow [Op_1^2]$ and OB_2 is empty, the remote operation Op_1^2 in IB_2 can be replayed as-is on the current context (i.e., the context right after the execution of local operations Op_2 and Op_3). When $Op_1^2 = \mathbf{Delete}([(Drawings, 0)])$ is replayed, the object at address $[(Drawings, 0)]$, which is Ob_1 (circle), is removed. After that, IB_2 is emptied.

Back to *site 1*, because $[Op_1] \dashrightarrow [Op_1^2, Op_3^1]$ and OB_1 is empty, remote operations Op_1^2 and Op_3^1 in IB_1 can be replayed in sequence as-is on the current context (i.e., the context right after the execution of local operation Op_1). When $Op_1^2 = \mathbf{Delete}([(Drawings, 0)])$ is replayed, the object at address $[(Drawings, 0)]$, which is Ob_0 (rectangle), is removed. Subsequently, when $Op_3^1 = \mathbf{Update}([(Drawings, 0)], fill, b)$ is replayed, the object at address $[(Drawings, 0)]$, which is Ob_2 (square), is filled with the *black* color. After that, IB_1 is emptied. It is clear that the intentions of all operations are preserved and the two sites are convergent at the end of the collaborative session.

3.3 Verification of the SLOT TCA

The above example has shown that the \mathbf{SLOT} control algorithm can preserve convergence and the intentions of operations. To systematically prove its correctness, we need to verify it against the *intention preservation* and *convergence* consistency properties. However, due to space limitation, we only formally verify the *intention preservation* property. The *causality preservation* property has nothing to do with \mathbf{SLOT} and will be discussed in a later section.

Transformation function $\mathbf{IT}(Op_a, Op_b)$: Op'_a can preserve the intention of Op_a by including the impact of Op_b into Op'_a , provided that the pre-condition $Op_a \sqcap Op_b$ is satisfied. Therefore, to prove \mathbf{SLOT} control algorithm can achieve *intention preservation*, we only need to prove it can ensure the pre-condition of every \mathbf{SIT} function invocation is satisfied, as described by Theorem 1.

Theorem 1 Given two lists L_a and L_b , where $L_a \sqcap L_b$, $\mathbf{SLOT}(L_a, L_b)$ ensures the pre-condition of every \mathbf{SIT} function invocation is satisfied.

Proof: Assume that $L_a = [Op_{a,0}, \dots, Op_{a,m-1}]$ ($m = |L_a|$) and $L_b = [Op_{b,0}, \dots, Op_{b,n-1}]$ ($n = |L_b|$). Furthermore, $Op_{a,i}^k$ ($0 \leq i < m$ and $0 \leq k \leq n$) denotes the transformed $Op_{a,i}$ that has been transformed against the first k operations in L_b , and $Op_{b,j}^l$ ($0 \leq j < n$ and $0 \leq l \leq m$) denotes the transformed $Op_{b,j}$ that has been transformed against the first l operations in L_a . In particular, $Op_{a,i}^0 = Op_{a,i}$ ($0 \leq i < m$) and $Op_{b,j}^0 = Op_{b,j}$ ($0 \leq j < n$).

SLOT(L_a, L_b) needs to perform $m \times n$ **SIT** function invocations, namely, **SIT**($Op_{a,i}^j, Op_{b,j}^i$): ($Op_{a,i}^{j+1}, Op_{b,j}^{i+1}$), where $i = 0, \dots, m-1$ and $j = 0, \dots, n-1$. We need to prove that for $\forall i \in \{0, \dots, m-1\}$ and $\forall j \in \{0, \dots, n-1\}$, $Op_{a,i}^j \sqcup Op_{b,j}^i$ holds.

For $m = 1$, $L_a = [Op_{a,0}]$, and **SLOT**(L_a, L_b) needs to perform n **SIT** function invocations, namely **SIT**($Op_{a,0}^j, Op_{b,j}$): ($Op_{a,0}^{j+1}, Op_{b,j}^1$), where $j = 0, \dots, n-1$. The theorem holds, i.e., for $\forall j \in \{0, \dots, n-1\}$, $Op_{a,0}^j \sqcup Op_{b,j}$ holds, because:

- for $n = 1$, $L_b = [Op_{b,0}]$, $Op_{a,0} \sqcup Op_{b,0}$ obviously holds because $L_a \amalg L_b$,
- for $n = N$, hypothesize that for $\forall j \in \{0, \dots, N-1\}$, $Op_{a,0}^j \sqcup Op_{b,j}$ holds(\boxtimes),
- for $n = N+1$, one more **SIT**($Op_{a,0}^N, Op_{b,N}$): ($Op_{a,0}^{N+1}, Op_{b,N}^1$) invocation is required, and by Property 1: *OCRP*, $Op_{a,0}^N \sqcup Op_{b,N}$ also holds because:
 1. $Op_{b,N-1} \mapsto Op_{b,N}$ as L_b is contextualized; and
 2. $Op_{b,N-1} \mapsto Op_{a,0}^N$ as a result of the post-conditions of **SIT**($Op_{a,0}^{N-1}, Op_{b,N-1}$): ($Op_{a,0}^N, Op_{b,N-1}^1$) invocation, which is legitimate since $Op_{a,0}^{N-1} \sqcup Op_{b,N-1}$ (by the induction hypothesis \boxtimes), and
- further based on the induction hypothesis \boxtimes , for $\forall j \in \{0, \dots, N\}$, $Op_{a,0}^j \sqcup Op_{b,j}$ holds, and therefore by the induction argument, for any n and for $\forall j \in \{0, \dots, n-1\}$, $Op_{a,0}^j \sqcup Op_{b,j}$ holds.

For $m = M$, hypothesize that the theorem holds, i.e., for $\forall i \in \{0, \dots, M-1\}$ and $\forall j \in \{0, \dots, n-1\}$, $Op_{a,i}^j \sqcup Op_{b,j}^i$ holds(\dagger).

For $m = M+1$, n more **SIT** invocations are required, namely, **SIT**($Op_{a,M}^j, Op_{b,j}^M$): ($Op_{a,M}^{j+1}, Op_{b,j}^{M+1}$), where $j = 0, \dots, n-1$. For $\forall j \in \{0, \dots, n-1\}$, $Op_{a,M}^j \sqcup Op_{b,j}^M$ holds because:

- for $n = 1$, $L_b = [Op_{b,0}]$, and by Property 1: *OCRP*, $Op_{a,M} \sqcup Op_{b,0}^M$ holds because:
 1. $Op_{a,M-1} \mapsto Op_{a,M}$ as L_a is contextualized; and
 2. $Op_{a,M-1} \mapsto Op_{b,0}^M$ as a result of the post-conditions of **SIT**($Op_{a,M-1}, Op_{b,0}^{M-1}$): ($Op_{a,M-1}^1, Op_{b,0}^M$) invocation, which is legitimate since $Op_{a,M-1} \sqcup Op_{b,0}^{M-1}$ (by the induction hypothesis \dagger),
- for $n = N$, hypothesize that for $\forall j \in \{0, \dots, N-1\}$, $Op_{a,M}^j \sqcup Op_{b,j}^M$ holds(\ast), and

- for $n = N+1$, one more **SIT**($Op_{a,M}^N, Op_{b,N}^M$): ($Op_{a,M}^{N+1}, Op_{b,N}^{M+1}$) is required, and $Op_{a,M}^N \sqcup Op_{b,N}^M$ also holds because:

1. $Op_{b,N-1}^M \mapsto Op_{a,M}^N$ as a result of the post-conditions of **SIT**($Op_{a,M}^{N-1}, Op_{b,N-1}^M$): ($Op_{a,M}^N, Op_{b,N-1}^{M+1}$), which is legitimate since $Op_{a,M}^{N-1} \sqcup Op_{b,N-1}^M$ (by the induction hypothesis \ast);
2. $Op_{a,M-1}^{N-1} \mapsto Op_{b,N-1}^M$ and $Op_{b,N-1}^{M-1} \mapsto Op_{a,M-1}^N$ as the results of the post-conditions of **SIT**($Op_{a,M-1}^{N-1}, Op_{b,N-1}^{M-1}$): ($Op_{a,M-1}^N, Op_{b,N-1}^M$), which is legitimate since $Op_{a,M-1}^{N-1} \sqcup Op_{b,N-1}^{M-1}$ (by the induction hypothesis \dagger);
3. $Op_{a,M-1}^N \mapsto Op_{b,N}^M$ as a result of the post-conditions of **SIT**($Op_{a,M-1}^N, Op_{b,N}^{M-1}$): ($Op_{a,M-1}^N, Op_{b,N}^M$), which is legitimate since $Op_{a,M-1}^N \sqcup Op_{b,N}^{M-1}$ (by the induction hypothesis \dagger);
4. $\Upsilon_{Op_{b,N-1}^{M-1}} \vdash Op_{b,N-1}^{M-1} \vdash Op_{a,M-1}^N = \Upsilon_{Op_{a,M-1}^{N-1}} \vdash Op_{a,M-1}^{N-1} \vdash Op_{b,N-1}^M$ (by TP-1 of **SIT**($Op_{a,M-1}^{N-1}, Op_{b,N-1}^{M-1}$): ($Op_{a,M-1}^N, Op_{b,N-1}^M$));
5. $\Upsilon_{Op_{b,N}^M} = \Upsilon_{Op_{a,M}^N}$ because $\Upsilon_{Op_{b,N}^M} = \Upsilon_{Op_{b,N-1}^{M-1}} \vdash Op_{b,N-1}^{M-1} \vdash Op_{a,M-1}^N$ (by step 2 and 3, and the definition of “ \mapsto ”) = $\Upsilon_{Op_{a,M-1}^{N-1}} \vdash Op_{a,M-1}^{N-1} \vdash Op_{b,N-1}^M$ (by step 4) = $\Upsilon_{Op_{a,M}^N}$ (by step 1 and 2, and the definition of “ \mapsto ”); and
6. further based on the induction hypothesis \ast , for $\forall j \in \{0, \dots, N\}$, $Op_{a,M}^j \sqcup Op_{b,j}^M$ holds, and therefore by the induction argument, for any n and for $\forall j \in \{0, \dots, n-1\}$, $Op_{a,M}^j \sqcup Op_{b,j}^M$ holds.

Further based on the induction hypothesis \dagger , for $m = M+1$, the theorem also holds, i.e., for $\forall i \in \{0, \dots, M\}$ and $\forall j \in \{0, \dots, n-1\}$, $Op_{a,i}^j \sqcup Op_{b,j}^i$ holds. By the induction argument, the theorem holds, i.e., for any m and n , and for $\forall i \in \{0, \dots, m-1\}$ and $\forall j \in \{0, \dots, n-1\}$, $Op_{a,i}^j \sqcup Op_{b,j}^i$ holds. The theorem has hereby been proved.

4 Protocols for Achieving Contextualization

If the pre-condition of every **SLOT** invocation is satisfied, *intention preservation* and *convergence* can be achieved by OT (e.g., by Theorem 1). The question is how to ensure the pre-condition of every **SLOT** invocation, i.e., the two lists to be transformed must be contextualized and context-equivalent.

To facilitate asynchronous collaboration, each collaborating site maintains an *OB* and an *IB* to separate local and remote operations, where *OB* stores unpropagated local operations and *IB* stores unreplayed remote operations. On the one hand, operations generated at a local site are executed instantly to gain good responsiveness and then appended to *OB*, waiting to be propagated via an operation propagation

protocol. On the other hand, operations propagated from remote sites are appended to IB , waiting to be replayed locally via an operation replaying protocol. The two protocols are essential to ensure OB and IB at any site are contextualized and context-equivalent.

4.1 CCOP: A Coordinated Operation Propagation Protocol

Most synchronous systems do not use coordinated operation propagation protocols. In these systems, any site can freely propagate new operations upon generation without coordinating propagation requests from other sites. For example *GOT*, *GOTO*, *adOPTed*, and *COT* rely on timestamping individual operations with state vectors to capture causal/concurrent relationships among operations. Some systems adopt coordinated propagation protocols for the purpose of reducing the complexity of TCAs or relaxing some constraints on transformation functions. *Jupiter* fall into this category.

We adopt a coordinated operation propagation protocol to achieve contextualization and list context equivalence. Enforced by a coordination rule, a site can only initiate a new propagation (of a list of operations) when there are no outstanding propagations, in other words, when all preceding propagations have been received by all designated sites. The coordination rule can be warranted by employing a sequencer or a token manager, or by running a centralized or distributed synchronization protocol.

At site s , to propagate all operations from OB_s , the **CCOP** (Coordinated Contextualized Operation Propagation) protocol is executed as follows.

Protocol 1 CCOP

1. Start the protocol by following the coordination rule.
2. Skip this step if IB_s is empty. Otherwise, transform the list from OB_s with the one from IB_s by **SLOT**(OB_s, IB_s).
3. Propagate the list from OB_s to designated sites and empty OB_s .
4. Append the propagated list to IB_t when the propagation arrives at a designated site t .
5. End the protocol when the propagation has arrived at all designated sites.

To ensure good local responsiveness, new local operations are allowed to be generated at any time, even during the execution of the **CCOP** protocol. Operations generated in the meantime are first stored in a temporary buffer and then moved to OB_s after the protocol is ended. The protocol may take a while to end, especially when the number of collaborating sites is big or the propagated list is big, but the protocol execution time has nothing to do with the local responsiveness (it only affects when local operations are to be replayed at remote sites).

It is worth pointing out that coordinated operation propagation may become a performance bottleneck in synchronous systems with a large number of collaborating sites. However, it is generally not a problem in asynchronous systems because: 1) the frequency of propagations is normally not high; 2) consequently, cases where multiple sites request for propagating operations at the same time are not common; and 3) even some cases do exist, postponement of some propagations would not affect the system performance as synchronous notification of each other's progress is not expected. Nevertheless, for the **SLOT**

control algorithm to be used in synchronous systems, the propagation protocol can be optimized in various ways. For example, The **SCOP** protocol used by the *NICE* collaborative editing system adopts a notification server to synchronize concurrent propagations before broadcasting them to destination sites (Shen & Sun 2002).

4.2 ACOR: An Adaptive Operation Replay-ing Protocol

At any collaborating site, when local operations are to be propagated from OB , the **CCOP** protocol ensures contextualization of OB and IB and context equivalence between them. Similarly, when remote operations are to be replayed from IB , an operation replaying protocol is required to ensure contextualization of OB and IB and context equivalence between them.

Because local and remote operations modify the same shared data source, execution of local operations and replaying of remote operations must be mutually exclusive. Furthermore, in case of contention between local and remote operations, local operations must be given the priority to ensure a good local responsiveness. Therefore, remote operations are ideally to be replayed when no local operations are being executed. However, it is infeasible to predict when a user is going to issue operations. In synchronous systems, remote operations arrive at a site one-by-one and each remote operation is replayed when no local operation is being executed at that site. If the user at that site issues an operation in the meantime when a remote operation is being replayed, execution of the local operation will be delayed until replaying of the remote operation is over. Postponement of the local operation execution by one remote operation replaying time would not affect local responsiveness much as replaying one remote operation usually takes very little time.

However, in asynchronous systems, remote operations arrive at a site batch-by-batch and each batch could consist of as many as hundreds of operations. If all operations in IB at that site were replayed as a continual stream, local operations would suffer starvation, resulting in poor local responsiveness. To tackle this issue, after replaying a remote operation from IB , the operation replaying protocol should give local operations a chance to execute (if any) before replaying the next one.

At site t , to replay all operations from IB_t , the **ACOR** (Adaptive Contextualized Operation Replay-ing) protocol is executed as follows.

Protocol 2 ACOR

```

1:  $s \leftarrow 0$ ;
2: if ( $|OB_t| > s$ ) then
3:    $e \leftarrow |OB_t| - 1$ ;
4:   SLOT( $IB_t, OB_t[s, e]$ );
5:    $s \leftarrow e + 1$ ;
6: end if
7: while ( $|IB_t| > 0$ ) do
8:   give_way();
9:   acquire_lock();
10:  if ( $|OB_t| > s$ ) then
11:     $e \leftarrow |OB_t| - 1$ ;
12:    SLOT( $IB_t, OB_t[s, e]$ );
13:     $s \leftarrow e + 1$ ;
14:  end if
15:   $Op \leftarrow IB_t[0]$ ;
16:  replay_operation( $Op$ );
17:  remove_operation( $IB_t, 0$ );
18:  release_lock();
19: end while
    
```

When site t is in the process of executing **ACOR** to replay remote operations from IB_t , it can still receive remote operations, which are first temporarily buffered and then moved to IB_t after the protocol is ended. This implies that IB_t will not grow during the protocol execution, and newly arrived remote operations will be replayed in the subsequent **ACOR** protocol execution. In contrast, OB_t may grow and the protocol must ensure operations in IB_t are transformed with all operations in OB_t no matter when these operations are generated and appended to OB_t (lines 2-6 or 10-14).

To minimize the impact on local responsiveness, each remote operation should “give way” to local operations (line 8). The *give_way()* procedure halts the protocol execution until the data source is not being modified by any local operation. Checking the state of the data source is done by a thread, which alternates between checking and sleeping (if the data source is unavailable) until the data source becomes available. The sleeping time is adaptive in the sense that it is dynamically calculated according to the type of the local operation being executed and the frequency in which recent local operations have been generated.

When a remote operation is ready to be replayed, the protocol will first exclusively lock the data source to prevent it from being modified by local operations (line 9). During the locking period, raw local events such as keystrokes may be buffered for the generation of new local operations. Then operations in IB_t must be transformed (line 10-14) with new local operations (generated in the meantime when the remote operation was giving way). Finally, after the remote operation is replayed (line 16), it is removed from IB_t (line 17) and the lock on the data source is released (line 18). After that, the next operation in IB_t will look for its chance to be replayed and the protocol proceeds until IB_t becomes empty.

4.3 Verification of the CCOP and ACOR Protocols

The example in Figure 2 has actually used the **CCOP** protocol to propagate local operations and the **ACOR** protocol to replay remote operations. It has shown that *intention preservation* and *convergence* have been achieved by using the two protocols in a collaborative session involving two sites. To systematically prove the two protocols together can maintain consistency in a distributed collaborative session involving arbitrary number of collaborating sites (two or more), we need to verify them against the three consistency properties.

For *causality preservation*, we need to prove that given any two operations Op_a and Op_b , if Op_a is causally before Op_b , then Op_a must be executed before Op_b at all sites. For *intention preservation*, we need to prove that all **SLOT** invocations in **CCOP** and **ACOR** have satisfied their pre-conditions. For *convergence*, we need to prove that the context of the shared data source is identical across all sites after all operations have been executed at all sites. However, due to space limitation, we only formally verify the *causality preservation*, as described by Theorem 2.

Theorem 2 *Given any two operations Op_a and Op_b , if Op_a is causally before Op_b , then Op_a must be executed before Op_b at all collaborating sites.*

Proof: There are two possibilities. One is that Op_a is generated before Op_b at the same site, e.g., site i . If Op_b is generated before the propagation of Op_a , Op_a will be positioned before Op_b in OB_i . There are two cases in propagating Op_a and Op_b . First, if Op_a and Op_b are propagated in the same list (i.e.,

in one **CCOP** protocol execution), for any remote site t , Op_a and Op_b will arrive at site t and be appended to IB_t at the same time, where Op_a is still positioned before Op_b . Then the execution of **ACOR** protocol at site t ensures Op_a is replayed before Op_b . Second, if Op_a and Op_b are propagated in two lists (i.e., in two **CCOP** protocol executions), the execution of two **CCOP** protocols in sequence ensures the propagation containing operation Op_a arrives at any remote site before the one containing operation Op_b does. At any remote site, Op_a is always replayed before Op_b no matter whether they are replayed in one **ACOR** protocol execution or in two **ACOR** protocol executions in sequence. If Op_b is generated after the propagation of Op_a , it will be the same as the second case.

Another possibility is that Op_a and Op_b are generated at two sites, e.g., site i and j respectively, and Op_b is generated after the execution of Op_a at site j . This implies that the propagation containing operation Op_a must have arrived at site j before Op_b is generated. According to the **CCOP** protocol, the propagation containing operation Op_b cannot be initiated before the one containing operation Op_a has arrived at all sites. This ensures the propagation containing operation Op_a must arrive at any remote site before the one containing operation Op_b does. Furthermore, at any remote site, Op_a is always replayed before Op_b no matter whether they are replayed in one **ACOR** execution or in two **ACOR** executions in sequence. The theorem has hereby been proved.

5 Significance and Applications

First of all, most TCAs designed for synchronous systems, such as *dOPT*, *GOT*, *GOTO*, *adOPTed*, and *COT*, require every operation to be timestamped with a state vector in order to capture their causal/concurrent relationships. The **SLOT** control algorithm, specifically designed for asynchronous systems, does not require operations to be timestamped because their causal/concurrent relationships are captured by separating local and remote operations in two buffers (*OB* and *IB*) and by enforcing operation propagation and replaying protocols (**CCOP** and **ACOR**) to achieve contextualization and list context equivalence.

Second, most TCAs designed for synchronous systems avoid using ET (Exclusion Transformation) functions (Sun et al. 1998) because it is difficult to design ET functions that satisfy the *Reversibility Property - ET(IT(Op_a , Op_b), Op_b) = Op_a)* (Sun & Sun 2006, ?). Function **ET**(Op_a , Op_b) transforms Op_a against Op_b in such a way that the impact of Op_b is effectively excluded in the output operation Op'_a . ET functions are required by algorithms, such as *GOT*, *GOTO*, and *SOCT3*, which are based on a single linear history buffer to store all executed local and remote (transformed) operations. The **SLOT** control algorithm has also avoided using ET functions.

Third, in addition to Property 4: *TP-1*, which is essential for preserving convergence, IT functions need to satisfy *Transformation Property 2 (TP-2)*, which requires IT functions to be defined in such a way that transformation of an operation against a set of mutually concurrent operations is irrelevant of the order in which the set of operations are mutually transformed (Sun & Ellis 1998).

Property 5 Transformation Property 2 (TP-2)

Given three concurrent operations Op_a , Op_b , and Op_c , if $Op'_a = \mathbf{IT}(Op_a, Op_b)$ and $Op'_b = \mathbf{IT}(Op_b, Op_a)$, then $\mathbf{IT}(\mathbf{IT}(Op_c, Op_a), Op'_b) = \mathbf{IT}(\mathbf{IT}(Op_c, Op_b), Op'_a)$.

While it is relatively easy to satisfy $TP-1$, it is rather difficult to design IT functions and verify them against $TP-2$ that is also essential for preserving convergence. The violation of $TP-2$ is mainly caused by uncoordinated operation propagation, where the same set of concurrent operations may arrive at different collaborating sites in different orders and hence may be transformed in different orders. TCAs subject to $TP-2$ include *dOPT*, *GOTO*, and *adOPTed*.

As it is non-trivial to satisfy $TP-2$, some TCAs choose to break its pre-condition either by turning to coordinated operation propagation, which forces the same set of concurrent operations to arrive at all collaborating sites in the same order or by re-ordering the set of concurrent operations according to a total order so that they are transformed in the same order at all sites. *Jupiter* belongs to the first category, while *GOT* and *COT* belong to the second category. Some work takes a different approach that solves the $TP-2$ problem instead of avoiding it. For example, SDT solved the problem by fixing problems in IT functions (Li & Li 2004) and a new OT framework was proposed to tackle $TP-2$ and other OT-related problems (Li & Li 2007). **SLOT** belongs to the first category because **CCOP** ensures the same set of operation lists arrive at different collaborating sites in the same order. Furthermore, because **SLOT** is only invoked between the lists in *OB* and *IB* at the same site, it is never possible for a list in *OB* to be transformed with lists in multiple *IBs* and in different orders.

Lying on the operation propagation and replaying protocols to achieve contextualization and list context equivalence, the **SLOT** control algorithm can be used in a wide range of distributed collaborative systems. It is particularly more efficient than other algorithms when used in asynchronous systems. This OT-based concurrency control solution, including the **SLOT** algorithm, and **CCOP** and **ACOR** protocols have been implemented in a few prototype collaborative systems, e.g. the *CoEclipse* system that supports collaborative programming.

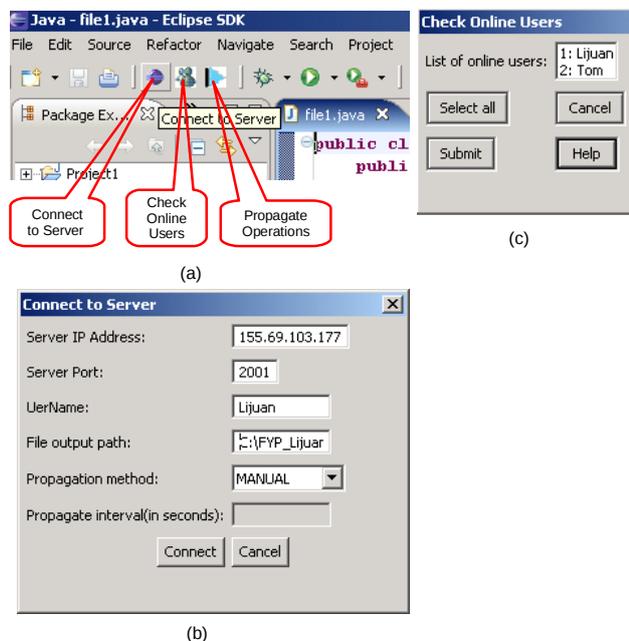


Figure 3: CoEclipse

Eclipse is well-known for the Java IDE (Integrated Development Environment). We developed *CoEclipse* to support collaborative programming in Java - a group of programmers can code the same Java project or source code file asynchronously or synchronously.

As shown in Figure 3(a), *CoEclipse* appears as three plug-in buttons on the *Eclipse* user interface, which are *Connect to Server*, *Check Online Users*, and *Propagate Operations* from left to right.

To propagate operations made on the shared source code to other programmers online, one needs to first connect to the collaboration server by pushing the *Connect to Server* button. The server is used to manage the list of online programmers and to facilitate running the operation propagation protocol. Figure 3(b) shows the dialog box before the connection is made. If the propagation method is set to *MANUAL*, each propagation has to be initiated by pushing the *Propagate Operations* button and the collaboration mode is asynchronous. If the propagation method is set to *AUTOMATIC* with a short interval, the collaboration mode is synchronous instead. One has the flexibility of choosing online programmers to whom her/his operations will be propagated by pushing the *Check Online Users* button. As shown in Figure 3(c), one may choose to propagate to all online programmers or only some of them.

6 Conclusions and Future Work

A collaborative system differs from other distributed systems by three characteristic components: multiple human users, interactive applications, and the high-latency Internet. Concurrency control is the key to managing the contention for distributed data resources and to ensure the orderly access to shared data. It is a major issue in most distributed systems and collaborative systems are of no exception. However, concurrency control techniques devised for distributed systems are generally unsuitable for collaborative systems because these characteristic components have special concurrency control requirements.

OT is a concurrency control technique originally invented for synchronous collaborative systems to meet those special requirements. However, to apply OT to asynchronous systems, a major technical challenge is to devise an efficient transformation control algorithm because existing algorithms are catered for synchronous systems only and are inefficient to be used in asynchronous systems.

Our major contribution in this paper is a novel, efficient, contextualization-based transformation control algorithm **SLOT** underpinned by an operation propagation protocol **CCOP** and an operation replaying protocol **ACOR** to achieve contextualization and list context equivalence. This algorithm is particularly more efficient than existing ones when used in asynchronous systems, and can also be used to support a wide spectrum of collaboration paradigms because it is as efficient when used in synchronous systems. Furthermore, the algorithm has most of the merits that some of existing algorithms have achieved, such as no timestamping of operations, no ET functions, and free of $TP-2$.

The correctness of the algorithm and protocols in terms of consistency maintenance has been formally verified. The usefulness of the algorithm and protocols has been demonstrated by a variety of prototype collaborative applications. We are currently investigating other collaborative techniques such as collaborative undo and asymmetric collaboration, by extending the solution presented in this paper.

Acknowledgment

The authors wish to thank Ms Lijuan Geng for the initial development of the *CoEclipse* prototype at Nanyang Technological University in Singapore.

References

- Bernstein, P., Goodman, N. & Hadzilacos, V. (1987), *Concurrency Control and Recovery in Database Systems*, Addison-Wesley.
- Ellis, C. & Gibbs, S. (1989), Concurrency control in groupware systems, in 'Proceedings of ACM SIGMOD Conference on Management of Data', pp. 399–407.
- Greenberg, S. & Marwood, D. (1994), Real time groupware as a distributed system: concurrency control and its effect on the interface, in 'Proceedings of ACM Conference on Computer Supported Cooperative Work', ACM Press, pp. 207–217.
- Gu, N., Yang, J. & Zhang, Q. (2005), Consistency maintenance based on the mark and retrace technique in groupware systems, in 'Proceedings of the ACM Conference on Supporting Group Work', ACM Press, pp. 264–273.
- Karsenty, A. & Beaudouin-Lafon, M. (1993), An algorithm for distributed groupware applications, in 'Proceedings of 13th Conference on Distributed Groupware Computing Systems', pp. 195–202.
- Knister, M. & Prakash, A. (1993), 'Issues in the design of a toolkit for supporting multiple group editors', *Journal of Usenix Association* **6**(2), 135–166.
- Li, D. & Li, R. (2004), Ensuring content and intention consistency in real-time group editors, in 'Proceedings of the IEEE International Conference on Distributed Computing Systems', pp. 748–755.
- Li, R. & Li, D. (2007), 'A new operational transformation framework for real-time group editors', *IEEE Transactions on Parallel and Distributed Systems* **18**(3), 307 – 319.
- Nichols, D. A., Curtis, P., Dixon, M. & Lamping, J. (1995), High-latency, low-bandwidth windowing in the jupiter collaboration system, in 'Proceedings of the ACM Symposium on User Interface Software and Technology', Distributed User Interfaces, pp. 111–120.
- Raynal, M. & Singhal, M. (1996), 'Logical time: capturing causality in distributed systems', *IEEE Computer Magazine* **29**(2), 49–56.
- Ressel, M., Nitsche-Ruhland, D. & Gunzenbauser, R. (1996), An integrating, transformation-oriented approach to concurrency control and undo in group editors, in 'Proceedings of ACM Conference on Computer Supported Cooperative Work', ACM Press, pp. 288–297.
- Shen, H. & Sun, C. (2002), Flexible Notification for Collaborative Systems, in 'Proceedings of ACM Conference on Computer-Supported Cooperative Work', ACM Press, pp. 77–86.
- Sun, C. & Ellis, C. (1998), Operational transformation in real-time group editors: Issues, algorithms, and achievements, in 'Proceedings of ACM Conference on Computer Supported Cooperative Work', pp. 59–68.
- Sun, C., Jia, X., Zhang, Y., Yang, Y. & Chen, D. (1998), 'Achieving convergence, causality-preservation, and intention-preservation in real-time cooperative editing systems', *ACM Transactions on Computer-Human Interaction* **5**(1), 63 – 108.
- Sun, C., Xia, S., Sun, D., Chen, D., Shen, H. & Cai, W. (2006), 'Transparent adaptation of single-user applications for multi-user real-time collaboration', *ACM Transactions on Computer-Human Interaction* **13**(4), 531–582.
- Sun, D. & Sun, C. (2006), Operation context and context-based operational transformation, in 'Proceedings of ACM Conference on Computer Supported Cooperative Work', pp. 279–288.

A Dynamic Archive Based Niching Particle Swarm Optimizer Using a Small Population Size

Zhaolin Zhai

Xiaodong Li

School of Computer Science and Information Technology
RMIT University, Melbourne, VIC 3001, Australia

Email: zhaolin.zhai@student.rmit.edu.au; xiaodong.li@rmit.edu.au

Abstract

Many niching techniques have been proposed to solve multimodal optimization problems in the evolutionary computing community. However, these niching methods often depend on large population sizes to locate many more optima. This paper presents a particle swarm optimizer (PSO) niching algorithm only using a dynamic archive, without relying on a large population size to locate numerous optima. To do this, we record found optima in the dynamic archive, and allow particles in converged sub-swarms to be re-randomized to explore undiscovered parts of the search space during a run. This algorithm is compared with *lbest* PSOs with a ring topology (LPRT). Empirical results indicate that the proposed niching algorithm outperforms LPRT on several benchmark multimodal functions with large numbers of optima, when using a small population size.

Keywords: Particle Swarm Optimization, Multimodal Optimization, Evolutionary Computation, a Dynamic Archive.

1 Introduction

Optimization techniques, such as Evolutionary Algorithms (EAs) (Back et al. 1997), Particle swarm optimization (PSO) (Kennedy & Eberhart 1995), and Differential Evolution (Price et al. 2005) have proven to be successful in solving difficult global optimization problems, typically characterized by searching a single global optimum. However, many real optimization problems in science and engineering do have more than one global optimal solution known as multimodal problems. For instance, in the field of wing design in aviation industry, there may be several different solutions which could perform equally well. It is desirable to locate as many optimal solutions as possible, so engineers are able to select the best solution depending on the preferred design variable ranges. To address this issue, a number of approaches to find multiple solutions have been proposed (Li et al. 2002, R. Brits & van den Bergh 2002, Bird & Li 2006, Li 2010). These approaches are generally referred to as niching or speciation algorithms. The notion of niching or speciation is originated from ecological science, in which all kinds of species (a class of individuals with common characteristics (Li et al. 2002)) are considered to be evolved in parallel by competition and species form different niches over time. In the context of EC, niches refer subpopulations

formed around global or local optima, and a niching algorithm aims to identify and maintain equally good solutions stably throughout a run (Mahfoud 1995). In a typical EC based niching algorithm, individuals in a population are partitioned somehow, generating a set of subpopulation known as species. These species are expected to be converged around different global or local optima (attraction basins) in the search space for locating different potential solutions.

Currently, existing niching algorithms largely depend on using a large population size to find large numbers of optima in the search space. However, without the prior knowledge on numbers of optimal solutions to the problem, it is extremely hard to determine how many individuals or particles are sufficient for a multimodal problem. In particular, for problems with numerous global optima, it is difficult for existing niching algorithms to locate all optima, if the population size is not sufficiently large. Furthermore, most reported experimental results on niching algorithms, is limited to low dimensional problems or higher dimensional problems with a small number of global optima (Bird & Li 2006, Li et al. 2002, Schoeman & Engelbrecht 2005, Li 2010). However, in many cases, as the dimensionality increases, the number of global and local optima in multimodal problems may go up quickly. For instance, the inverted Vincent function has 6^n number of global optima, where n is the number of dimensions.

This paper introduces a dynamic archive based PSO niching algorithm (*rpso-sp*) to alleviate the population dependence problem in niching algorithms. *rpso-sp* is able to handle multimodal optimization problems using a dynamic archive for saving best found solutions. To make each particle keep doing search in a run, a multi-start technique is employed in *rpso-sp* through re-randomizing converged sub-population, allowing sub-population explore undiscovered parts of the search space continuously. The paper is organized as follows. Section 2 gives an introduction to the basic PSO algorithms. Section 3 provides a review of some state-of-the art PSO niching techniques. Section 4 describes the newly proposed *rpso-sp* niching algorithm. Experimental results and analysis are provided in Section 5 and 6, followed by conclusions in Section 7.

2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a recently developed stochastic optimization algorithm first introduced by Kennedy & Eberhart (1995), inspired by social behaviors observed among insects and animals such as bird flocking or fish schooling. Like Genetic Algorithms (GAs), PSO uses a population of agents, referred to as particles, to form a swarm. Each particle in the swarm represents a candidate solution to an optimization problem. In a standard PSO algorithm, each particle moves through the search space by adjusting its position based on its own experience that of neighborhood particles. We denote the

i^{th} particle in the swarm as \vec{x}_i , which is evaluated as a potential solution at each iteration. The i^{th} particle moves with a velocity \vec{v}_i toward promising regions by consulting the best position \vec{p}_i (the position giving the best fitness value so far) found by itself and that of its neighbors \vec{p}_g . As to deciding neighbors for each particle, two types of neighborhood topologies are extensively used in PSO. In a global best (*gbest*) PSO, each particle influences every other particle; \vec{p}_g is obtained from the entire swarm. In the local best (*lbest*) PSO, each particle influences its immediate or close neighbors. We adopted the constriction coefficient variation of PSO (Clerc & Kennedy 2002) in this paper. Updating \vec{v}_i and \vec{x}_i for the i^{th} particle is based on the following equations:

$$\vec{v}_i = \chi(\vec{v}_i + \vec{R}_1[0, \varphi_1] \otimes (\vec{p}_i - \vec{x}_i) + \vec{R}_2[0, \varphi_2] \otimes (\vec{p}_g - \vec{x}_i)) \quad (1)$$

$$\vec{x}_i = \vec{x}_i + \vec{v}_i \quad (2)$$

where $\vec{R}_1[0, \varphi_1]$ and $\vec{R}_2[0, \varphi_2]$ generate two vectors containing random values uniformly distributed in the range $[0, \varphi_1]$ and $[0, \varphi_2]$ respectively. φ_1 and φ_2 are usually set to $\frac{\varphi}{2}$. φ is a positive constant. \otimes indicates point-wise vector multiplication. A constriction coefficient χ is used to restrict particles' visiting regions (commonly $\chi = 0.7298$). χ is calculated according to $\chi = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}}$, where $\varphi = \varphi_1 + \varphi_2 = 4.1$ (Clerc & Kennedy 2002).

3 PSO Niching Techniques

PSO niching methods commonly attempt to locate multiple optima to multimodal problems by partitioning a swarm into a number of sub-swarms. These sub-swarms run independently as local optimizers to find multiple optimal solutions in parallel. Currently, a variety of niching PSO algorithms have been developed. Parsopoulos and Vrahitis introduced a method to identify particles as potential solutions when the fitness value of this solution is lower than a predefined threshold value (Parsopoulos & Vrahitis 2001). Brits et al proposed NichePSO (R. Brits & van den Bergh 2002) and they extended Parsopoulos and Vrahitis' niching method by adopting multiple sub-swarms generated from a main swarm. A sub-swarm is created around a particle with little change of its fitness over a number of iterations. Then a set of produced sub-swarms do local search in parallel. Another niching PSO algorithm is species particle swarm optimizer (SPSO) proposed by Li (Li 2004). The basic idea is similar to a species conserving genetic algorithm (SCGA) in (Li et al. 2002), except that PSO is used as a local optimizer instead of using GA. In SPSO, a niche radius must be predefined. It is a typical niching parameter, used to specify the upper bound on the distance between two individuals being considered to be in the same niche (Li et al. 2002). To avoid predefining such a parameter, Li introduced an *lbest* PSO (as described in Section 2) with a ring topology (LPRT), adopting a ring topology to form niches without requiring a niche radius (Li 2010).

4 A Dynamic Archive based PSO Niching Algorithm (*rpso - sp*)

The reason existing PSO niching techniques often requiring a large population size (Li et al. 2002, Bird & Li 2006, Li 2004, 2010), for solving complex multimodal problems is that a large particle swarm is more likely to generate a great number of sub-swarms. The more sub-swarms a

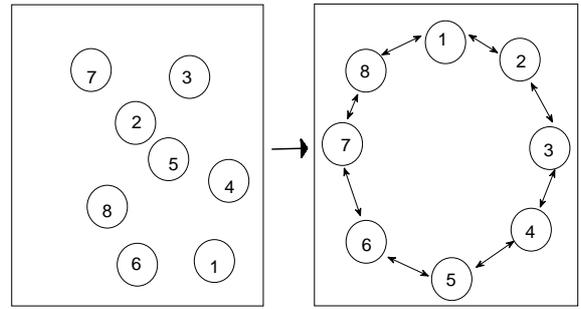


Figure 1: An example of mapping 8 particles in the swarm to a ring topology. Assuming two dimensional search space is used

niching method produces, the more local or global optima the niching method possibly can find. However, without the prior knowledge on numbers of optimal solutions to the problem, it is almost impossible to figure out the appropriate number of particles for a niching algorithm. To alleviate such a problem, a new niching PSO algorithm, *rpso - sp*, is proposed in this paper.

4.1 Constructing sub-swarms by using a ring topology

rpso - sp adopts LPRT's idea on forming sub-swarms for two reasons: simplicity and niching parameter-free (Li 2010). Existing niching algorithms commonly use a relatively complex procedure to dynamically group particles as sub-swarms (Bird & Li 2006, Schoeman & Engelbrecht 2005) based on given niching parameters and particles' current positions. Practically, many niching algorithms group particles within a threshold distance (known as a niche radius) in the search space as sub-swarms. LPRT instead maps all particles in the search space to a ring topology and Figure 1 gives such an example. Left side of the figure shows particles' actual positions in the search space (assuming two dimensional search space); the right side shows how a population of 8 particles is mapped into a ring topology. Then LPRT forms sub-swarms by grouping a fixed number of particles, which have the closest index values on a ring topology. Figure 2 and Figure 3 illustrate two variants instances of LPRT: *r3pso* and *r3pso - lhc* (Li 2010). As shown in these two Figures, both *r3pso* and *r3pso - lhc* have sub-swarms consisting of 3 particles except that the number of particles on the tail of a ring is less than 3. For instance, sub-swarm C only contains 2 particles, shown in Figure 3. As to the difference between these two variants, *r3pso* has overlapped sub-swarms which are formed by any particle and its left and right neighbors on the ring topology indicated in Figure 2. *r3pso - lhc* is the same as *r3pso*, but without overlapping neighbors shown in Figure 3. Each particle in *r3pso - lhc* only belongs to one sub-swarm. Multiple formed sub-swarms in *r3pso - lhc* do local search independently, like local hill climbers (lhc). For detailed description about LPRT, please refer to (Li 2010). LPRT's method uncovers two new features, compared with other niching algorithms. Firstly, members in each sub-swarm remain unchanged throughout a run. Secondly, particles in other parts of the search space have chances to form a sub-swarm, as LPRT creates sub-swarms regarding particles' indexes on a ring topology instead of particles' actual positions in the search space. This feature is clearly shown in Figure 1.

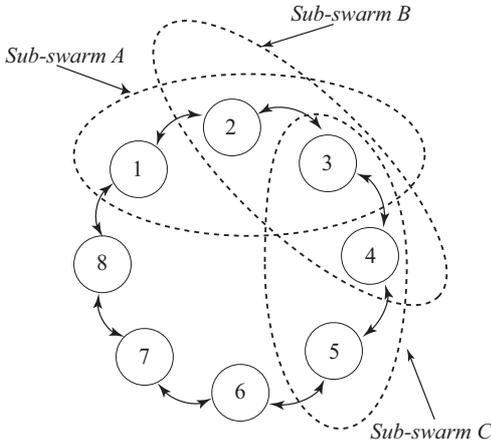


Figure 2: An example of $r3pso$. Sub-swarms A, B and C consist of 3 particles and they are overlapped.

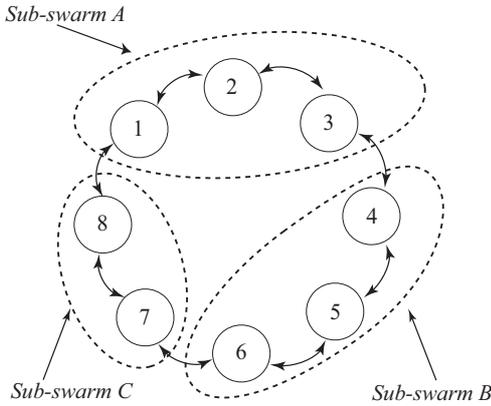


Figure 3: An example of $r3pso-lhc$. It is the same as $r3pso$, but without overlapping neighbors. All sub-swarms A, B and C do local search independently, like local hill climbers (lhc).

4.2 A Dynamic Archive Recording Found Optima

$rpso-sp$ extends LPRT by utilizing a dynamic archive (DA) to record optima found by converged sub-swarms. And these converged sub-swarms are re-randomized to search other parts of the search space, achieved through a multi-start technique. As far as the basic concept of niching is concerned, niching algorithms can be considered to be either sequential or parallel (Brits et al. 2007). If a PSO niching method repetitively applies $gbest$ PSO to the search space until all global optima are found, this algorithm is categorized as being sequential niching. In contrast, if a PSO niching method partitions the swarm into several sub-swarms for forming different niches simultaneously, this method is in the domain of parallel niching. Obviously, $rpso-sp$ presented in this paper holds characteristics from both sequential niching (employing a multi-start technique) and parallel niching (using a ring topology to form sub-swarms). So $rpso-sp$ is a hybrid of sequential and parallel niching ideas. In addition, existing parallel niching algorithms commonly record found optimal solutions in members of the population. This could possibly lead to identified solutions missing due to operations in PSO. To better maintain equally good solutions, a dynamic archive is used in $rpso-sp$ for preservation of optimal solutions. By doing so, $rpso-sp$ can lower the risk of losing found optimal solutions.

Additionally, to make the comparison between $rpso-sp$

and $LPRT$ clear in Section 6, variants of $rpso-sp$ follow the similar naming rules as $LPRT$ does (Li 2010). It is noticed that numbers in variants names indicates the number of members in each sub-swarm. Like $LPRT$, $rpso-sp$ also has several variants with varying sub-swarm sizes. $r3pso-sp$ and $r3pso-sp-lhc$ are two typical variants of $rpso-sp$. They are the same as $r3pso$ and $r3pso-lhc$ respectively, except that a dynamic archive is used in each variant for recording optima found by converged sub-swarms.

Our proposed $rpso-sp$ algorithm works generally in the following steps:

- Step 1:** Initialize particles to random positions in the search space and use a ring topology to form sub-swarms. At the same time, an empty dynamic archive DA is built.
- Step 2:** Sub-swarms run $gbest$ PSOs in parallel. The best solution found by the converged sub-swarm is checked with the DA to confirm that whether a found solution is qualified to add into the DA . Then converged sub-swarms are re-randomized to do search again. In terms of identifying converged sub-swarms, a sub-swarm is considered to be converged when the velocity of any particle in the sub-swarm approximately reduces to 0. The reason behind this is simple. Assume that a particle in a sub-swarm stops moving in the search space, this implies that no better positions can be found either by other particles in the sub-swarm or by the particle itself, according to Equation (1). Furthermore, the stopped particle almost is a leading particle with the best solution in the sub-swarm.
- Step 3:** When stopping criteria are met, $rpso-sp$ terminates.

The detailed algorithm for building a DA is presented in Algorithm 1; it is performed at each iteration step in $rpso-sp$. We assume maximization in this paper.

As can be seen in Algorithm 1, a dynamic archive S records two types of solutions p found by sub-swarms.

- Type 1:** p has a better fitness value than a threshold δ ($f(p) > \delta$). δ holds the best fitness value found by sub-swarms so far. Under this condition, p is the best solution found since a run starts. p will replace the saved solution s in the dynamic archive if p and s are considered to be similar solutions. Namely Euclidean distance between p and s is smaller than identification radius R ($\|p - s\| \leq R$; see equations (3)–(5)). If it is not the case, it means p has no similar solution in S . So p will be simply added into S . R is used to identify different solutions.
- Type 2:** p has a relatively good fitness value compared with recorded solutions in the dynamic archive $|f(p) - \delta| < \epsilon$. ϵ is set by users as acceptable accuracy, used to identify a qualified solution p . In this case, if p and s are similar $\|p - s\| \leq R$, and p has better fitness value ($f(p) > f(s)$), then p will replace s in the dynamic archive. If p has no similar solution in S , p will be directly added into S .

Equation (3) calculates the minimum distance $dist_i$ between each particle k_i and other particles k_j . K represents a set, consisting of n particles in the swarm. $\|k_i - k_j\|$ computes Euclidean distance between k_i and k_j in the search space. Then the average of minimum distances r_m in the m^{th} iteration of a run is obtained over Equation (3) and Equation (4), suggested by Bird & Li (2006). r_m in fact reflects the convergence degree of all particles in the swarm. We assume that the most of sub-swarms have

```

input :  $p$  - a potential solution obtained from converged sub-swarms
output :  $S$  - a list of found solutions in a dynamic archive

begin
   $found \leftarrow FALSE$ ;  $update \leftarrow FALSE$ ;
  if  $S = \emptyset$  then
     $S \leftarrow S \cup \{p\}$ ;  $\delta \leftarrow f(p)$ ;
  end
  else
    if  $f(p) > \delta$  then
       $\delta \leftarrow f(p)$ ;  $update \leftarrow TRUE$ ;
    end
    if  $update$  or  $|f(p) - \delta| < \epsilon$  then
      for each  $s \in S$  do
        if  $\|p - s\| \leq R$  then
          if  $f(p) > f(s)$  then
             $s \leftarrow p$ ;
             $found \leftarrow TRUE$ ;
            break;
          end
          else
             $found \leftarrow TRUE$ ;
            break;
          end
        end
      end
      if not found then
         $S \leftarrow S \cup \{p\}$ ;
      end
    end
  end
end

```

Algorithm 1: Pseudocode for building a dynamic archive.

already converged on a number of niches. Then the minimum distance between each particle and other particles, is close to zero. Accordingly, the average of minimum distances r_m is reaching zero. In short, while r_m is approaching zero, the convergence degree becomes very high.

$$dist_i = \min\{\|k_i - k_j\|; \forall k_i, k_j \in K \wedge k_i \neq k_j\} \quad (3)$$

$$r_m = \frac{\sum_{i=1}^n dist_i}{n} \quad (4)$$

$$R = \min\{r_1, r_2, \dots, r_m\} \quad (5)$$

The identification radius R employed in Algorithm 1, is used for deciding that whether a newly found solution is already in the DA . R denotes the smallest value among the average of minimum distances produced up to the m^{th} iterations of a run, shown in Equation (5). The reason why we choose R in this way is easily explained. In the first half of a run, it is highly possible that niches on the big attraction basins in multimodal problems have been identified, as big attraction basins easily absorb more particles than small attraction basins. Along with more and more niches having been found, discovering niches located on the smaller attraction basins becomes a major challenge. To handle this problem, a smaller and smaller identification radius actually is more preferable. Therefore, it makes sense to choose R in this way which can be adaptively chosen in a run.

5 Experimental setup

To measure the performance of PSO niching algorithms, a set of extensively used multimodal benchmark functions was used. Table 1 indicates the test functions used in this study ranging from simple to more challenging ones. f_1 has 5 evenly spaced global peaks. f_2 has 5 unevenly spaced global peaks. f_3 has four unevenly spaced global peaks without any local peaks. The inverted Shubert function f_4 and the inverted Vincent function f_5 are more challenging N-dimensional multimodal functions. Due to having a large number of unevenly spaced local and global optima when the function dimensionality n increases, both f_4 and f_5 are widely used in this study. For inverted Shubert 2D function, it has 18 global peaks spaced in 9 groups. The distance between global peaks in the same group is much closer than global peaks in the different groups. In general, f_4 has $n \cdot 3^n$ global peaks in 3^n groups. f_5 has 6^n unevenly spaced global peaks. f_6 only keeps a single global peak when its dimension increases. f_7 has 5^n clearly known global peaks evenly spaced in n-dimensional search space without local peaks.

We used *peak ratio* (PR) as the performance measurement throughout experiments in this study. Peak ratio defines the proportion of found peaks to the total number of known peaks. For hard test functions with a large number of global peaks, peak ratio is more informative when comparing the niching ability between different niching algorithms. To count the number of found peaks for each algorithm, an acceptance threshold γ was set to identify whether a found solution is close enough to a known optimum. If the fitness of a found solution is within γ of the desired global best fitness, this solution is considered found.

6 Experimental Results

Five sets of experiments were carried out. Firstly, We evaluated several $rpso - sp$ variants and chose a relatively good performer as the representative of $rpso - sp$. We then compared $rpso - sp$ and $LPRT$ in four aspects: basic niching ability, sensitivity to population size, niching ability by using a small population size and scalability on high dimensional multimodal functions.

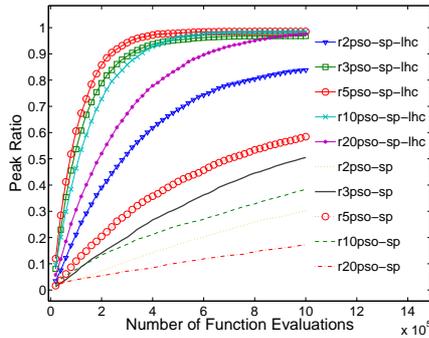
In this study, all experimental results are average values over 50 runs. MNFE and NFE indicate the maximum number of function evaluations to be allowed in a run and number of function evaluations spent during a run respectively. PS and PR denote population size and peak ratio respectively.

6.1 Choosing a $rpso - sp$ Variant comparing with $LPRT$

To make an informed decision on which variant of $rpso - sp$ should be adopted, a set of $rpso - sp$ variants were tested on the inverted Shubert 3D function $f_4(3D)$ and a relatively better performer was chosen as the base algorithm for $rpso - sp$. As can be seen in Figure 4, $r3pso - sp$ (like $r3pso$), performs much worse than the variant $r3pso - sp - lhc$ no matter how big sub-swarm sizes are. It is clear that $r3pso - sp - lhc$ performs reasonably well although it is not the best one among $rpso - sp - lhc$ variants. $r3pso - sp - lhc$ thereby is chosen as the core niching algorithm used in our experiments and it is simply denoted as $rpso - sp$. It is noticed that the performance of variants does not simply increase with increasing sizes of sub-swarms. For example, $r20pso - sp - lhc$ performs much worse than $r3pso - sp - lhc$ does. In this paper, we focus on comparison between $rpso - sp$ and two variants of $LPRT$:

Table 1: Test functions. n and GP are function dimensions and the number of global peaks respectively.

Name	Function	Range	GP
Equal Maxima (Deb 1989)	$f_1(x) = \sin^6(5\pi x)$	[0, 1]	5
Uneven Maxima (Deb 1989)	$f_2(x) = \sin^6(5\pi(x^{3/4} - 0.05))$	[0, 1]	5
Himmelblau's function (Deb 1989)	$f_3(x, y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$	[-6, 6]	4
Inverted Shubert function (Li et al. 2002)	$f_4(\vec{x}) = -\prod_{i=1}^n \sum_{j=1}^5 j \cos[(j+1)x_i + j]$	[-10, 10]	$n \cdot 3^n$
Inverted Vincent function (Li 2010)	$f_5(\vec{x}) = \frac{1}{n} \sum_{i=1}^n \sin(10 \cdot \log(x_i))$	[0.25, 10]	6^n
Inverted Rastrigin function (Li 2010)	$f_6(\vec{x}) = -\sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10)$	[-1.5, 1.5]	1
Inverted Deb's 1st function (Deb 1989)	$f_7(\vec{x}) = 1 - \frac{1}{n} \sum_{i=1}^n (1 - \sin^6(5\pi x_i))$	[0, 1]	5^n


 Figure 4: The performance of $rpso-sp$'s variants on the inverted Shubert 3D function. $\gamma = 0.1$, $PS = 100$, $MNFE = 1,000,000$.

$r3pso$ and $r3pso-lhc$, which showed the best performances among all $LPRT$ variants on a range of multimodal test functions (Li 2010).

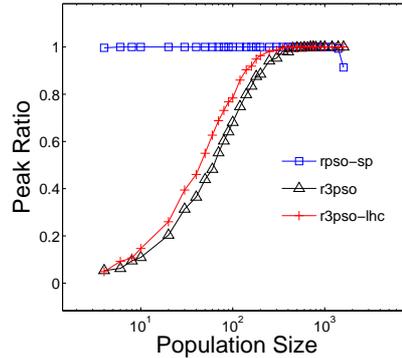
6.2 Results on One or Two Dimensional Test Functions.

Table 2 shows that $r3pso$ and $r3pso-lhc$ are able to successfully locate all global peaks on $f_1(1D)$, $f_2(1D)$ and $f_6(2D)$ within $MNFE = 200,000$, as well as $rpso-sp$. However, they perform poorly on $f_4(2D)$ and $f_7(2D)$. Peak ratios for $r3pso$ and $r3pso-lhc$ on $f_7(2D)$ are only 46% and 49% respectively. Meanwhile, both $r3pso$ and $r3pso-lhc$ fail to find all global peaks on $f_3(2D)$ in several runs. In contrast, $rpso-sp$ performs consistently well on these nine benchmark functions and it can locate all global peaks within $MNFE = 200,000$.

6.3 The Effect of varying Population Sizes

To explore the effect of population size on the performance of niching algorithms, population sizes ranging from 4 to 1600 were used for $rpso-sp$, $r3pso$ and $r3pso-lhc$, on optimizing $f_4(2D)$ and $f_5(2D)$. Figure 5 and Figure 6 show $r3pso$ and $r3pso-lhc$ have a good niching ability only when population size is very large ($PS > 250$ on $f_4(2D)$ and $PS > 1200$ on $f_5(2D)$). When smaller population sizes are used, PR drops dramatically for both variants. For example, when $PS < 10$, $PR < 0.2$ is for both variants on $f_4(2D)$. In contrast, $rpso-sp$ performs consistently well for different population sizes. Even when $PS < 10$, $rpso-sp$ is still able to find almost 18 optima on $f_4(2D)$.

To sum up, $rpso-sp$ shows a much lower sensitivity to population size than both $r3pso$ and $r3pso-lhc$. $rpso-sp$ performs consistently well when varying the population size. On the other hand, the performance of


 Figure 5: Peak ratios for Inverted Shubert 2D Function as PS increases from 4 to 1600. $\gamma = 0.0001$, $MNFE = 200,000$

$r3pso$ and $r3pso-lhc$ are largely dependent on population sizes. Without a large enough PS , these algorithms may fail to achieve desired results.

6.4 The effect of increasing number of evaluations

Since the goal of a niching algorithm is to find and maintain niches on all global optima, we designed a set of experiments to monitor peak ratios in the whole run for each algorithm by using a relatively small population size: $PS = 50$. Figure 7 and Figure 8 demonstrate the niching behaviors for each algorithm. Although $rpso-sp$ is unable to find all global optima on $f_4(3D)$ and $f_5(3D)$ in most cases, it performs far better than $r3pso$ and $r3pso-lhc$. Furthermore, Figure 8 displays that $rpso-sp$ has a potential to obtain higher peak ratios if more evaluations are given.

As expected, one of advantages $rpso-sp$ has is that it is able to remember all found optima. In other words, the issue of maintaining niches in niching algorithms is handled by employing a dynamic archive to save found optima. More importantly, a population size is no longer a crucial parameter for $rpso-sp$. Since it is able to find more optima continuously, without depending on the specified population size. Finally, compared with $r3pso-lhc$ and $r3pso$, $rpso-sp$ can reach higher PR within a small number of NFE as shown in both Figure 7 and Figure 8.

6.5 The effect of increasing dimensionalities

To examine the scalability of niching PSO variants, a set of experiments were conducted on f_4 and f_5 by increasing the dimension from 2 to 5, as shown in Figure 9 and Figure 10. $rpso-sp$ still performs better than other niching variants, although the performance of all three variants

Table 2: Peak ratios on low dimensional functions for $r3pso$, $r3pso - lhc$ and $rpso - sp$. $\gamma = 0.0001$, $PS = 50$, $MNFE = 200,000$.

	$f_1(1D)$	$f_2(1D)$	$f_3(2D)$	$f_4(2D)$	$f_5(1D)$	$f_6(2D)$	$f_7(2D)$
$r3pso$	100%	100%	85%	51%	78%	100%	46%
$r3pso - lhc$	100%	100%	98%	54%	90%	100%	49%
$rpso - sp$	100%	100%	100%	100%	100%	100%	100%

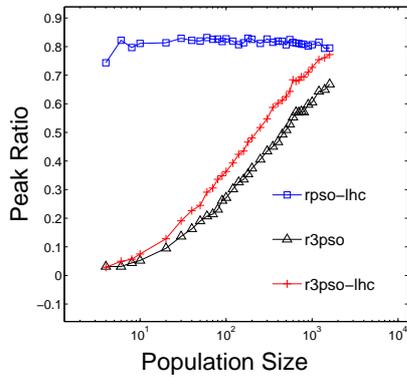


Figure 6: Peak ratios for Inverted Vincent 2D Function as PS increases from 4 to 1600. $\gamma = 0.0001$, $MNFE = 500,000$

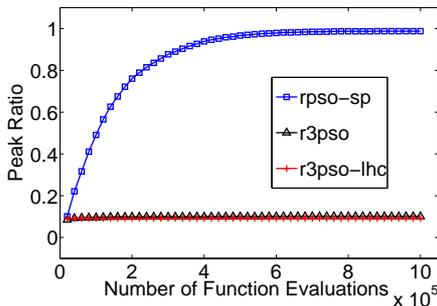


Figure 7: Peak ratios for Inverted Shubert 3D as the number of function evaluations increases. $MNFE = 1,000,000$, $PS = 50$, $\gamma = 0.01$.

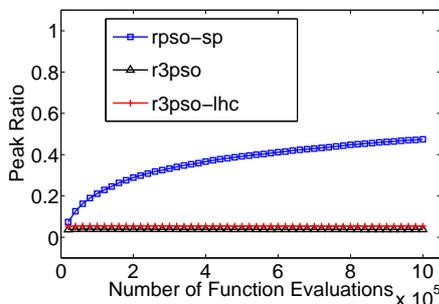


Figure 8: Peak ratios for Inverted Vincent 3D as the number of function evaluations increases. $MNFE = 1,000,000$, $PS = 50$, $\gamma = 0.0001$.

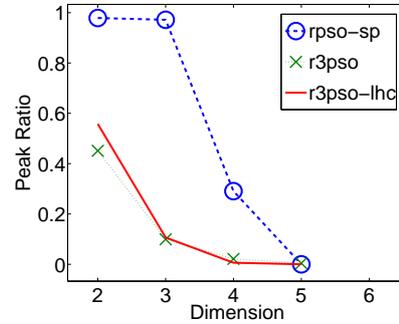


Figure 9: Peak ratios for Inverted Shubert Function (f_4) as the dimensionality increases from 2 to 5. $\gamma = 0.1$, $MNFE = 1,000,000$, $PS = 50$.

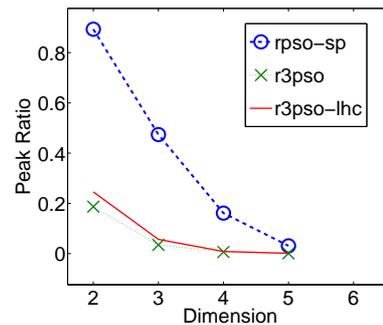


Figure 10: Peak ratios for Inverted Vincent Function (f_5) as the dimensionality increases from 2 to 5. $\gamma = 0.0001$, $MNFE = 1,000,000$, $PS = 50$.

degrade quickly with increasing dimensions. It is noticeable that $rpso - sp$ managed to keep a peak ratio ≈ 0.2 on both $f_4(4D)$ and $f_5(4D)$. Accordingly, the absolute number of optima found is around 94 and 210 on $f_4(4D)$ and $f_5(4D)$ respectively. This implies that $rpso - sp$ still maintain a relatively better scalability to higher dimensional problems even when the population size is much smaller than the number of known global peaks on benchmark functions. However, all PSO niching variants reach much lower peak ratios when $dimension = 5$.

7 Conclusions

The primary advantage of $rpso - sp$ over previous niching techniques is that it does not depend on a large population size to optimize multimodal functions with a large number of global optima. Our experiments clearly demonstrate that $rpso - sp$ is able to provide competitive performance even when only small population sizes have been used. Although the performance of $rpso - sp$ drops quickly as the function dimensionality increases, it per-

forms consistently well or better than the ring topology based PSO niching algorithms without using a dynamic archive. Since a user does not have to specify niching parameters and only a small population size is required, *rpso - sp* holds a great promise in real world problem solving. Future work will investigate how to further enhance the scalability of *rpso - sp* and apply *rpso - sp* to real-world problems solving.

References

- Back, T., Fogel, D. B. & Michalewicz, Z., eds (1997), *Handbook of Evolutionary Computation*, IOP Publishing Ltd., Bristol, UK, UK.
- Bird, S. & Li, X. (2006), Adaptively choosing niching parameters in a PSO, in M. Cattolico, ed., 'Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, July 8-12, 2006', ACM, pp. 3-10.
- Brits, R., Engelbrecht, A. & van den Bergh, F. (2007), 'Locating multiple optima using particle swarm optimization', *Applied Mathematics and Computation* **189**(2), 1859 - 1883.
- Clerc, M. & Kennedy, J. (2002), 'The particle swarm - explosion, stability, and convergence in a multidimensional complex space', *IEEE Trans. on Evol. Comput.* **6**, 58-73.
- Deb, K. (1989), Genetic Algorithms in multimodal function optimization (Master thesis and TCGA Report No. 89002), PhD thesis, Tuscaloosa: University of Alabama, The Clearinghouse for Genetic Algorithms.
- Kennedy, J. & Eberhart, R. (1995), Particle swarm optimization, in 'Proc. Conf. IEEE Int Neural Networks', Vol. 4, pp. 1942-1948.
- Li, J.-P., Balazs, M. E., Parks, G. T. & Clarkson, P. J. (2002), 'A species conserving genetic algorithm for multimodal function optimization', *Evol. Comput.* **10**(3), 207-234.
- Li, X. (2004), Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization, in K. Deb, ed., 'Proc. of Genetic and Evolutionary Computation Conference 2004(LNCS 3102)', pp. 105-116.
- Li, X. (2010), 'Niching without niching parameters: Particle swarm optimization using a ring topology', *Evolutionary Computation, IEEE Transactions on* **14**(1), 150-169.
- Mahfoud, S. W. (1995), Niching methods for genetic algorithms, PhD thesis, Urbana, IL, USA.
- Parsopoulos, K. & Vrahatis, M. (2001), Modification of the particle swarm optimizer for locating all the global minima, in R. N. M. K. V. Kurkova, N. Steele, ed., 'Artificial Neural Networks and Genetic Algorithms', Springer, pp. 324-327.
- Price, K., Storn, R. M. & Lampinen, J. A. (2005), *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA.
- R. Brits, A. E. & van den Bergh, F. (2002), A niching particle swarm optimizer, in 'Proc. of the 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002(SEAL 2002)', pp. 692-696.
- Schoeman, I. & Engelbrecht, A. (2005), 'A parallel vector-based particle swarm optimizer', pp. 268-271.

Optimized Relative Lempel-Ziv Compression of Genomes

Shanika Kuruppu¹Simon J. Puglisi²Justin Zobel¹

¹ NICTA VRL, Department of Computer Science and Software Engineering
The University of Melbourne, Parkville, Victoria 3010
Email: {kuruppu, jz}@csse.unimelb.edu.au

² School of Computer Science and Information Technology
RMIT University, Melbourne, Victoria 3001
Email: simon.puglisi@rmit.edu.au

Abstract

High-throughput sequencing technologies make it possible to rapidly acquire large numbers of individual genomes, which, for a given organism, vary only slightly from one to another. Such repetitive and large sequence collections are a unique challenge for compression. In previous work we described the RLZ algorithm, which greedily parses each genome into factors, represented as position and length pairs, which identify the corresponding material in a reference genome. RLZ provides effective compression in a single pass over the collection, and the final compressed representation allows rapid random access to arbitrary substrings. In this paper we explore several improvements to the RLZ algorithm. We find that simple non-greedy parsings can significantly improve compression performance and discover a strong correlation between the starting positions of long factors and their positions in the reference. This property is computationally inexpensive to detect and can be exploited to improve compression by nearly 50% compared to the original RLZ encoding, while simultaneously providing faster decompression.

Keywords: DNA, Compression, Lempel-Ziv, LZ77, Suffix Array, Subsequence matching

1 Introduction

Genetic sequencing of many individuals from the same species is vital for a better understanding of variation between individuals and will ultimately lead to improved medical treatments and evolutionary understanding. With the recent advent of high-throughput sequencing technology, large scale acquisition of genomes is becoming a common exercise. Projects such as the 1000 Genomes project,¹ the proposed Genome 10K vertebrate genome project (Hausler et al. 2009), and many other similar activities are leading to collections of large quantities of highly re-

This work was supported by the Australian Research Council and by the NICTA Victorian Research Laboratory. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Center of Excellence program.

Copyright ©2011, Australian Computer Society, Inc. This paper appeared at the 34th Australasian Computer Science Conference (ACSC 2011), Perth, Australia, January 2011. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 113, Mark Reynolds, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

¹<http://www.1000genomes.org/page.php>

dundant DNA sequences, creating challenges for efficient storage and retrieval of these sequences.

In previous work, we proposed RLZ (Kuruppu et al. 2010), an algorithm that compresses a collection of genomes or sequences from the same species with respect to the reference sequence for that species using a simple greedy technique, akin to LZ77 parsing (Ziv & Lempel 1977). RLZ parses a genome into factors, or substrings, each of which occurs in the reference sequence. Having parsed a genome up to position i , the next factor is the longest match starting at i , which occurs anywhere in the reference. (We delay a precise description of RLZ until Section 3.) Compared to other DNA compression methods, RLZ is fast, and is able to achieve good compression; only *XMcompress* (Cao et al. 2007) proved significantly more effective, and was orders of magnitude slower.

While the greedy approach of taking the longest match at each point in the parsing is fast and effective, in this paper we show that significantly better compression performance can be obtained by simple non-greedy techniques, and by a more judicious treatment of the factors produced by the parsing. The refined approach is also intended to better handle increasing genetic distance between organisms, by more efficient handling of short factors, which arise in the presence of numerous mutations. As our results show, for our application area – compression of the DNA of related organisms – the size is nearly halved, equalling or bettering all previous methods.

We first review related work on DNA compression. Section 3 details relative Lempel-Ziv parsing, and the basic (greedy) RLZ approach for compressing a collection of related genomes. Then, in Section 4, we describe the integration of non-greedy parsing with RLZ, including the encoding of small factors and an efficient parsing algorithm to keep compression speed fast. Section 5 explores a phenomenon manifest by the RLZ algorithm when compressing a pair of related genomes: a strong correlation between a factor's start point and its position of occurrence in the reference. We show how this phenomenon can be exploited and combined with non-greedy parsing to simultaneously attain handsome improvements in both compression performance and decompression speed. Section 6 details the results of experiments with the new approach, before conclusions and reflections are offered in Section 7.

2 Compression of DNA

DNA sequence compression was introduced by Grumbach and Tahi with the *BioCompress* algorithm (Grumbach & Tahi 1993). While *BioCompress* fol-

lows the principle ideas of LZ77 (Ziv & Lempel 1977), the algorithm is made DNA-specific by simple modifications such as encoding nucleotides with 2 bits per base and detecting reverse complement repeats. BioCompress showed that DNA-specific compression algorithms could outperform general-purpose compression algorithms. Two further variations on the *BioCompress* theme are *Cfact* (Rivals et al. 1996) and *Off-line* (Apostolico & Lonardi 2000).

The above algorithms and their variants produce promising compression results for DNA sequences using exact repeat detection. However, *GenCompress* (Chen et al. 2000) showed that by considering approximate repeats, these results can be improved. Due to SNPs (single nucleotide polymorphisms) and mutations that occur over time, repeated regions between DNA sequences are not necessarily exact, even for sequences from the same species. Since *GenCompress*, most DNA compression algorithms, including *CTW+LZ* (Matsumoto et al. 2000), *DNACompress* (Chen et al. 2002), *DNAPack* (Behzadi & Fessant 2005), *GeNML* (Korodi & Tabus 2007), and *XMcompress* (Cao et al. 2007), have been based on efficient methods of approximate repeat detection.

Early DNA compression algorithms were only able to compress very small files. The first compressor for a large genome was the statistical algorithm *NML* (Korodi & Tabus 2005) and its successor *GeNML* (Korodi & Tabus 2007). These algorithms divide the input sequences into blocks and compress each block using a *Maximum Likelihood Model*. *XMcompress* (Cao et al. 2007) is another statistical compression algorithm, which uses “experts” to determine the probability distribution of each symbol, which in turn is used to encode the symbol. The algorithm is able to compress a human genome using less resources than *GeNML* while producing better compression results, and is the best single sequence compression algorithm we are aware of.

Two recent algorithms have focused on compressing large datasets of DNA sequences from the same species. Christley et al. (2009) uses various encoding techniques to compress variation data (mutations and indels) of human genomes. The compression is relative to the human reference sequence and known variations in a SNP database. While the method produces excellent compression results for human variation data, a large amount of overhead is required to store the reference genome and SNP database. The method does not support compression of assembled genomes nor random access into the compressed data. A similar approach is taken by Brandon et al. (2009).

Mäkinen et al. (2010) proposed algorithms that not only compress sets of related genomes, but also support the following retrieval functionality: *display(i, j, k)*, which returns the substring from position j to k in sequence i ; *count(p)*, which returns the number of occurrences of substring p ; and *locate(p)*, which returns the positions where p substring occurs in the collection. This family of *self-indexing* techniques achieves good compression results, as well as good search times and is the inspiration behind the RLZ algorithm. Our previous work (Kuruppu et al. 2010) shows that RLZ provides better compression and much faster random access times than the data structures of Mäkinen et al. As it is the basis for our present work, we describe RLZ in finer detail in the next section.

3 The RLZ Algorithm

The input to the RLZ algorithm is a set of sequences, as follows:

Definition 1. Let \mathcal{C} be a collection of r sequences. Each sequence $T^k \in \mathcal{C}$ has length n , where $1 \leq k \leq r$ and $N = \sum_{k=1}^r |T^k|$.

The sequence T^1 is called the reference sequence. The RLZ algorithm takes each other sequence in the collection and encodes it as a series of *factors* (substrings) that occur in the reference sequence. The manner in which each sequence is broken into factors is similar to the famous LZ77 parsing (Ziv & Lempel 1977). We now give a precise definition.

Given two strings T and S , the Lempel-Ziv factorisation (or parsing) of T relative to S , denoted $LZ(T|S)$, is a factorisation $T = w_0 w_1 w_2 \dots w_z$ where w_0 is the empty string and for $i > 0$ each factor (string) w_i is either:

- (a) a letter which does not occur in S ; or otherwise
- (b) the longest prefix of $T[|w_0 \dots w_{i-1}| \dots |T|]$ which occurs as a substring of S .

For example, if $S = tcttctct$ and $T = ttctgttc$ then in $LZ(T|S)$ we have $w_1 = ttct$, $w_2 = g$ and $w_3 = ttc$. For the purposes of compression, factors are specified not as strings, but as (p_i, ℓ_i) pairs. Each p_i denotes the starting position in S of an occurrence of factor w_i (or a letter if w_i is generated by rule (a)) and ℓ_i denotes the length of the factor (or is zero if w_i is generated by rule (a)). Thus, in our example above, we have:

$$LZ(T|S) = (2, 4)(g, 0)(2, 3).$$

For technical convenience, for the remainder of this paper we assume that no factors are generated by rule (a) above; that is, if a symbol c occurs in T then c also occurs in S . For DNA strings, this is not an unreasonable assumption, but for other types of data (even proteins) it may be flawed. However, if S is not so composed we can simply add the at most $\sigma - 1$ missing symbols at the end.

We now define the RLZ encoding of the collection.

Definition 2 (RLZ). Let T^1 be the reference sequence. Each sequence, T^i for $2 \leq i \leq r$ is represented with respect to T^1 as

$$LZ(T^i|T^1) = (p_1, \ell_1), (p_2, \ell_2), \dots, (p_{z_i}, \ell_{z_i}),$$

resulting in z factors in total where $z = \sum_{i=2}^r z_i$.

Using this representation, the collection \mathcal{C} can be stored in at most $n \log \sigma + z \log n + z \log \frac{N}{z} + O(z)$ bits. In this bound, the $n \log \sigma$ term is for the reference sequence, which is stored uncompressed; the $z \log n$ term is for the p_i components of each factor, which are pointers into the reference; and the $z \log \frac{N}{z}$ term is for (essentially) the Golomb encoding of the ℓ_i components.

As a baseline in this paper we assume that RLZ encodes each p_i component using $\log n$ bits and a Golomb code ($M = 64$) for each ℓ_i . We refer to this encoding method, combined with greedy parsing, as *standard RLZ*. In this paper we concentrate only on raw compression and not on random access. However, we make some comments on how random access can be achieved in the final section.

4 Non-greedy parsing

In the standard relative LZ parsing algorithm described above, we take, at each point in the parse, the factor which has the longest match in the reference sequence. This is a greedy decision, which does not account for the possibility that a smaller overall encoding may be achieved by taking a shorter match (leading to a different parsing). This section explores such a *non-greedy* approach.

For a non-greedy strategy to be effective, the (p_i, ℓ_i) factor pairs must be coded with a variable-length code: it is well known (Ferragina et al. 2009, Horspool 1995) that if a fixed number of bits is used to code each pair, then the greedy method does as well as any other. The particular variability we focus on is the encoding of *short factors*, which, given the small DNA alphabet, can be encoded much more succinctly as literal strings than as (p_i, ℓ_i) pairs.

Lookahead by h . The non-greedy method we use is a generalization of that used in the **gzip** compressor, as investigated by Horspool (1995). The idea is as follows. Assume the parsing is up to position i . If the lookahead limit $h = 1$ then, instead of encoding the longest match, (p, ℓ) , starting at i , the longest match starting at $i + 1$, (p', ℓ') is also considered. If $\ell' > \ell$ then the letter at position i is encoded as a single literal followed by the factor (p', ℓ') . This idea can be generalized so that the lookahead can be extended up to position $i + h$, instead of just $i + 1$. Horspool explored this approach on English text, testing the gains possible by looking ahead up to $h = 7$ letters. He found that, on English at least, the LZ77 algorithm improved when looking ahead up to 5 to 6 letters, after which no further gains were made.

Longest factor in a region. A slight variation to the simple lookahead by h algorithm is to continue the lookahead until the longest factor within a region is found, and encode the section before the longest factor as a series of shorter factors followed by the longest factor. The longest factor within a region is defined as a factor of position p' and length l' , starting from position i' in the current sequence, where no other factor in between positions i and i' has a length $\geq l'$, and no other factor in between positions i' until $i' + l'$ has a length $\geq l'$.

The algorithm operates in a very similar manner to that of the *lookahead by h* algorithm. The main difference is that instead of the lookahead amount being a constant h , the amount varies depending on the longest factor found so far. First, the longest factor at position i is found of position p and length l . To ensure that this is indeed the longest factor, we set $h = l$ and lookahead by up to l positions to see if there is a longer factor. If there is a longer factor at i' with position p' and length l' , then we now set $h = l'$ and continue to see if there's a factor that has a length above l' . This process continues until no factor that is longer than the current longest factor can be found. In the meantime, each longest factor found so far is kept in an array so that when the actual longest factor is found, the series of shorter factors leading up to the actual longest factor can also be encoded. This algorithm essentially finds the local maximum in terms of the factor length.

Efficient non-greedy parsing algorithm. We have described an efficient algorithm for *greedy* RLZ

parsing which runs in $O(N)$ time using the suffix tree of T^1 or in $O(N \log n)$ time using the (more compact) suffix array data structure (Kuruppu et al. 2010). It is possible to implement the non-greedy approaches in the same time bounds. In fact, it is possible to compute the factor information for *every* position in the input in $O(N)$ time using a suffix tree. We apply an algorithm for computing so-called *matching statistics*, due to Chang & Lawler (1994) (see also Gusfield (1997), Abouelhoda et al. (2004), and Maaß (2006)). Formally, the matching statistics of string T w.r.t. string S is a table of pairs (p_j, ℓ_j) , where $0 \leq j < |T|$ such that:

- (a) $T[j..j + \ell_j]$ is the longest prefix of $T[j..|T|]$ which occurs as a substring of S , and
- (b) $T[j..j + \ell_j] = S[p_j..p_j + \ell_j]$.

There may be more than one such p_j , and it does not matter which is chosen. Continuing our earlier example, for strings $S = tcttctct$ and $T = ttctgttc$ the matching statistics of T w.r.t. S are given in the following table:

j	0	1	2	3	4	5	6	7
(p_j, ℓ_j)	(2,4)	(0,3)	(1,2)	(0,1)	(0,0)	(2,3)	(0,2)	(1,1)

At position $j = 0$ in T , the longest prefix of $T[0..|T|]$ matching a substring in S is the substring $S[2..6] = ttct$ of length 4 so it is encoded as (2, 4). At position $j = 1$ in T , the longest prefix of $T[1..|T|]$ matching a substring S is the substring $S[0..3] = tct$ of length 3 which is encoded as (0, 3). A special case occurs at position 4 where no prefix of $T[4..|T|] = gttc$ occurs in S . Therefore, we encode it using the start position 0 and a length of 0 to indicate that there was no match.

Clearly any LZ parsing (greedy or non-greedy) of T relative to S can be derived from a subsequence of the matching statistics of T w.r.t. S . To our knowledge, the link between relative Lempel-Ziv parsing and matching statistics computation has never been made in the literature — the two methods appear to have been studied in quite different contexts (classification and approximate pattern matching respectively).

Short factor encoding. The non-greedy parsing algorithm described above is likely to create short factors which can be encoded in a more efficient manner than as position and length pairs. For example, if a factor represents a single nucleotide, then it is cheaper to encode it using 2 bits per nucleotide (spelling out the literal A , C , G or T symbol) rather than using the standard RLZ encoding of $\log n + \ell \bmod M + \log M + 1$ bits, where M is the divisor of the Golomb code. With this in mind, we define a *short factor* as any factor which has a literal encoding smaller than its position and length pair encoding.

To encode short factors, first we use a 0 bit to indicate that a short factor is about to be encoded followed by the Golomb encoded length of the short factor (in bases). Then we simply encode each nucleotide from the standard DNA alphabet as 2 bits per base. Any non-short factor is prefixed with a 1 bit to indicate to the decompressor to decode it as position and length pair. We use $M = 8$ for Golomb encoding simply because the datasets used for our experiments show that most short factors have an average length of 12. However, this can be adjusted depending on dataset properties. Finally, we remark

that the utility of an efficient short-factor encoding is not limited to non-greedy parsing: short factors are also produced during greedy parsing.

5 Reducing space for position values

One unfortunate aspect of RLZ is the large size of the position components. Because matches are allowed anywhere in the reference sequence, each p_i value can range between 0 and $n - 1$ and so $\log n$ bits of storage for each seem to be necessary. That is, although RLZ was developed as a method for coding one sequence relative to another similar sequence, in practice we allowed each factor to come from any location in the reference, so the ordering of factors was not leveraged at encoding time. We now describe a way to reduce the space of the p_i values, exploiting a property particular to the problem of compressing related genomes.

While inspecting the RLZ factors we noticed the factor sequence consisted, with few exceptions, of alternating short and long factors. Moreover, the p component of the i th long factor always seemed to be less than the p component of the $(i + 1)$ th long factor, forming a long subsequence of increasing p components in the factor sequence. An example of this behaviour for the *S. cerevisiae* genome 273614N is as follows:

```

...
10030697 10
* 16287 23
10086342 13
8689589 13
* 16336 48
3831041 11
* 16395 28
9166835 12
11588317 13
* 16448 84
787019 13
...

```

In this table, the left-hand values are the position in the reference sequence and the right-hand values are the factor length. The factors marked with * are long factors (relative to the remaining shorter factors) and their position (left-hand) values form an increasing subsequence. On closer inspection, we found that the *longest increasing subsequence* (LISS) was comprised of roughly half the factors, and these factors tended to be long, as Figure 1 illustrates this distribution. These factors can be identified in $O(z \log z)$ time using the algorithm by Schensted (1961). From here on we identify the factors participating in the LISS as *LISS factors* and the remaining factors as *interleaving factors*.

Such long LISSs are present in the RLZ factors because of the close genetic relationship of the sequence being compressed to the reference sequence. The dataset contains sequences that are evolutionarily related to the reference sequence but with mutations, insertions, deletions and rearrangements scattered across the sequence. When a sequence is factorised relative to the reference, similarity is captured by very long factors that form an alignment between the two sequences. The shorter factors in between correspond to areas of genetic mutation and rearrangement, which characterize the difference between individuals of the same species. Indeed, one of

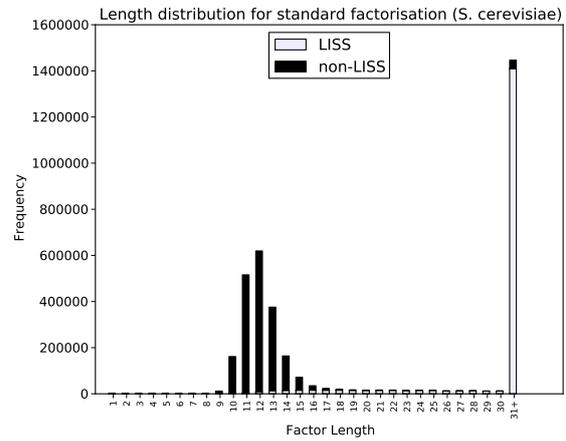


Figure 1: The factor length distribution for *S. cerevisiae* when the standard factorisation algorithm is used.

the most common mutations is replacement of a single nucleotide, thus converting, for example, one long factor into two shorter factors separated by a single base.

The presence of such LISS in the RLZ factors suggests position components (of factors that participate in the LISS) can be encoded more efficiently than $\log n$ bits each, by encoding differences, rather than absolute positions. More precisely, we store a bit vector of z bits total, where the i th bit is set to 1 if and only if the factor is an LISS factor and is 0 otherwise. Each interleaving factor is encoded in the standard manner. The first LISS factor is also encoded in the standard way, using $\log n$ bits per position and Golomb encoding the length. The remaining LISS factor positions are encoded by Golomb encoding the difference between the current position and the previous LISS factor position, and the lengths are Golomb encoded.

We found encoding differences between LISS factor positions as above led to a significant boost in compression performance. However, a further saving is possible. Let factor (p_i, l_i) be a LISS factor, (p_{i+1}, l_{i+1}) be an interleaving factor, and (p_{i+2}, l_{i+2}) be another LISS factor. Then, given the likelihood that the interleaving factors encode mutations, factor $i + 2$ will begin at or close to $p_i + l_i + l_{i+1}$. In other words, after encoding a mutation, the algorithm will go back to detecting factors from the position where the reference and current sequence align. This position can be predicted from the last position where the two sequences aligned and the length of the mutation. Using this observation, it is unnecessary to encode differences since the positions can be well predicted from the cumulative length of the last LISS factor and the lengths of the interleaving factors.

With this in mind, we use the following technique to encode the positions for LISS factors. The first LISS factor is still encoded in the same manner as an interleaving factor. Any subsequent LISS factor position is encoded by first checking if the factor position is exactly at the predicted position using the previous LISS factor position and the cumulative length since the previous LISS factor position. If this condition is satisfied then a 0 bit is output. Otherwise, a 1 bit is output and a further 0 or 1 bit is output depending on if the actual position is to the left or to the right of the predicted position, respectively. Then the difference between the actual and expected position is Golomb encoded.

6 Results

We use three datasets to analyse the compression performance of the new techniques. The first two datasets are two different species of yeast; *S. cerevisiae* with 39 genomes and *S. paradoxus* with 36 genomes.² The third dataset is 33 strains of *E. coli* sequences.³

Tests were conducted on a 2.6 GHz Dual-Core AMD Opteron CPU with 32Gb RAM and 512K cache running Ubuntu 8.04 OS. The compiler was GCC v4.2.4 with the `-O9` option.

First, we discuss the compression performance for the various combinations of factorisation and encoding algorithm. There are four main combinations:

- Lookahead factorisation (`lookahead`)
- Lookahead factorisation with short-factor encoding (`lookahead+shortfac`)
- Lookahead factorisation with LISS encoding (`lookahead+liss`)
- Lookahead factorisation with LISS encoding and short-factor encoding (`lookahead+liss+shortfac`)

For the lookahead factorisation algorithm, we experiment with lookahead limits ranging from 0 to 30 followed by the longest factorisation algorithm, described in Section 4. A lookahead limit of zero equates to the standard greedy factorisation algorithm.

Figure 2 shows the compression performance (in Mbytes) for the various algorithmic combinations. The baseline standard algorithm is indicated with a horizontal dashed line. Unsurprisingly, for all datasets, using lookahead with the standard (p, ℓ) factor encoding leads to worse compression. Using the `lookahead+shortfac` combination encodes short factors efficiently and improves compression. The `lookahead+liss` combination also reduces the compressed size for the yeast datasets but not for the *E. coli* dataset. While LISS encoding is sometimes able to reduce compressed size, it does not encode shorter factors efficiently. For all three datasets, the combination of lookahead factorisation along with LISS encoding and short-factor encoding `lookahead+liss+shortfac`, provides clearly superior compression to the other methods.

To further analyse the LISS encoding, we use the factor length distribution for the *S. cerevisiae* dataset. Figure 1 shows the length distribution of factors when the standard factorisation algorithm is used. Each bar shows the split between LISS and non-LISS factors for a given factor length. Most short factors have a length ranging from 10–15 while the majority of LISS factors have length at least 31. This is why, even when using the standard greedy factorisation, it is beneficial to use specialized short-factor encoding.

Figure 3 shows the length distribution of factors when the longest factorisation algorithm is used. There is a remarkable difference between this distribution and that in Figure 1 (produced by the greedy parsing). When a lookahead factorisation algorithm is used, a large proportion of the interleaving (non-LISS) factors end up having a length of one. An example of this behaviour for the *S. cerevisiae* genome 2736147N is as follows:

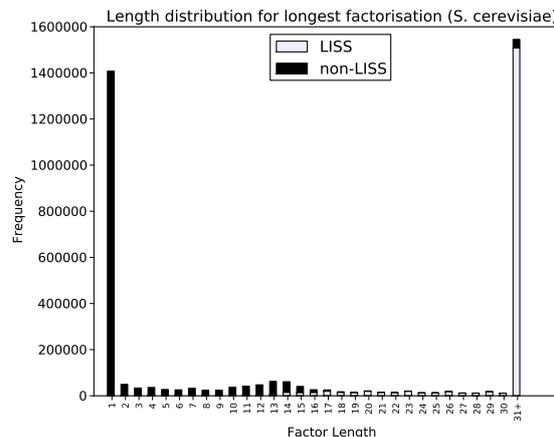


Figure 3: The factor length distribution for *S. cerevisiae* when the longest factorisation algorithm is used.

```

...
* 25359 203
2927486 1
* 25563 58
2533639 1
* 25622 97
4585768 1
* 25720 156
11302741 1
* 25877 230
...
    
```

A LISS factor is frequently followed by a non-LISS factor of length one. We hypothesise that these single-symbol factors are single nucleotide polymorphisms (SNPs), or point mutations. When the longest factor at a given position is found and it is an alignment to the reference sequence, then the alignment stops when a nucleotide is reached that is not shared between the sequence and its reference. Without the lookahead algorithm, a relatively short factor is found from the next position to be encoded and then the aligning continues. With the lookahead algorithm, by looking ahead by just one position, a single point mutation is skipped and the alignment to the reference can continue just by encoding the mutation as a factor of length one. We plan to more closely analyse this SNP hypothesis in the future.

Figure 4 and Figure 5 show, respectively, compression and decompression⁴ times for a range of lookahead limits. Both LISS encoding and short-factor encoding was enabled for this experiment. In general, compression is faster when lookahead is used. This is explained by the use of the efficient parsing algorithm described in Section 4. The longest factor can be found in $O(N)$ time (or $O(N \log n)$ time using a suffix array) on average, and a longer region of the sequence is covered each time a longer factor is found by looking ahead. However, when the longest factorisation algorithm is used, the compression is slower. In order to find the longest factor within a region, a lot more comparisons are required as the lookahead may happen up to hundreds to thousands of positions.

Decompression time is more variable but always

⁴The compression time only includes the time to generate and encode the factors and does not include the time required to generate the suffix array and its other associated data structures, nor the time to compress the reference sequence. These times are included in the results presented later in Table 1. Similarly, the decompression time only includes the time to decode the factors and does not include the time taken to decode the reference sequence. These times are also included in Table 1.

²<ftp://ftp.sanger.ac.uk/pub/dmc/yeast/latest>

³<ftp://ftp.ensemblgenomes.org/pub/bacteria/release-5/fasta/>

Figure 2: The variation in Compressed size (in Mbytes) of the *S. cerevisiae*, *S. paradoxus*, and *E. coli* datasets for changes in the lookahead limit, using the four combinations of encoding techniques.

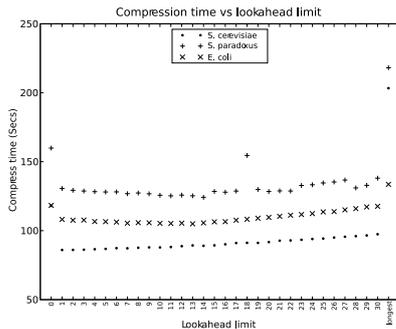
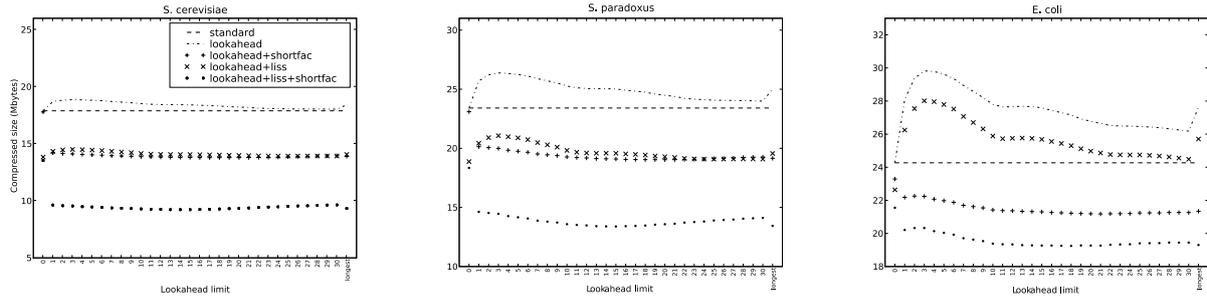


Figure 4: The variation in the time taken to compress the data (in seconds) when the lookahead limit is varied for *S. cerevisiae*, *S. paradoxus* and *E. coli* datasets. The time only includes the time taken to discover the factors and encode them.

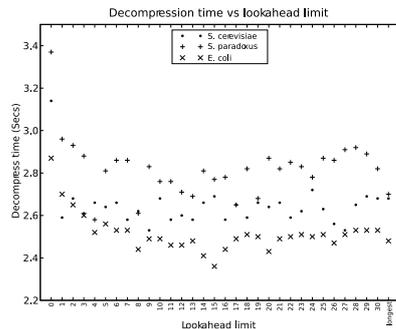


Figure 5: The variation in the time taken to decompress the data (in seconds) when the lookahead limit is varied for *S. cerevisiae*, *S. paradoxus* and *E. coli* datasets. The time only includes the time taken to decode the factors.

faster when some amount of lookahead is performed. The main reason for this is reduced cache misses. The most expensive operation during decompression is the access to the reference sequence that is required to copy the symbols for (that is, decode) each factor. With lookahead, we increase the chance that a factor will be considered short, and encoded as a literal — decoding literals requires no access to the reference. Moreover, lookahead produces longer non-short factors, which increase the number of symbols decoded per access to the reference, and also reduces cache misses.

Figure 6 illustrates the tradeoff between space and throughput achieved by the variants. The *lookahead+liss+shortfac* combination provides the best results across all datasets tested.

Finally, we directly compare the results of the standard RLZ algorithm (RLZ-std) to the results of the optimized RLZ algorithm (RLZ-opt) (*lookahead+liss+shortfac*) in Table 1. For this,

we used the two yeast datasets and a dataset of four human genomes, consisting of the reference human genome⁵, the Craig Venter genome⁶, the Korean genome⁷ and the Chinese genome⁸. We also compare the optimized RLZ algorithm to three other DNA compression algorithms. COMRAD (Kuruppu et al. 2009) is a dictionary compression algorithm that is also suitable for compressing datasets of DNA sequences from the same or similar species. RLCSA (Mäkinen et al. 2010) is one of the self-index implementations that supports queries such as *display()*, *count()* and *locate()* (we turned off support for *count()* and *locate()* for this experiment to make the algorithms comparable). Finally, XM (Cao et al. 2007) is a statistical DNA compressor that is also able to compress DNA sequences with respect to a reference sequence and is the DNA compression algorithm with the best compression results that we know of.

For the *S. cerevisiae* dataset, prior to RLZ-opt, COMRAD had the best compression results. However, RLZ-opt was able to compress the dataset to a lower size of 0.15 bpb and this is almost half of the compressed size achieved by RLZ-std. For the *S. paradoxus* results, XM had the best results compared to RLZ-std, but RLZ-opt was able to achieve an equivalent result to XM. For the *H. sapien* results, the non-RLZ algorithms were not able to compress the dataset very well. RLZ-opt was also unable to achieve a much better result compared to RLZ-std. However, most of the 753.90 Mbytes with RLZ-std (or the 707.15 Mbytes with RLZ-opt) consists of the compressed reference sequence, which has a size of 645.34 Mbytes. The compressed size of the three other genomes was 111.56 Mbytes (0.10 bpb) using RLZ-std and 64.81 Mbytes (0.06 bpb) using RLZ-opt, almost a halving of the compressed size. The overall compressed result would also improve when more genomes are available to be added to the dataset. The compression and decompression times for RLZ are much lower compared to the other algorithms. RLZ-opt takes slightly longer to compress compared to RLZ-std but RLZ-opt is faster to decompress as was discussed previously.

As a note on memory usage, in order to support the lookahead functionality, memory usage is three times that of the usage when standard factorisation is used, but this is still a small proportion of overall collection size. When standard factorisation is

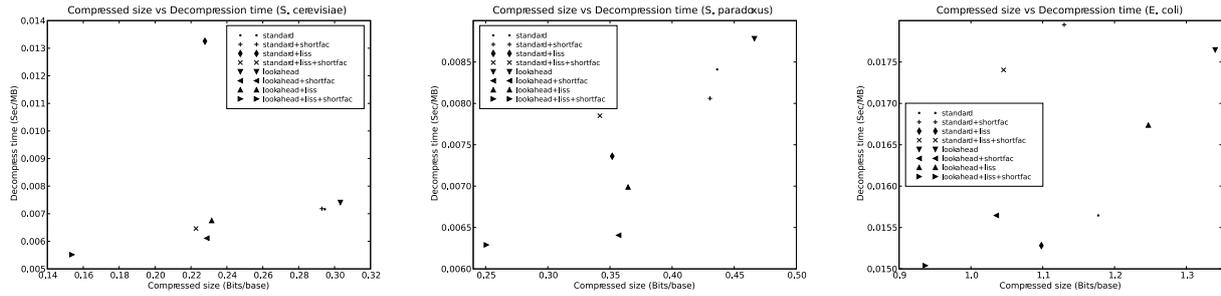
⁵ftp://ftp.ncbi.nih.gov/genomes/H.sapiens/Assembled.chromosomes/hs_ref.GRC37_chr*.fa.gz

⁶ftp://ftp.ncbi.nih.gov/genomes/H.sapiens/Assembled.chromosomes/hs_alt.HuRef_chr*.fa.gz

⁷ftp://ftp.kobic.kr/pub/KOBIC-KoreanGenome/fasta/chromosome*.fa.gz

⁸ftp://public.genomics.org.cn/BGI/yanhuang/fa/chr*.fa.gz

Figure 6: Space-decompression time tradeoff for the RLZ variants.



Dataset	S. cerevisiae				S. paradoxus				H. sapien			
	Size (Mbytes)	Ent. (bpb)	Comp. (sec)	Dec. (sec)	Size (Mbytes)	Ent. (bpb)	Comp. (sec)	Dec. (sec)	Size (Mbytes)	Ent. (bpb)	Comp. (sec)	Dec. (sec)
Original	485.87	2.18	—	—	429.27	2.12	—	—	11831.71	2.18	—	—
RLCSA	41.39	0.57	781	312	47.35	0.88	740	295	3834.82	2.54	34525	14538
COMRAD	15.29	0.25	1070	45	18.33	0.34	1068	50	2176.00	1.44	28442	1666
XM	74.53	1.26	18990	17926	13.17	0.25	30580	28920	—	—	—	—
RLZ-std	17.89	0.29	143	9	23.38	0.44	182	6	753.90	0.51	15451	573
RLZ-opt	9.33	0.15	233	8	13.44	0.25	241	6	707.15	0.48	17861	526

Table 1: Compression results for two repetitive yeast collections and a set of four human genomes. The first row is the original size for all datasets (size in megabases), the remaining rows are the compression performance of RLCSA, COMRAD and XM algorithms followed by the standard RLZ algorithm and the improved RLZ algorithm using lookahead+liss+shortfac algorithms. The four columns per dataset show the size in Mbytes, the 0-order entropy (in bits per base), time taken to compress (in seconds) and time taken to decompress (in seconds), respectively. The time taken to compress includes the time taken to generate the suffix arrays and other associated data structures, and the time to compress the reference sequence. The time taken to decompress includes the time taken to decompress the reference sequence.

used, the memory usage for *S. cerevisiae*, *S. paradoxus* and *E. coli* are 45.78 Mbyte, 43.58 Mbyte and 17.22 Mbyte, respectively. With lookahead factorisation, 118.75 Mbyte, 113.0 Mbyte and 43.89 Mbyte, respectively. This is due to the increased space ($2n \log n$ bits) required to store the inverse suffix array SA^{-1} (satisfying the property $SA^{-1}[SA[i]] = i$) and longest common prefix (LCP) array SA_{LCP} ($SA_{LCP}[i]$ contains the length of the longest common prefix between $SA[i-1]$ and $SA[i]$) to implement a variation of the efficient non-greedy parsing algorithm described in Section 4.

7 Concluding Remarks

As described in our previous work (Kuruppu et al. 2010) RLZ allows fast access to arbitrary parts of the collection with a very slight space overhead. For the non-greedy parsings we have considered in this paper, fast access can be achieved by applying the same data structures. For the approach which separates long and short factors (using the LISS), more care is required to achieve random access as the position components of the long factors are differentially encoded. The key idea to enabling random access is to store absolute position values periodically in the sequence of differences.

Our next aim is to augment the RLZ representation with succinct data structures that allow fast indexed search over the compressed collection. We believe this functionality can be added to our approach, using yet unpublished techniques, not dissimilar to earlier work (Navarro 2004, 2008).

Finally, while we were able to drastically improve compression via the presence of a LISS in the factors,

the best way to exploit this phenomenon remains unclear and warrants further analysis. Use of the LISS turns RLZ from a one pass to a two pass algorithm. An online LISS finding algorithm, such as that of Liben-Nowell et al. (2006), may be effective in our setting as the LISS is long (roughly half the sequence) and tends to manifest early in the factor set.

With the cost of acquiring a single human genome falling below \$10,000, the volume of genomic data will grow dramatically; with our methods, collections of related genomes can be stored with great efficiency. Approaches such as ours are key to management of the volumes of biological data now being produced.

References

- Abouelhoda, M. I., Kurtz, S. & Ohlebusch, E. (2004), ‘Replacing suffix trees with enhanced suffix arrays’, *Journal of Discrete Algorithms* **2**(1), 53–86.
- Apostolico, A. & Lonardi, S. (2000), Compression of biological sequences by greedy off-line textual substitution, in ‘DCC ’00: Proceedings of the Conference on Data Compression’, pp. 143–152.
- Behzadi, B. & Fessant, F. L. (2005), ‘DNA compression challenge revisited: A dynamic programming approach’, *Symposium on Combinatorial Pattern Matching* **3537**, 190–200.
- Brandon, M., Wallace, D. & Baldi, P. (2009), ‘Data structures and compression algorithms for genomic sequence data’, *Bioinformatics* **25**(14), 1731–1738.
- Cao, M. D., Dix, T., Allison, L. & Mears, C. (2007), A simple statistical algorithm for biological sequence

- compression, in 'DCC '07: Proceedings of the Conference on Data Compression', pp. 43–52.
- Chang, W. I. & Lawler, E. L. (1994), 'Sublinear approximate string matching and biological applications', *Algorithmica* **12**(4–5), 327–344.
- Chen, X., Kwong, S. & Li, M. (2000), A compression algorithm for DNA sequences and its applications in genome comparison, in 'RECOMB '00: Proceedings of the fourth annual international conference on Computational molecular biology', ACM, p. 107.
- Chen, X., Li, M., Ma, B. & Tromp, J. (2002), 'DNA-Compress: fast and effective DNA sequence compression', *Bioinformatics* **18**(12), 1696–1698.
- Christley, S., Lu, Y., Li, C. & Xie, X. (2009), 'Human genomes as email attachments', *Bioinformatics* **25**(2), 274–275.
- Ferragina, P., Nitto, I. & Venturini, R. (2009), On the bit-complexity of Lempel-Ziv compression, in 'SODA '09: Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms', Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 768–777.
- Grumbach, S. & Tahi, F. (1993), Compression of DNA sequences, in 'DCC '93: Proceedings of the Conference on Data Compression', pp. 340–350.
- Gusfield, D. (1997), *Algorithms on Strings, Trees, and Sequences : Computer Science and Computational Biology*, Cambridge University Press, Cambridge, UK.
- Haussler, D. et al. (2009), 'Genome 10k: A proposal to obtain whole-genome sequence for 10 000 vertebrate species', *Journal of Heredity* **100**(6), esp086–674.
- Horspool, R. N. (1995), The effect of non-greedy parsing in Ziv-Lempel compression methods, in 'DCC '95: Proceedings of the Conference on Data Compression', IEEE Computer Society, Washington, DC, USA, p. 302.
- Korodi, G. & Tabus, I. (2005), 'An efficient normalized maximum likelihood algorithm for DNA sequence compression', *ACM Trans. Inf. Syst.* **23**(1), 3–34.
- Korodi, G. & Tabus, I. (2007), Normalized maximum likelihood model of order-1 for the compression of DNA sequences, in 'Data Compression Conference, 2007. DCC '07', pp. 33–42.
- Kuruppu, S., Beresford-Smith, B., Conway, T. & Zobel, J. (2009), 'Repetition-based compression of large DNA datasets', Poster at: 13th Annual International Conference on Research in Computational Molecular Biology (RECOMB09).
- Kuruppu, S., Puglisi, S. J. & Zobel, J. (2010), Relative Lempel-Ziv compression of genomes for large-scale storage and retrieval, in 'Proc. 17th International Symposium on String Processing and Information Retrieval (SPIRE)', LNCS, Springer. To appear.
- Liben-Nowell, D., Vee, E. & Zhu, A. (2006), 'Finding longest increasing and common subsequences in streaming data', *Journal of Combinatorial Optimization* **11**(2), 155–175.
- Maaß, M. G. (2006), 'Matching statistics: efficient computation and a new practical algorithm for the multiple common substring problem', *Software, Practice and Experience* **36**(3), 305–331.
- Mäkinen, V., Navarro, G., Sirén, J. & Välimäki, N. (2010), 'Storage and retrieval of highly repetitive sequence collections', *Journal of Computational Biology* **17**(3), 281–308.
- Matsumoto, T., Sadakane, K. & Imai, H. (2000), 'Biological sequence compression algorithms', *Genome Informatics* **11**, 43–52.
- Navarro, G. (2004), 'Indexing text using the Lempel-Ziv trie', *Journal of Discrete Algorithms* **2**(1), 87–114.
- Navarro, G. (2008), 'Indexing LZ77: the next step in self-indexing', Keynote talk at 3rd Workshop on Compression, Text and Algorithms. Slides: <http://spire2008.csse.unimelb.edu.au/talks/gn08-wcta.pdf>.
- Rivals, E., Delahaye, J., Dauchet, M. & Delgrange, O. (1996), A guaranteed compression scheme for repetitive DNA sequences, in 'DCC '96: Proceedings of the Conference on Data Compression', p. 453.
- Schensted, C. (1961), 'Longest increasing and decreasing subsequences', *Canad. J. Math* (13).
- Ziv, J. & Lempel, A. (1977), 'A universal algorithm for sequential data compression', *IEEE Transactions on Information Theory* **23**(3), 337–343.

A Novel Bit Freezing Technique to Improve Gene Specific Co-regulation Discovery from Gene Expression Databases

Ji Zhang¹, Yonglong Luo²

¹ Department of Mathematics and Computing,
University of Southern Queensland, Australia

² Department of Computer Science,
Anhui Normal University, Wuhu, China

Abstract

Discovering gene co-regulatory relationships is a new yet important research problem in DNA microarray data analysis. The problem of gene specific co-regulation discovery is to, for a particular gene of interest, called the *target gene*, identify its strongly co-regulated genes from the database and the experimental condition subsets where such strong gene co-regulations are observed. The study on this problem can contribute to a better understanding and characterization of the target gene. The existing technique, mainly using genetic algorithm (GA) to discover co-regulation conditional subsets, is slow due to its expensive fitness evaluation and long solution encoding scheme. In this paper, we propose a novel technique to improve the performance of gene specific co-regulation discovery using a bit freezing approach. Through freezing converged bits in the solution encoding strings, this innovative approach can contribute to fast crossover and mutation operations, achieve an early stop of the GA and facilitate the construction of kNN Search Table Plus (kNN-ST⁺) that leads to more accurate approximation of fitness function. Experimental results with a real-life gene microarray data set demonstrate the improved efficiency of our technique compared with the existing method.

1 Introduction

DNA microarray is an enabling technology to provide a global view of the expression of a large number of genes. A gene microarray data set is typically presented as matrix where each row represents a gene and each column is an experimental condition (such as a time point or a sample) when the gene expression is extracted. Finding gene co-regulatory relationships is an important research focus in microarray data analysis. One interesting research problem, called *Single Gene Approach* for gene microarray analysis [14], was recently studied in [17, 18]. This problem can be formulated as follows: *for a particular gene of interest, called target gene, identify its strongly co-regulated genes and the condition subsets where such strong gene co-regulations are observed.* The discovered co-regulated genes and the associated condition subsets are specific to the target gene, which can help biologists to better understand and characterize it. This is useful in many applications such as the investigation of the most differentially expressed genes in a disease study or the function prediction of unknown genes.

Copyright ©2011, Australian Computer Society, Inc. This paper appeared at the 32nd Australasian Computer Science Conference (ACSC 2011), Perth, Australia, January 2011. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 113, Mark Reynolds, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.

This paper proposes an innovative bit freezing technique to further improve the efficiency of the method proposed in [18]. The major technical contributions of this paper are summarized as follows:

1. First, we propose a new technique to freeze the converged bits in the binary solution encoding strings. Those string bits, each representing an experimental condition, will be frozen to '0' or '1' if their values converge. By doing this, the length of the solution encoding strings can be significantly shortened and the overhead involved in crossover and mutation operations can be reduced;
2. Second, by utilizing the bit freezing technique, we have devised a novel approach to achieve an early stop of GA. This approach is able to automatically determine whether the GA should be continued or can be early terminated anytime during the evolution process. This is a very desired feature for the GA as it can potentially save a substantial portion of the computational overhead incur in the GA;
3. Third, we propose kNN Search Table Plus (kNN-ST⁺). It is an improved version of the kNN Search Table (kNN-ST) that was previously used in gene co-regulation discovery [18]. Compared with kNN-ST, kNN-ST⁺ can be dynamically re-constructed to seamlessly reflect the progression of bit freezing. As a result, kNN-ST⁺ is smaller than kNN-ST and it provides tighter lower and upper bounds for the fitness function value;
4. Finally, the experimental results with a real-life gene microarray data demonstrate the better efficiency of our technique than that of the existing method in discovering gene specific co-regulations.

Roadmap. The remainder of this paper is organized as follows. Section 2 presents a review of the existing methods for gene specific co-regulation discovery. The general Genetic Algorithm (GA) framework for gene specific co-regulation discovery is discussed in Section 3. Section 4 introduces our bit freezing technique. In Section 5, we elaborate on the improvements that can be achieved through the bit freezing technique. Extensive experimental evaluation results are reported in Section 6 and the last section concludes this paper.

2 Related Work

The traditional gene co-regulation discovery problem mainly aims to find groups of genes from the gene expression database that are similar to each other either in the full set or a subset of experimental conditions. Clustering analysis is arguably the most commonly used technique

for studying gene co-regulations by grouping closely co-regulated genes together. The major clustering algorithms in discovering gene co-regulations include hierarchical clustering method [11], k -means algorithm [11], Self-Organization Maps (SOMs) [12] and SVD-based clustering algorithm [8]. Because they mainly perform clustering based on the entire set of conditions (*i.e.*, full dimensionality), thus they miss out those interesting co-regulations embedded in the lower dimensional condition subsets. To find the gene co-regulations in some subsets of conditions, a few subspace clustering methods for gene expression data, such as Coupled Two-Way Clustering [7], bi-cluster [3] and δ -cluster [15], are also proposed. However, as have been pointed out in other research work, a common key drawback for clustering methods, no matter whether they find co-regulations on the full or partial dimensionality, is that there is no guarantee that the target gene's most-co-regulated genes are in the same cluster where the target gene is located. They may be located in a few different clusters because the results of clustering are quite sensitive to the parameters such as the number of clusters to be obtained or the value of the inter-cluster dissimilarity threshold that users choose. In addition, which is probably more computationally prohibitive, the clustering analysis needs to be performed in a large number of condition subsets. As such, clustering is not a direct nor efficient way for discovering gene-specific co-regulations.

There are also some work on gene co-regulation discovery from time series perspective by considering the explicit temporal nature of the features in the gene microarray data [1][2][4][5][6]. However, there are cases that the features do not have explicit temporal meaning and gene co-regulations may occur in the subsets of features that are far apart from each other in the gene microarray data set.

Recently, there are some research work studying the mining of the gene specific co-regulations particularly [17, 18]. Instead of resorting to clustering techniques, they model the co-regulations directly from the single gene perspective. Consequently, they are more efficient and effective to unveil gene specific co-regulations. In [17], the authors proposed an approach for mining local gene-specific co-regulation using genetic algorithm (GA) [9]. The basic idea of this approach is to first find the condition subsets in which the target gene g is most significantly co-regulated with others and the co-regulated genes of g are then selected from its nearest neighbors in these condition subsets. A sliding window is used to scan all the conditions sequentially and the search of condition subsets is performed within each window position. This method is able to find the closely co-regulated genes for the target gene and the associated condition subsets where such co-regulation occur. However, this method is slow. The major speed bottleneck of this approach is the fitness computation in the GA, which involves a k NN search for the target gene in each condition subset. The typically large number of genes in the microarray data and the number of condition subsets evaluated in the GA lead to a slow kNN search.

To enhance the efficiency of the gene specific co-regulation, kNN Search Table (kNN-ST) and integer-based encoding scheme are proposed in [18]. KNN-ST is a lightweight $M \times k$ table that contains the distance between the target gene and each of its k nearest neighbors in each single condition. The advantages of KNN-ST include 1) It is very efficient to produce the lower and upper bounds of the fitness function; 2) It only requires a linear time to construct the table. Please refer to [16] for detailed discussion of kNN-ST and proof for lower and upper bound of the fitness function. Yet, kNN-ST has some major limitations: 1) the whole table is static during the GA and 2) more importantly, the estimation of the fitness function value is based on kNNs in each single condition. This

leads to relatively loose lower and upper bounds of the true fitness function values. Integer-based encoding scheme involves converting binary bits into integer ones in the solution encoding. By doing this, the length of the solution encoding string can be considerably reduced. Nevertheless, in order to produce the same crossover and mutation results as the binary strings, a relatively complex mapping needs to be performed between integer and binary strings. Issues will become more complicated if the crossover locus that is randomly generated falls into an integer itself in the integer-based representation.

3 General Genetic Algorithm Framework

Before our bit freezing technique is discussed, we will first introduce the general algorithmic framework for mining gene specific co-regulations. The problem of gene specific co-regulation discovery typically has two requirements: 1) find strongly co-regulated genes with regard to the target gene and 2) find the associated subsets of conditions where such strong gene co-regulations occur. Because there are a large number of condition subsets that we need to potentially evaluate, then the second requirement is much more computationally difficult to tackle. As this research problem is a search problem by nature, thus Genetic Algorithm (GA) is chosen in previous papers (and this paper as well) to find those good condition subsets. Once the good condition subsets have been found, the top co-regulated genes in each of those condition subsets can be found in a linear complexity order.

Most aspects of the GA design for mining gene co-regulations is identical to the standard GA. An initial population of solutions are first randomly generated. Fitness of these solutions are then evaluated and good solutions are selected to undergo genetic operations such as crossover and mutation to produce the population of the next generation. This process continues until a specific number of generations are finished. The top solutions amongst all the solutions that have been evaluated in the GA will be returned as the final solutions.

In the following, a more detailed discussion is presented on the design of solution encoding and fitness function which that are relatively unique for dealing with this problem.

Solution encoding. Standard binary solution encoding scheme is normally used. In this encoding, all solutions are represented by strings with fixed and equal length M , where M is number of experimental conditions for the gene expression data. Using binary alphabet $\Sigma = \{0, 1\}$ for gene alleles, each bit in the solution will take on the value of "0" and "1", indicating whether or not its corresponding condition is selected, respectively ("0" indicates the corresponding condition is absent and vice versa for "1"). For a simple example, the solution (1,0,0,1,0,1) when $M = 6$ means that this solution is a 3-dimensional subset which contains the 1st, 4th and 6th conditions.

Fitness Function. For gene expression data, Pearson's Correlation Coefficient (PCC) is often used as the similarity metric to indicate how strongly two genes are co-regulated. Suppose $x = \{x_1, x_2, \dots, x_{|s|}\}$ and $y = \{y_1, y_2, \dots, y_{|s|}\}$ are the projections of two genes x and y in a condition subset s . The similarity between x and y in s is formulated as follows:

$$\text{sim}(x, y, s) = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{[n \sum x_i^2 - (\sum x_i)^2][n \sum y_i^2 - (\sum y_i)^2]}}$$

The experimental condition subset search in the GA involves finding those subsets s that are able to maximize

the following fitness function:

$$f(g, s) = \frac{1}{k} \sum_{i=1}^k |sim(g, g_i)|, g_i \in kNNSet(g, s, D)$$

where $kNNSet(g, s, D)$ denotes the set of kNNs from the database D for the target gene g in condition subset s .

4 Bit Freezing Technique in GA

In this section, we will introduce the bit freezing technique. As being mentioned earlier, the critical performance bottleneck in the previous approaches for searching the co-regulated genes of the target genes results from the typically high-dimensionality of the gene expression data, which leads to the high cost in crossover and mutation operations in the GA.

To better tackle this problem, we developed a novel approach to significantly reduce the overhead of the crossover and mutation by heuristically freezing the bits in the binary solution representation (encoding) strings. This idea is motivated by that fact in gene co-regulation discovery problem, some experimental conditions are critical or not important at all with regard to the gene co-regulation phenomena. Therefore, some experimental conditions will be always present or absent in the good solutions generated in the populations. In most cases, we can start to observe such convergence for most, if not all, experimental conditions in the early stage of the GA. When this happens, it will be safe to freeze those bits corresponding to these experimental conditions to either '0' or '1' in the solution encoding strings. The advantage for doing this is obvious: we can reduce the length of solution encoding strings, and the crossover and mutation operations will only be operated on those unfrozen bits, leading to a fast generation of new solutions in each generation of the GA. This strategy can improve the overall efficiency of the GA without compromising the quality of the co-regulated experimental conditions produced by the algorithm. In addition, reducing the length of the encoding strings through freezing binary bits is more natural and straightforward than the integer-based encoding proposed in [18] and can be implemented more easily.

4.1 w Moving Average of Bit Value

To quantitatively identify those converged experimental conditions, we calculate the moving average of the value of each bit in the solution encoding strings. A sliding window with a size of w is used that covers w consecutive generations to compute the moving average.

As there are a number of solutions obtained in each generation, thus the bit value in a given generation is actually the average value of that bit for all the solutions in that generation. It is defined as

$$v(b_i, t) = \frac{\sum_{j=1}^{pop_size} s_j(b_i, t)}{pop_size}$$

where $s_j(b_i, t)$ denotes the value of i^{th} bit of the j^{th} solution in the t^{th} generation, and we have $1 \leq i \leq M$, $1 \leq j \leq pop_size$ and $1 \leq t \leq Gen$. pop_size corresponds to the number of solutions generated in each generation and it is assumed to be fixed for all generations in the GA. Gen is the maximum number of generations the GA will performs.

w moving average of a bit b_i at the t^{th} generation in the GA, denoted as $w_MA(b_i, t)$, is defined as the average bit value in k most recent consecutive generations, that is

$$w_MA(b_i, t) = \frac{\sum_{j=t-k+1}^t v(b_i, j)}{k}$$

4.2 Design for Bit Freezing

It is very simple and straightforward to design the approach to freeze bits. It involves specifying the values of two thresholds, T_{low} and T_{high} , used to freeze bits to '0' or '1', respectively. Due to the use of these two parameters, the range $[0.0, 1.0]$, where bit values distribute, is divided into three sub-regions, namely, frozen-to-'0' region ($[0.0, T_{low}]$), unfrozen region ($[T_{low}, T_{high}]$) and frozen-to-'1' region ($[T_{high}, 1.0]$). The rules used to freeze a bit b_i at the t^{th} generation are detailed as follows:

1. **Rule for freezing b_i to '0'**: if $w_MA(b_i, t) \leq T_{low}$, then b_i is frozen to '0' thereafter;
2. **Rule for freezing b_i to '1'**: if $w_MA(b_i, t) \geq T_{high}$, then b_i is frozen to '1' thereafter;
3. **Rule for keep b_i active (unfrozen)**: If b_i does not satisfy the above two rules, then b_i is continued to be an active (unfrozen) bit.

Even though they are human specified parameters, T_{low} and T_{high} are fairly intuitive to set. A rule-of-thumb is that T_{low} should be relatively close to 0 while T_{high} should be relatively close to 1. $T_{low} \in [0.15, 0.25]$ and $T_{high} \in [0.75, 0.85]$ are the reasonable value ranges for both parameters.

The size of the above regions has an effect on the overall performance of the GA. The smaller the unfrozen region is, the higher speed can be generally achieved, even though there is a higher chance that a small number of truly co-regulated experimental conditions may be missed. But based our experimental evaluation, the above parameter specification works quite well.

Under our bit freezing framework, it is necessary to record the state of all the bits in the solution encoding throughout the GA. We use Bit State Table (BST) to accomplish this. BST is defined as follows.

Definition 1. A **Bit State Table (BST)** is a $1 \times M$ table where each entry takes one of three symbols, '0', '1' or '?'. M corresponds to the total number of experimental conditions in the gene expression data. The i^{th} entry in BST takes the symbol of '0' (or '1' or '?') if the i^{th} bit in the encoding string is frozen to '0' (or frozen to '1' or remains active).

Please note that BST is dynamic in nature as its entries may be changed from '?' to '0' or '1' sometime in the process of the GA. All the entries in a BST take the symbol of '?' initially at the beginning of the GA. '1' bits and '0' bits cannot change to other states while active bits can change to either '1' or '0' any time or remain active for the whole life span of the GA. Once it has been frozen, the bit will no longer participate in the crossover and mutation operations in the subsequent generations of the GA. Those active bits will continue to be involved in the crossover and mutation operations until they are frozen to '1' or '0' or the end of the GA is reached, whichever comes earlier.

4.3 Starting Generation for Bit Freezing

All the binary bits in solution encoding strings are "active" at the beginning of the GA and we do not start to freeze bits immediately when GA commences. Instead, we wait until a specific number of generations have elapsed before bit freezing evaluation is triggered. This is to give the GA

sufficient time to reach convergence for some, if not all, bits.

The generation when bits are started to be frozen is denoted as G_{start} . A large value for this parameter leads to a more accurate judgment of the bit convergence but delaying bit freezing implies a relatively higher initial search overhead for the GA. This parameter is also relatively easy to set. Generally speaking, 20 generations will be enough for most scenarios.

5 Performance Improvement Using Bit Freezing Technique

The immediate improvement that can be achieved using the bit freezing technique is the faster speed for crossover and mutation operations. When they are frozen, bits will no longer participate in the crossover and mutation operations in the subsequent generations. Thus, the *shortened solution encoding* which only contains the active bits from BST can be safely used. Both crossover and mutation on the shortened solution encoding are performed in the same way as on other regular binary strings. The only difference is that the new children generated using shortened solution encoding need to be mapped back to the full-length encoding for fitness evaluation. A mapping between bit index in the shortened solution encoding and that in the full solution encoding is therefore required. To facilitate such mapping/conversion process, we devise Bit Index Mapping (BIM) and partial Effective Bit Vector (EBV). BIM is used for mapping bit index between the shortened and the full solution encodings, while EBV quickly shows which experimental condition(s) participate in the condition subset, an information required in the subsequent fitness function evaluation. BIM and EBV are defined as follows.

Definition 2. A **Bit Index Mapping (BIM)** is a $l \times 2$ table with $BIM(i, 1)$ (the first entry of each row) being the index of an active bit b in the shortened solution encoding while $BIM(i, 2)$ (the second entry of the same row) being the index of b in BST.

As an example, let us assume that the BST is (1, ?, 0, ?, ?, 0, ?, 1) and a solution in a shortened encoding is (0, 1, 1, 0), then BIM can be constructed as (1→2, 2→4, 3→5, 4→7), suggesting that the 1st bit in the shortened encoding is mapped to the 2nd bit in the full encoding, and etc.

Definition 3. An **Effective Bit Vector (EBV)** of a **solution** (a condition subset) is a 1-dimensional vector that contains the index of bits that participate in this solution.

For example, the EBV of a solution encoding (1, 0, 0, 1) is (1,4) as the 1st and 4th bits in the solution encoding is effective (*i.e.*, participate in the condition subset).

The entries in the full EBV come from two sources. Some entries can be directly obtained from the '1' bits of the BST, which constitutes a *partial EBV*. Suppose the BST is (1, ?, 0, ?, ?, 0, ?, 1), then the partial EBV obtained from the BST is (1, 8). The remaining entries of the full EBV are obtained by mapping the bits in the shortened solution encoding using BIM.

For a better understanding, an example is given in Figure 1 to show how to convert a shortened solution encoding to the full EBV using BIM and partial EBV. The BIM and partial EBV are obtained offline from the sample BST given on the top of the figure. The shortened solution encoding (0, 1, 1, 0) on the left upper corner represents a sample solution that is obtained from a genetic operation (crossover or mutation). To convert this shortened solution encoding (0, 1, 1, 0) to the full EBV, the following two steps are taken:

1. The mapping index of those '1' bits (effective bits)

are looked up in the BIM and returned. In this example, only the 2nd and 3rd (highlighted) bits in the shortened solution encoding need to be mapped. Other two '0' bits are ignored in this mapping. By looking up the BIM, we obtain the index for these two bits as 4 and 5. This means they are mapped to the 4th and 5th bits in the full solution encoding;

2. Augment the partial EBV with the mapped bit index from Step 1 to produce the full EBV, which is (1, 4, 5, 8), meaning that this solution represents a 4-dimensional subset with contains the 1st, 4th, 5th and 8th conditions.

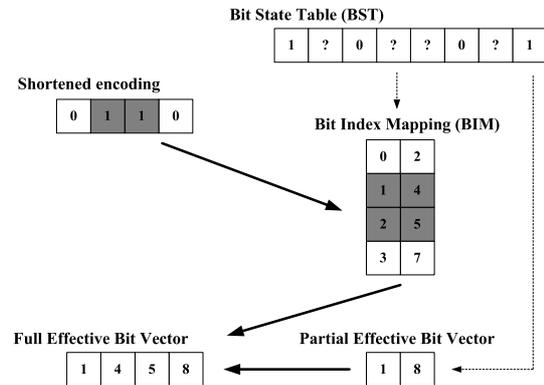


Figure 1: Converting shortened encoding to Full Effective Bit Vector

It is worthwhile pointing out that, due to the dynamic nature of BST, different BIM and partial EBV may be used for different generations in the GA. Both of them are constructed offline and are updated when new bits are frozen.

Complexity analysis. If no bit freezing is applied, then the complexity for both crossover and mutation will be $O(2M)$ including the work to get full EBV, where M will be the number of experimental conditions. If our bit freezing technique is incorporated, the complexity in looking up BIM will be $O(M_{active})$, where $O(M_{active})$ denotes the number of active bits, the complexity of generating full EBV using partial EBV together with the mapped bit index from BIM is another $O(M_{active})$. Totally, the complexity will be $O(2M_{active})$, which is lower than the case where bit freezing is not used. In addition, as M_{active} gets increasingly small as GA proceeds, the overhead saving using bit freezing will become more significant.

6 Experimental Results

This section reports the extensive experimental results on our bit freezing technique. Both efficiency and effectiveness of the algorithm are evaluated.

6.1 Dataset

In our experiments, the Spellman's dataset is used that can be downloaded from <http://genome-www.stanford.edu/cellcycle/data/rawdata>. This dataset contains all the data for the alpha factor, cdc15, and elutriation time courses, and includes the data for the Clb2 and Cln3 induction experiments. We used only the alpha-factor and CDC28 datasets for our experiments, as did in [10]. The dataset used for experimental purpose contains 6178 genes at 35 time points, forming a 6178×35 matrix (this dataset can be downloaded at <http://www.comp.nus.edu.sg/jiliping/p2/YeastData.xls>). Random sampling is performed

horizontally and vertically on the aforementioned real-life dataset to create test datasets with desired number of gene and experimental conditions.

6.2 Experimental Setup

We mainly have the following two groups of parameters to set up in the experiments:

1. *Bit freezing parameters*: Bit freezing starting generation $G_{start} = 20$, the size of the sliding window $w = 10$ for computing the moving average, and bit freezing thresholds $T_{low} = 0.2$ and $T_{high} = 0.8$;
2. *GA-related parameters*: The maximum number of generations for the GA $Gen = 200$, the population size in each generation $pop_size = 50$, the frequency of applying crossover $p_c = 0.8$ and the frequency of applying mutation $p_m = 0.2$.

The above setting is applied for all experiments except those where the bit freezing method parameters, *i.e.*, G_{start} , w , T_{low} and T_{high} , will be varied when experiments are performed to evaluate their respective effect on the performance.

The running time reported in the experiments are averages over 5 samples with the same number of genes and/or conditions. All the experimental evaluations are carried out on a Pentium 4 PC with 2G RAM.

6.3 Competitive Methods to Compare

In the experiments, comparative study will be carried out to compare the performance of our bit freezing technique with existing methods. The competitive methods we choose in the comparative study are the standard GA without major improvements incorporated and the GA method using kNN-ST and integer-based encoding [18].

6.4 Efficiency Study

The speed improvement is an important objective our algorithm needs to achieve. In efficiency study, we investigate the execution time of our bit freezing technique and compare it with the existing methods.

The comparison is first conducted under varying number of genes and experimental conditions in the gene expression database. The results are presented in Figure 2 and 3. The results show that our bit freezing technique is 6-8 times (on average) faster than the standard GA and 1.5-2 times (on average) faster than the method using kNN-ST and integer-based solution encoding under both varying number of genes and experimental conditions. The performance improvement is contributed by the three boosting strategies, *i.e.*, faster genetic operations (crossover and mutation), early stop of GA and kNN-ST⁺. To get a better idea about the efficiency contribution for each of these three boosting strategies, we conduct component analysis to find out what percentage of efficiency improvement can be achieved by using each strategy alone. Figure 4 presents two pie charts, demonstrating the contribution of the three strategies when compared with the two competitive methods. The comparison result with the standard GA and kNN-ST&integer-based encoding method are presented in Figure 4(a) and 4(b), respectively. There are some intriguing findings from these two pie charts. First, the major playing factors in efficiency improvement against the standard GA are shortened solution encoding and the kNN-ST⁺. However, when comparing with the more recent work that uses kNN-ST and integer-based encoding, the contribution of these two factors becomes remarkably smaller while the early stop of GA emerges to become the major contributor.

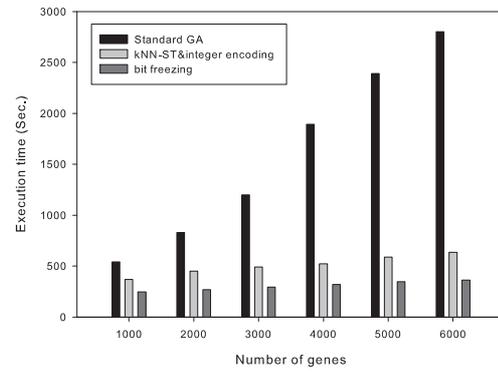


Figure 2: Speed comparison amongst three methods under varying number of genes

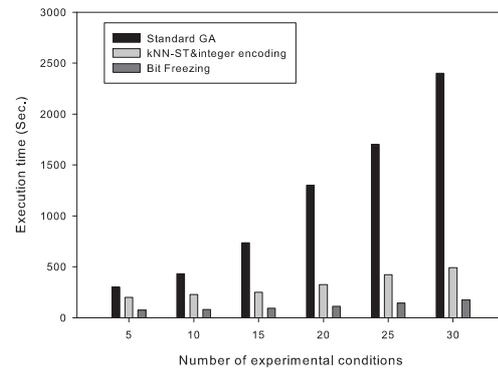


Figure 3: Speed comparison amongst three methods under varying number of experimental conditions

6.5 Effectiveness Study

Effectiveness study investigates how well the GA method can find the good condition subsets when bit freezing technique is incorporated. For this study, an appropriate effectiveness metric needs to be devised. *Accuracy* is utilized in this experiment to measure, amongst all the solutions returned by the bit freezing enabled GA, how many of them are really the top solutions. It is defined as follows:

$$Accuracy = \frac{a}{top_n} \times 100\%$$

where a is the number of true top_n solutions returned by our algorithm. The critical issue involved in calculating accuracy is that we need to first obtain the so-called golden truth result, *i.e.*, the set of the true top_n solutions. Unfortunately, due to the NP nature of the research problem, the set of the true top_n solutions are unknown. However, it is still possible for us to obtain a good approximation of true top_n solutions through the following strategy. The standard GA for mining gene specific co-regulations are first executed for multiple times and the top solutions with regard to the target gene are then collected, from where the approximated set of top solutions are generated. To ensure that the approximated set of top solutions are as close as possible to the real ones, we continue to run the algorithm until the top solutions returned become stabilized. This stabilized set of solutions are used as the golden truth result for evaluating our bit freezing enabled GA method in the effectiveness study.

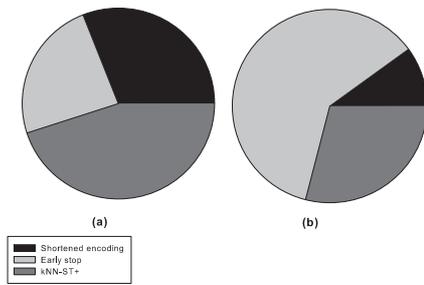


Figure 4: Efficiency improvement contribution analysis

Top-n	100	200	300	400	500
Accuracy	86.3%	89.4%	90.5%	93.9%	94.6%

Figure 5: Accuracy result

We evaluate the accuracy of our method under different values of top_n . top_n denotes the number of top solutions to be obtained at the end of the GA. In the experiment, top_n ranges from 100 to 500 and the corresponding accuracy of our method is shown in Figure 5. The results reveals that our method performs pretty accurate and bit freezing does not compromise the effectiveness of the method. Furthermore, the accuracy of our method gets better as top_n increases, indicating that our method works particularly well when a relatively large number of top condition subsets are sought.

7 Conclusions and Future Work

This paper presents a novel bit freezing technique to improve the efficiency of gene co-regulation discovery from large gene expression data. This bit-freezing technique is motivated by the inherent convergence behavior of bit value for most of the experimental conditions. Our technique is promising as it offers a number of different strategies to improve efficiency, including speed-up of crossover and mutation operations, early stop the GA whenever it is possible and construction of dynamic kNN Search Table Plus for reducing the number of solution candidates that have to be otherwise accurately evaluated.

We have conducted extensive experiments for evaluating our method. Satisfactory speed and accuracy performance are achieved when bit freezing technique is incorporated in the GA. We believe similar performance can be achieved for other gene expression databases under reasonable experimental setup.

In the future, we will continue to work on enhancing the performance of gene co-regulation discovery problem. Some interesting issues are worthy of further investigation. For example, how to generalize our bit-freezing approach so that it can be applied to those cases where non-binary individual encoding are used? How can our bit-freezing approach be used to alleviate the problem of missing data value commonly associated with gene expression data? We believe the research into these intriguing issues will contribute to the further development of the bit-freezing approach and lead to new research findings.

8 Acknowledgment

The authors would like thank Dr. Paulo deSouza from CSIRO ICT Centre Australia for his constructive input to this paper.

References

- [1] R. Amato, A. Ciaramella, N. Deniskina, C. Del Mondo, D. di Bernardo, C. Donalek, G. Longo, G. Mangano, G. Miele, G. Raiconi, A. Staiano and R. Tagliaferri. 2006. A Multi-step Approach to Time Series Analysis and Gene Expression clustering. *Bioinformatics*, 22(5): 589-596.
- [2] Z. Bar-Joseph. Analyzing time series gene expression data. 2004. *Bioinformatics*, 20(16):2493-2503.
- [3] Y. Cheng and G.M. Church, Biclustering of Expression Data. 2000. In *Proc. International Conference on Intelligent Systems for Molecular Biology (ISMB)*, vol. 8, pp. 93-103.
- [4] S. Erdal, O. Ozturk, D. Armbruster, H. Ferhatosmanoglu, and W. Ray. 2004. A time series analysis of microarray data. In *4th IEEE International Symposium on Bioinformatics and Bioengineering*.
- [5] J. Feng, P. E. Barbano, and B. Mishra. 2004. Time-frequency feature detection for timecourse microarray data. In *2004 ACM Symposium on Applied Computing (SAC'04)*.
- [6] V. Filkov, S. Skiena, and J. Zhi. 2001. Analysis techniques for microarray time-series data. In *5th Annual International Conference on Computational Biology*.
- [7] G. Getz, E. Levine, and E. Domany. 2000. Coupled Two-Way Clustering Analysis of Gene Microarray Data, in *Proc. National Academy of Science*, vol. 97, no. 22, pp. 12079-12084.
- [8] G. H Golub and C. F. Van Loan. 1983. *Matrix Computations*, Johns Hopkins University Press.
- [9] J. Holland. 1992. *Adaption in Natural and Artificial Systems*. MIT Press, Cambridge.
- [10] L. Ji and K. L. Tan. 2005. Identifying Time-Lagged Gene Clusters on Gene Expression Data. *Bioinformatics*, Vol. 21, No. 4, pp. 509-516.
- [11] R. A. Johnson and D. W. Wichern. 1998. *Applied Multivariate Statistical Analysis*, Prentice Hall International, USA.
- [12] T. Kohonen. 1995. *Self-Organization Maps*. Springer-Verlag, Berlin Heidelberg.
- [13] M. F. Ramoni, P. Sebastiani, and I. S. Kohane. 2002. Cluster analysis of gene expression dynamics. *Proceedings of the National Academy of Sciences, USA*, 99(14):9121-9126.
- [14] T. Speed, J. Fridlyand, Y. H. Yang and S. Dudoit. 2001. Discrimination and clustering with microarray gene expression data. *2001 Spring Meeting of International Biometric Society Eastern North American Region (ENAR'01)*.
- [15] J. Yang, W. Wang, H. Wang, and P.S. Yu. 2002. δ -Cluster: Capturing Subspace Correlation in a Large Data Set. In *Proc. 18th International Conference on Data Engineering (ICDE'02)*, pp 517-528.

- [16] J. Zhang, Q. Gao, and H. Wang. 2006. A Novel Method for Detecting Outlying Subspaces in High-dimensional Databases Using Genetic Algorithm. *ICDM'06*, pp 731-740.
- [17] J. Zhang, Q. Gao and H. Wang. 2006. Discover Gene Specific Local Co-regulations Using Progressive Genetic Algorithm. *ICTAI'06*, pp783-790.
- [18] J. Zhang, Q. Liu and K. Xu. Gene Specific Co-regulation Discovery: An Improved Approach. *2009International Conference on Computational Intelligence (ICCS'09)*, Baton Rouge, USA, 2009.

Enhancing the Believability of Embodied Conversational Agents through Environment-, Self- and Interaction-Awareness

Kiran Ijaz¹Anton Bogdanovych²Simeon Simoff²

¹ Faculty of Engineering and Information Technology
University of Technology,
Sydney, NSW, Australia.
Email: kijaz@it.uts.edu.au

² School of Computing and Mathematics
University of Western Sydney,
NSW, Australia.
Email: {a.bogdanovych, s.simoff}@uws.edu.au

Abstract

Research on embodied conversational agents' reasoning and actions has mostly ignored the external environment. This paper argues that believability of such agents is tightly connected with their ability to relate to the environment during a conversation. This ability, defined as awareness believability, is formalised in terms of three components - environment-, self- and interaction-awareness. The paper presents a method enabling virtual agents to reason about their environment, understand the interaction capabilities of other participants, own goals and current state of the environment, as well as to include these elements into conversations. We present the implementation of the method and a case study, which demonstrates that such abilities improve the overall believability of virtual agents.

Keywords: Virtual Worlds, Artificial Intelligence, Embodied Agents

1 Introduction

Virtual agents and online virtual worlds they populate are a catalyst that will accelerate the development of embodied Artificial Intelligence (AI) (Livingstone 2006). Having both humans and agents fully immersed into and constrained by the same computer-simulated environment provides fantastic opportunities for embodied AI researchers to study human behaviour, investigate cognition related aspects and search for more practical application domains for the discipline. Virtual agents have the opportunity to engage and learn from their interaction with the social network of human beings operating in these worlds. However, for that to happen, they need to keep humans engaged in meaningful joint activities - they need to be *believable* in terms of their "life" in the computerised environment they operate in.

Surprisingly enough, the majority of virtual agents are not fully integrated with their environment in terms of their reasoning. Most of existing research in the area is focused on agents' conversational abilities (Gandhe & Traum 2007), use of gestures (Hart-

mann et al. 2005) and emotions (Cunningham et al. 2005), social role awareness (Prendinger & Ishizuka 2001) while providing the agent with a limited awareness about the objects in the complex dynamic virtual world ((Johnson & Lester 2000) and (Lester et al. 1999)), its own state in relation to the environment and ignoring the interactions of other participants with the environment (Doyle 2002).

To illustrate the importance of integrating such features into agent conversations consider a scenario outlined in Figure 1. Here a human user learns in a Virtual World about the history and culture of the ancient city of Uruk (Bogdanovych et al. 2010) through controlling an avatar of Fisherman Anel. There is a clear task he has to accomplish (catch some fish) and this task involves interacting with virtual agents representing ancient citizens of Uruk. As shown in the figure the human isn't clear about how to progress toward the assigned task and asks the agent (Fisherman Jigsaw) about his options. To be able to reply in a similar manner as shown in the picture, the agent must know its interaction state, its location in the environment in relation to other objects and avatars, its own goals and beliefs and interaction capabilities of the user.

In 3D Virtual Worlds the integration of agents and environment in terms of agent reasoning is a feasible task due to the fact that the perception problem in such environments is minimised. Each agent can operate with precise coordinates of other participants and objects in the environment, request their names, properties, distances to them and operate with a number of own parameters (i.e. eye direction, body rotation etc.) to determine visibility of the objects, predict the movement of other actors and identify the target of their attention. Furthermore, in virtual worlds like Second Life (<http://www.secondlife.com>) it is even possible to supply agents with information about the elements constituting a particular object, as each of the objects there is composed of a number of primitives. We envision that by developing such mechanisms to utilise this information provides a very powerful toolkit for a sophisticated agent reasoning apparatus that significantly increases the believability of agent behaviour and its capacity to engage humans.

In this paper, we suggest that supplying virtual agents with an ability to reason about the objects in their environment, own state, goals and beliefs as well as the interactions of other participants would result in more believable virtual agents, which can be used in a much wider range of problems than at present. The

Copyright ©2011, Australian Computer Society, Inc. This paper appeared at the 32nd Australasian Computer Science Conference (ACSC 2011), Perth, Australia, January 2011. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 113, Mark Reynolds, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.



Figure 1: Scenario: A conversation between two Fishermen.

remainder of the paper is structured as follows. In Section 2 we analyse the concept of believability and investigate believability features. In Section 3, we discuss and formalise the believability features that haven't received an appropriate research attention. Section 4 presents our approach to integrating these features into virtual agents. In Section 5, we illustrate our approach through a case study and evaluate it in Section 6. Finally, Section 7 concludes the paper.

2 Believability of Virtual Agents

Virtual agents are represented as graphical characters (avatars) that may or may not resemble a human body. It should always be the goal for these agents to act "believably" given the representation they have; hence, enabling and evaluating their believability is essential.

2.1 The Notion of Believability

The notion of believability originates in the field of animation and theatre (Mateas 1997). A believable character can be defined as a lifelike in its behaviour, whose actions appear to be real, who engages its audience and is perceived as human-like. A classical work of Walt Disney Studio on these animated characters "illusion of life" (Thomas & Johnston 1981) elaborates on the requirements for believability. Though these characters are not real they continue to impact the audiences' imagination to accept them as believable. Believability and realism have been differentiated by (Mateas 1997) and (Doyle 2002). According to the authors, a believable character does not necessarily mean to be a real character or who tells the truth; it is real in the context of its environment.

Earlier research in believability has been heavily influenced by the Carnegie-Melon set of requirements for believable agents, which is based on research in drama and story telling (Loyall 1997). These include personality, self-motivation, change, social relationships, and "illusion of life". Personality infuses everything that a character does - behaviour, style, "thought", "emotion", e.g. unique ways of do-

ing things. Self-motivation assumes that agents have their own internal drives and desires which they pursue whether or not others are interacting with them, and they demonstrate their motivation. Change implies that characters change with time, in a manner consistent with their personality. Agents' behaviour and interactions should be consistent with their social relationships. "Illusion of life" is used as a label for a collection of features such as: pursuing multiple, simultaneous goals, having elements of broad capabilities (e.g. movement, perception, memory, language), and reacting quickly to stimuli in the environment.

Contemporary AI uses the term "believability" in relation to engaging life-like systems. Reactivity, interactivity and appropriate decision making while observing behaviours are few characteristics which make autonomous agents suitable to achieve believability (Riedl & Stern 2006). Believability is also a main requirement of modern computer games. As suggested by (Livingstone 2006), "the need for modern computer games is not unbeatable AI, but believable AI".

2.2 Believability of Embodied Conversational Behaviour

Current research in believability of embodied conversational behaviour of virtual agents is focused on the believability aspects of the conversation itself and the non-verbal communication cues, including facial expressions, gestures and gaze.

2.2.1 Believable Conversations

Believable conversation in relation to the content is a term often associated with "chatbots" - software programs that try to keep a human engaged in a textual or auditory conversation. Virtual agents with such capability have been used in various commercial applications, games, training systems and web-based applications. Chatbots like Eliza (Weizenbaum 1966) and ALICE (Wallace 2004b) are based on pattern matching strategies. Technically, chatbots parse the user input and use keyword pointing, pattern matching and corpus based text retrieval to provide the most

suitable answer from their “knowledge base” (Gandhe & Traum 2007).

2.2.2 Believable Facial Expressions

Facial expressions can be used to complement the word stream through expressing emotions, i.e. happiness, sadness etc (Cunningham et al. 2005). These emotional expressions have cross cultural boundaries, but generally existing work deals with a list of emotion expressions: {happy, sad, fear, anger, disgust, agreement, disagreement and surprise} as presented in (Cunningham et al. 2005).

A comprehensive survey of techniques for automatic analysis of facial expressions was presented by (Pantic & Rothkrantz 2000). It has been further noticed that most of the existing work deals with specific cases of image and video analysis (Pantic & Rothkrantz 2000).

2.2.3 Believable Gestures and Upper Limb Movements

Gestures are one of those factors in non-verbal communication which allow us to interact in a lively manner. Gesture selection and their correct execution may increase the expressivity of the conversation (Hartmann et al. 2005). Believable gestures are related to gestures selection being correctly aligned with the flow of conversation and the generation of realistic movements of agent’s upper limbs during the conversation (Hartmann et al. 2005).

2.2.4 Believable Gaze

Gaze helps to convey the cognitive state of a participant or synchronise a conversation as explained in (Lee et al. 2007). Various gaze models like avert, examining the current task, gaze at visitors, etc. were simulated by (Heylen et al. 2005). They measured the believability of the agent based on factors like satisfaction, engaging, natural eye, head movements and mental load among others; and this study showed the significant improvements in communication between humans and virtual agents. Lance in (Thiebaut et al. 2009) investigated a hybrid approach combining head posture, torso posture and movement velocity of these body parts with gaze shift.

3 Awareness Believability

Awareness is essential part of our conversational behaviour. In a conversation we are aware of where we are (environment awareness), who we are (self-awareness) and generally how the interaction is progressing (interaction awareness). Therefore, awareness is an essential component of the believability of embodied conversational behaviour, which we label as “awareness believability”. Further, we develop each of the subcomponents of awareness believability.

3.1 Environment Awareness

The importance of environment awareness for agent reasoning is best illustrated in (Elpidorou 2010), where it is suggested that our consciousness does not arise from the brain alone but from the brain’s exchange with its environment. Humans are embodied in space and use various cues related to space, like pointing and referring to areas of and things in it, in all they do (for more details see the chapters 1,2 in (O’Keefe & Nadel 1978)).

Existing literature presents a very limited picture on the use of environment awareness by animated agents. Agents like Cosmo (Lester et al. 1999) and Steve (Johnson & Lester 2000) are able to recognise and point to objects in the particular static environment but completely ignore their interactions with other participants, don’t cater for dynamic environment and can not orient themselves in a different environment. We suggest that awareness of environment objects alone would not be enough to achieve complete believability and virtual agents further require to be aware of other participants in the context of time.

Our key features of environment awareness include the positions of objects and avatars in the environment, how these evolve with time and the direction vectors associated with avatars (Gerhard et al. 2004). We formalise environmental awareness as follows:

$$EA = \{Objects, Avatars, Time\} \quad (1)$$

Here EA is the set of components of environment awareness and includes the objects in the environment, other avatars representing agents and human participants with respect to the current time.

3.2 Self-awareness

Knowing own context and state within the environment, i.e. being self aware, is essential for a virtual agent to interact believably (Doyle 2002). To achieve that Doyle (Doyle 2002) proposes to annotate the environment and grant agents with access to this annotation. One of the most studied features of self-awareness for virtual agents and animated characters is social role awareness (Prendinger & Ishizuka 2001). However, self-awareness is a much richer concept and many of its characteristics remain understudied, in particular existing works mostly ignored many vital characteristics that arise in dynamic environments.

Hallowell defines self-awareness (Hallowell 1955) as the recognition of one’s self as an object in the world of objects and highlights the importance of the perception as the key function in self-awareness. The list of elements we have identified to enable self-awareness is as follows:

$$SA = \{G, P, B, Sc, St, ObjUsed, Role, Gest\} \quad (2)$$

Here SA represents the set of components of self-awareness and includes the local goals of the agent (G), its current plans (P) and beliefs (B), current scene where the agent participates (Sc), its state within this scene (St), objects used by the agent ($ObjUsed$), the role it plays ($Role$) and the gestures being executed ($Gest$).

3.3 Interaction-Awareness

Believability of interactions goes beyond traditional focus on modeling the visual co-presence (Gerhard et al. 2005), Context awareness (perceiving other agents/objects in static environment) (Bickmore et al. 2007) and communication style (e.g. short vs long utterances, usage of specific vocabulary) of the agents. Human behaviour in interactions is a result of the mix of being rational, informed, impulsive, and the ability to influence others and cope with the influences from others. All these nuances impact the richness of human interactions, hence, must be taken into account when considering the believability of interactions between virtual agents and humans.

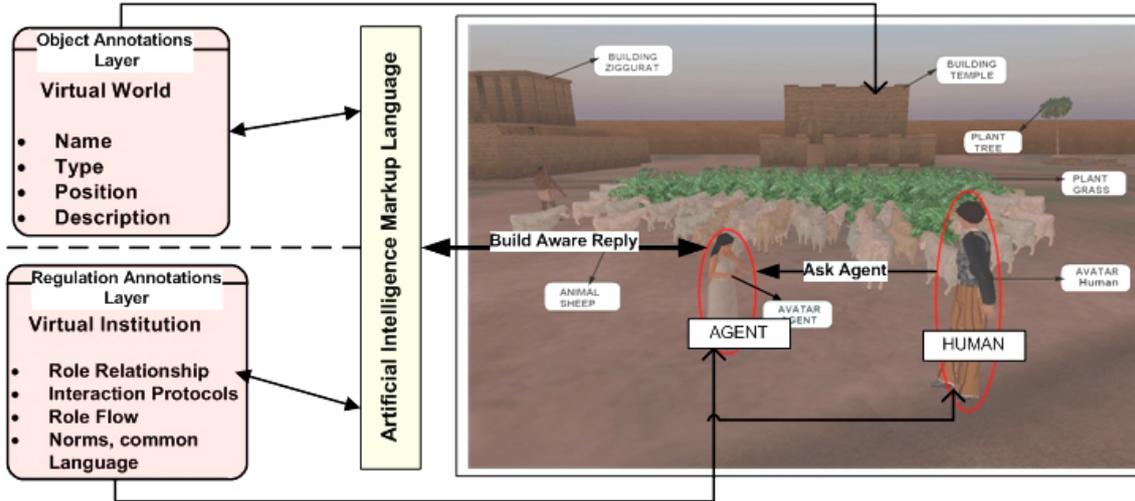


Figure 2: Layered Architecture.

Thus, interaction-awareness is defined as the state of an agent who is “able to perceive important structural and/or dynamic aspects of an interaction that it observes or that it is itself engaged in” (Dautenhahn et al. 2003). The components of the interaction-awareness model are outlined below.

$$IA = \{AV_{vis}, AV_{sc}, Act, Obj, State, Pos, Or\} \quad (3)$$

Here IA represents the set of components included in our interaction awareness model. AV_{vis} corresponds to the set of currently visible avatars. The AV_{sc} is a set of all avatars within the scene where the agent participates in a given moment of time. Act represents the set of actions each of the agents in the current scene is able to perform given its state. Obj refers to the list of objects the avatar can use. $State$ is the state of the avatar in the world. Pos is the position of the agents in the virtual world and Or is agent’s orientation vector in the virtual world space. In the next section we outline the implementation details of our approach.

4 Implementation of Awareness Believability

In order to incorporate the aforementioned believability features in regulated interactive environments we propose to conduct two levels of environment annotation as shown in Figure 2: (i) *object annotation* - the annotation of objects in the environment with appropriate names, object types and descriptions (such annotations are fully supported by the Second Life technology that we use for design and visualisation of the dynamic virtual world); and (ii) *regulation annotation* - annotation of the social norms, interaction protocols, roles and other kinds of regulations of interactions. The Virtual Institutions (VI) technology (Bogdanovych 2007) is used to enable regulation annotation.

In our system human users interact with virtual agents present in the virtual world through a communication layer and these agents rely on two further layers to generate an intelligent response as shown in Figure 2. If the query asked was about the agent’s environment, object annotations layer would be requested by the agent to generate a response which contain the objects’ information. Whereas, queries regarding the agent’s interactions and self awareness would be passed to the regulation annotations(VI)

layer by the agent. VI layer passes interaction annotations (i.e. agent’s goal, plans, objects used etc.) back to the Artificial Intelligence Markup Language (AIML) module to further build a response. This communication module has the responsibility of generating the text reply for the agent based on the information received from the virtual world or the virtual institution layer. Subsequent sections would provide a detail insight about these layers.

Technological separation of the environment layer from the Institutional regulations allowed us to implement a generic solution, where same features could be deployed in a new dynamic environment without modifying the core functionality. Secondly, this layered approach enables to deal with a dynamic environment, where objects could be changed, inserted or deleted at any time. All annotations outlined in Figure 2 can still be detected by the agent for newly added objects. Further, we briefly describe the Virtual Institutions technology and show how it can enable the integration of the awareness believability features in such dynamic virtual worlds.

4.1 Virtual Institutions Technology

The Virtual Institutions technology (Bogdanovych 2007) provides tools for formal specification of institutional rules, verification of their correctness, mapping those to a given virtual world and enforcing the institutional rules on all participants (both humans and autonomous agents) at deployment, creating a *regulated virtual world*.

The specification is expressed through three types of conventions and their corresponding dimensions (Esteva 2003): *Conventions on language – Dialogical Framework*, determines language ontology and illocutionary particles that agents should use, roles they can play and the relationships among the roles; *Conventions on activities – Performative Structure*, establishes the different interaction protocols (scenes) the agents can engage in, and the role flow policy among them. A scene is a logically independent activity that can be enacted by a group of agents. In Virtual Institutions each scene often corresponds to a particular space in the virtual world; *Conventions on behaviour – Norms*, capture the consequences of agents’ actions within the institution (modelled as commitments and obligations) that agents acquire as consequence of some performed actions and that they must fulfil later on. Thus, the Virtual Institutions

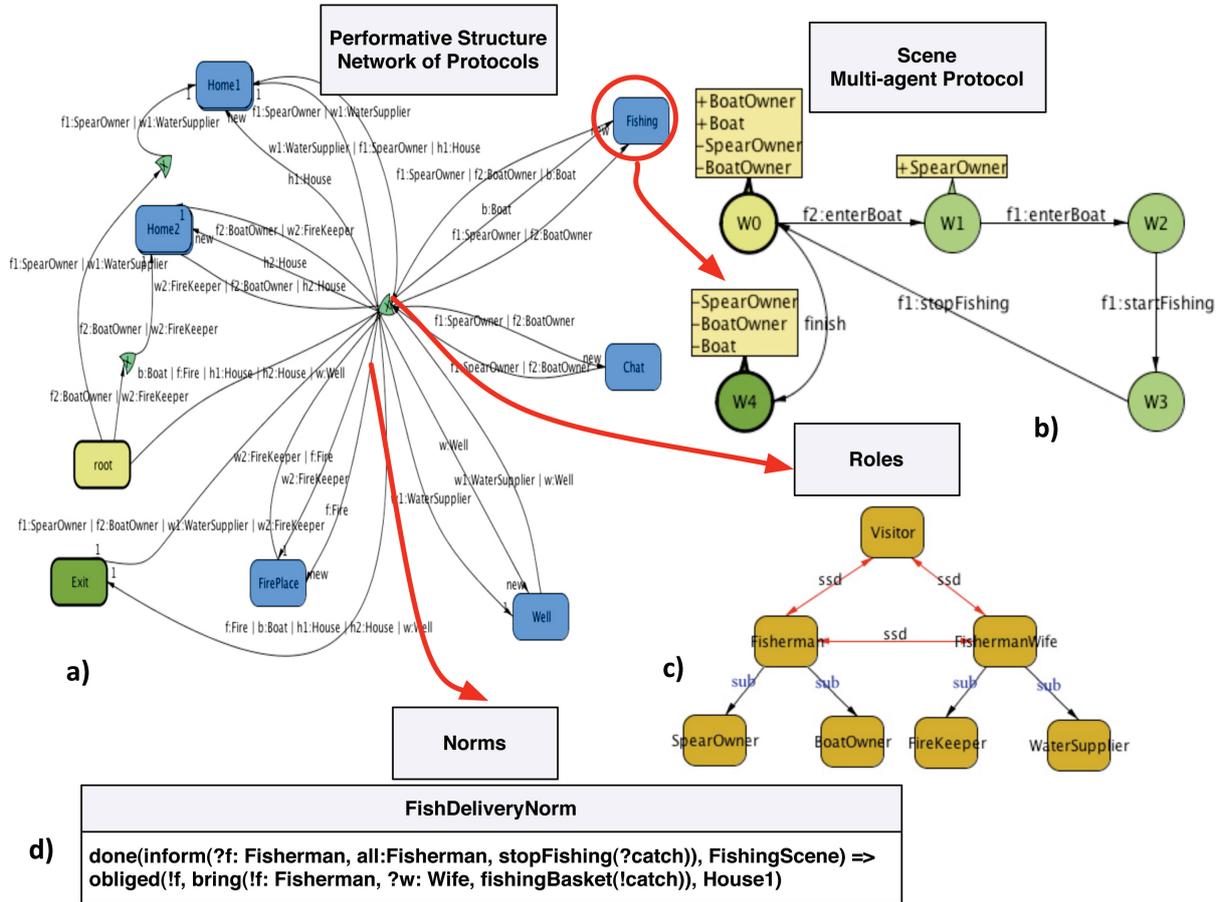


Figure 3: The interaction related information provided by the Normative Platform.

technology helps us define the interaction protocol for all the specified scenes, relationships and role flow among the participants and norms of the society. An example of specifying the aforementioned dimensions is shown in Figure 3.

At deployment, the specification is connected to the virtual world (or respective area, i.e. an island in Second Life) and the state of the virtual world is mapped onto the state of the institution. Having the institutional formalisation aligned with the virtual world drives the decision making of virtual agents controlling the avatars. Each agent has access to the institutional formalisation and can sense the change of the institutional state and reason about the actions (both own actions and actions performed by other participants) that resulted in the state change. Hence, for an agent it makes no difference if it interacts with an agent or a human as the result of the other party’s actions can be sensed through the normative platform and interpreted on a high level using the specification.

4.2 Implementing Interaction-Awareness

Enabling interaction-awareness means making an agent understand its own opportunities in interacting with other participants and predict the possible actions other participants may perform in a given scene provided the agent knows what the current state of the scene is. The VI technology tracks all the participants in every scene and maintains the state of every scene. Every action performed by any participant, as well as the corresponding state changes, can be sensed by the agent through the VI technology.

The VI technology provides the agent with high-level information of interaction opportunities of all participants. The agent can relate this information with its conversational rules and answers questions like “what can I do in this scene”, “what can I do next”, “what are you waiting for”, etc.

The kind of information an agent can get through the VI technology is shown in Figure 3. The role hierarchy of possible participants is shown in Figure 3 c. The Performative Structure (Figure 3 a) illustrates possible role flow and deals with five major scenes so far including fishing, Well, Fire place, chat and fishermen Homes. To give an insight into the interaction protocol Figure 3 b illustrates the enactment of the fishing scene. Here the scene is represented as a finite-state machine where the state of the scene is changing as the result of the defined actions (i.e. enterBoat, startFishing, stopFishing, etc.). Finally, Figure 3 d presents an example of a social norm that all participants must abide by. In this case the norm dictates a fisherman to bring all the catch to his wife.

The agent can sense the movement over the performative structure, agents entering or leaving these scenes, track the actions that result in the state changes and also estimate the actions of other participants that can make the scene evolve to a new state. How this information can be used to dynamically generate text responses to various user queries is explained in the next section.

4.3 Implementing Environment-Awareness

Our environment-awareness method enables embodied virtual agents to have an up-to-date knowledge

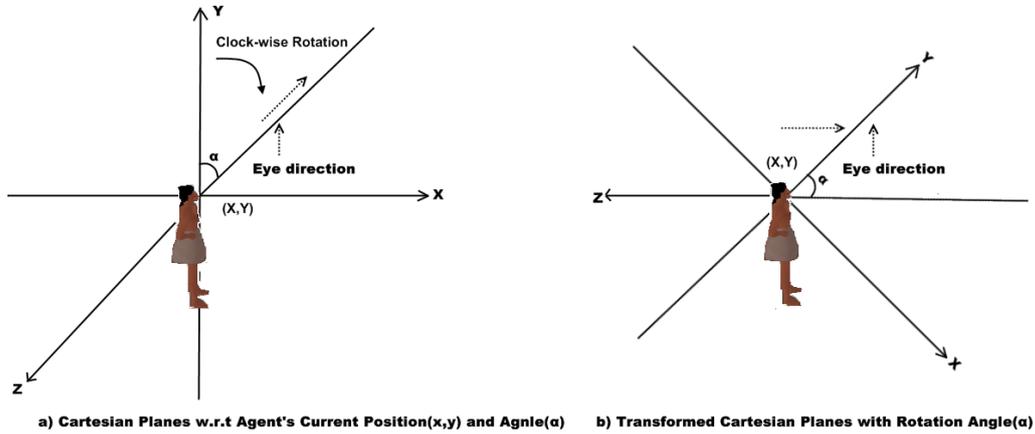


Figure 4: Aligning cartesian planes with agent's orientation.

of their surroundings. Our aim is for an agent to be able to interact with humans and correctly respond to human inquiries about the objects and avatars in the environment. Agents' conversational abilities in our system are supported by the Artificial Intelligence Markup language (AIML) (Wallace 2004a). When a participant in the environment inquires something like: "What is that building on your left?" - we first extract the desired information from an AIML rule. The corresponding rule would be: "What is that *", where "*" represents any number of characters in the remainder of the query. First step of environment awareness process is to tokenize the input. These tokens are matched with the list of directions, objects and relative pronouns as shown in Table 1.

To locate the object(s) of interest - a virtual agent uses the data from the Object Annotation Layer to search for keywords like object name, type, etc as shown in Figure 2. For example: House and direction: "Fisherman house on the left". Moreover, it needs the exact relative pronouns like "You" or "Me" to co-relate the right objects in the space. As the user inquiry could be about an object on either participant's left or virtual agent's left, this hint helps the agent to specify participant's correct direction of interest.

Table 1: Environment Awareness

Environmental Entities List		
Pronouns	Directions	Objects
You/your	In-front	Building(House, Ziggurat, Temple ...)
	Behind	Plant(Tree,Grass ...)
Me/My	Left	Avatar(Agents,Humans)
	Right	Animal(Sheep,Donkey ...)

The agent further needs to locate the specified direction in the context of current space, which is relative to the pronoun given to search the "object" of interest in the virtual environment. The agent knows its current position vector and body orientation with respect to standard cartesian coordinate system. Identification of conversational agent's direction like "left" or "right" requires coordinate transformation in respect to the agent's current position and orientation.

Figure 4 (a) shows the position of embodied virtual agent in current cartesian coordinates space, where the agent currently looking towards the side pointed as eye direction and was away from the Y-plane with angle(α). Defining a direction like "left" or "right" with respect to agent's current orientation requires to

rotate the cartesian planes clockwise or anti-clockwise with angle(α).

Agent's Eye Direction w.r.t Y-plane = Angle of Rotation = alpha - α , where (x, y) are the coordinates representing some point in the current space. In two dimensions, the rotation matrix has the following form:

$$\mathfrak{R}(\alpha) = \begin{vmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{vmatrix} \text{ (rotation by angle } \alpha \text{)}.$$

The new coordinates (x', y') for point (x,y) will be:
 $x' = x * \cos(\alpha) - y * \sin(\alpha)$; $y' = x * \sin(\alpha) + y * \cos(\alpha)$.

To rotate the coordinates anti-clockwise, we need to replace α with $-\alpha$.

$$\mathfrak{R}(-\alpha) = \begin{vmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{vmatrix} \text{ (rotation by angle } -\alpha \text{)}.$$

This transformation of cartesian planes has been demonstrated in Figure 4 (b).

Once the agent has figured out the desired direction e.g. "left" in the current request, it further needs to search all the objects of class "Building" to locate the appropriate object. Each agent in the environment has its vision sphere as presented in Figure 5, which helps it to locate the object of its interest within the vision radius. In the given scenario the agent has found two buildings inside the vision sphere, which are Ziggurat and Temple of Uruk. If multiple objects of same class have been found - the agent would return the list of the objects and request more details. Otherwise, the object would be sensed within the vision sphere and its name is passed to the AIML module. The role of the AIML module is to produce the relevant object description given this name.

Each AIML rule(category) has two main tags to generate a dialog which are *< Pattern >* and the *< Template >* as shown in AIML example below. The Template tag generates a response to the inquiry requested in the Pattern tag. The AIML rule below also shows the *< Environment >* tag, which is a custom tag responsible for invoking our environment awareness routine. The current AIML rule could be divided into two parts, "What is that" - is the general rule to invoke the corresponding template with custom *< Environment >* tag. The second part is represented by a wildcard character "*", which encloses the user input part of the pattern. The *< star/ >* tag substitutes the value matched by "*".

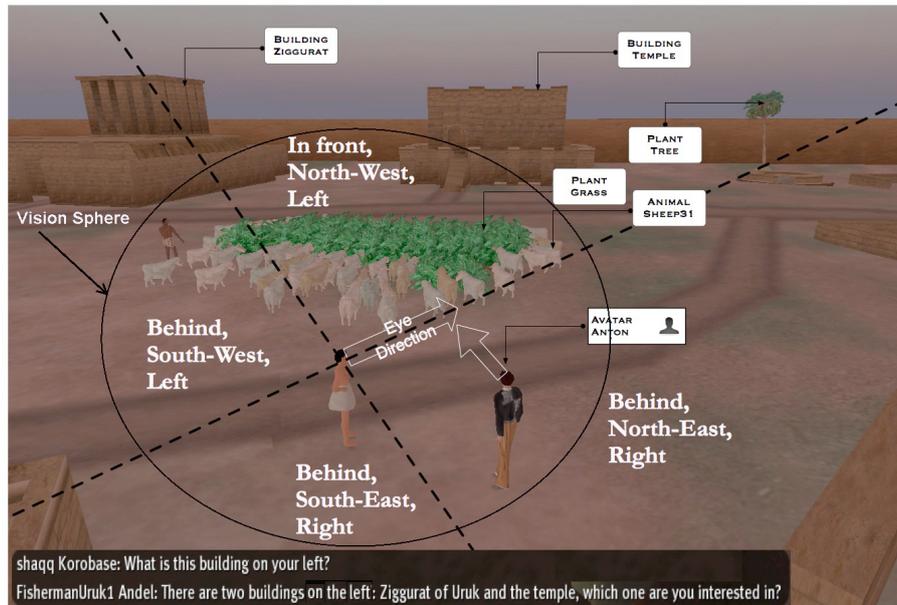


Figure 5: Agent orientation in Uruk.

```

<Category>
<Pattern>What is that * ?</Pattern>
<Template><Environment/><Star/></Template>
</Category>
    
```

Natural Language Processing (NLP) and the pattern matching is done by the AIML engine which also enables the custom tags. Our environment awareness routine is further responsible for completing the query and scanning the desired environmental objects. The OpenMetaverse library interacts with AIML to pass on the information about an agent's surrounding.

4.4 Implementation of Self-Awareness

The self-awareness component of an agent helps it to reflect on its own state in the virtual world, explain the reasons for performing certain actions, using certain objects or walking in a particular direction when a human user inquires about those. Moreover, to be more believable an agent should also be aware of its current goals and plans to give an appropriate response to the human user.

In its reasoning to respond to the query, the agent partially relies on the Virtual Institution (as shown in figure 2) that provides the agent with the corresponding knowledge about the current scene and the agent's state within this scene. The rest of the details the agent either extracts based on its location in the environment (i.e. identifying the objects in the field of view of the avatar, objects being used or animation being played) or from its internal beliefs. Figure 6 exemplifies some parameters of the self-awareness model.

5 Case Study: The City of Uruk, 3000 B.C

Our case study demonstrates the kind of a complex application domain that becomes feasible once we have conversational agents that are environment-, self- and interaction-aware. Further we label them as "aware agents" and those that don't possess such qualities are called "unaware agents".

The case study tackles the domain of history education and aims at recreating the ancient city of Uruk

and reenacting the life in it from the period approximately 3000 B.C. in the virtual world of Second Life, showing the visitors how it looked like and how its residents behaved in the past. This work is based on the prototype developed by our group (Bogdanovych et al. 2010).

Our objective is to provide engaging and interactive learning experience to history students, immersing them in the daily life of Uruk. Through embodied interactions virtual agents will teach various aspects of Uruk history, culture and the daily life of ancient Sumerians. Currently, we have two fisherman families "living" in the Uruk environment and performing their daily routines. Virtual agents' behaviours and social interactions have been institutionalised with the help of the VI technology. Before this work, these virtual agents had their daily routines in place but interactions with other participants were very limited - they could only talk about pre-fed historical facts with the visitors and were not aware of their surroundings while performing daily life routines. Through environment- self- and interaction-awareness the agents now can explain what they are doing, why they are doing it in a certain way, refer to the objects in the environment and even help visitors to perform their assigned tasks and showing which actions must they perform to further progress in their assigned mission. Next we show how we evaluate the impact on believability of the awareness features.

6 Believability Evaluation

There exists no formal definition of believability, nor there are clear methods to measure it. Thus, we have adapted and modified to our needs the approach in (Gorman et al. 2006). The subjective nature of the approach has stimulated another aim of our work - the design a rigorous objective evaluation of believable conversational agents and calculating a believability index as a measure of their human-likeness.

6.1 The Design of Experiment

To test the believability of our agents, we designed an experiment and analysis technique adapting the

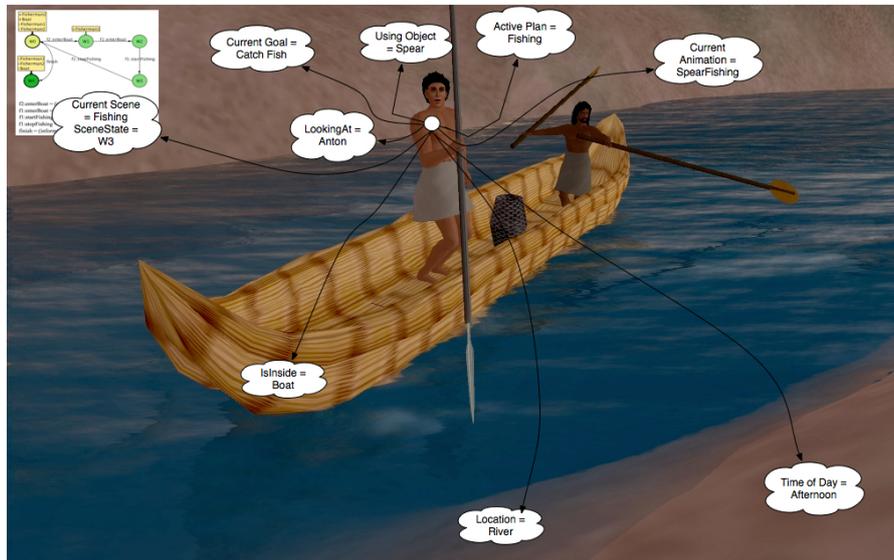


Figure 6: Self-awareness model.

methodology from (Gorman et al. 2006). The study sample included participants of diverse demographics. We provided participants with the background about Uruk and this study. During experiments the participants didn't know whether they had been conversing with avatars controlled by agents or humans.

We had to evaluate the believability of conversational agents, hence, presenting participants only video clips or other test mediums, as performed in (Gorman et al. 2006), was not acceptable due to the issues of biased responses and guess work. To minimise both we ensured that (i) participants interact with our conversational agents in the actual virtual world; and (ii) designer has no control over the routines of the agents with the flow of participants' conversations with them.

We also needed to present participants with highly immersive, engaging and interactive experience, which was essential in this study. From our past experience we have learned that navigation in virtual worlds requires some practice. Hence, the experiments were supervised by a researcher in order to assist participants with interfacing the agents.

Each participant was assisted to enter as an avatar into the city of Uruk. The participant was then requested to converse with two fishermen agents in the city, where our research assistant kept track of the conversation flow, ensuring that some of their questions were related to environment-, self- and interaction-awareness of virtual agents. The assistant navigated the agent herself, while the participant was directing her where to go, whom to talk with and what to ask. This allowed the participant to focus on examining the believability of the agent's dialogue in the context of its environment, actions and the behaviour of other participants in the city. As a result the participant had to assess the believability of each dialogue on the scale: *{1:Definitely Human; 2:Probably Human; 3:Not Sure; 4:Probably Artificial; 5:Definitely Artificial}*.

The rating of agents' conversations and behaviours was later used for calculating the believability index. The participants were requested to repeat this exercise for all parts of their conversations with our agents. They also had to provide specific feedback where they thought an agent's reply was artificial.

To minimise the chances of participants averaging

out believability index (i.e. when a participant rates some virtual agent's responses as "Human" just because she rated few previous responses as "Artificial", we advised participants in the introduction that their rating should be purely based on their perception of the conversational behaviour of respective avatars in the virtual world.

6.2 Measuring Believability

For measuring believability we modified the equations for believability index from (Gorman et al. 2006) to reflect the interactive nature of our experiment, where the questions asked may differ across participants. Such index reflects participant's certainty with which s/he perceived a virtual agent as human-like or artificial. The equation for calculating the believability index for each dialogue is shown below:

$$h_p(c_i) = \frac{|r_p(c_i) - A|}{A - B} \quad (4)$$

where $h_p(c_i)$ is the perception of participant p of correspondence c_i as human-like and $r_p(c_i)$ is the rating of participant p for the same correspondence c_i . A and B represent the "Artificial" and "Human" value of the virtual agent response on the rating scale. Alternatively, $h_p(c_i)$ would be "0" if the respondent identified virtual agent's response as "Artificial" or "1" if s/he identified it as "Human", where all other values represent uncertain choices. The believability index for any participant is the average of his perceptions:

$$b_n = \frac{\sum_{0 < p \leq n} h_p(c_i)}{n} \quad (5)$$

where n is the total number of responses per experiment. The overall believability in virtual agent's conversation B , based on the rating given by m participants is

$$B = \frac{\sum b_n}{m} \quad (6)$$

In a similar fashion, we could also measure the believability of each category for all participants in their conversations.

Table 2: Believability comparison for aware and unaware agents

Category	Unaware Agents	Aware Agents
Believability of environment-awareness	0.22	0.76
Believability of self-awareness	0.26	0.75
Believability of interaction-awareness	0.30	0.77
Overall Believability	0.27	0.76

6.3 Data Collection and Analysis

This believability analysis was conducted based on the conversations with aware and unaware agents - we wanted to evaluate how believable our agents would be if we supplied them with awareness features. The experiment was based on the three awareness believability dimensions we have identified: (1) environment-awareness; (2) self-awareness; and (3) interaction-awareness. Those who lack these features are regarded here as unaware agents. We divided our participants into two different groups (both groups were immersed in our case study). The first group conversed with aware agents, the second group - with unaware agents. The experiments were conducted over a two week period. After cleaning the incomplete responses, the data that was analysed included the responses of 22 participants - 11 per group.

Table 2 shows a summary of the results for comparing aware and unaware agents. The overall believability index for aware agents was 0.76 vs 0.27 for unaware agents. The comparison along our awareness believability dimensions shows that for environment-aware queries in 76% of the cases participants perceived aware agent as human vs only 22% of misclassification of unaware agents as humans. For queries about agent’s own goals, plans, actions etc., aware agents were ranked as human 75% of the times vs 26% of misclassification of unaware agents as humans. In the case of interaction-awareness, aware agents were believed to be humans for 77% cases vs 30% of misclassification of unaware agents as humans. These results indicate that it was relatively difficult for participants to differentiate between aware agents and humans based on their conversations.

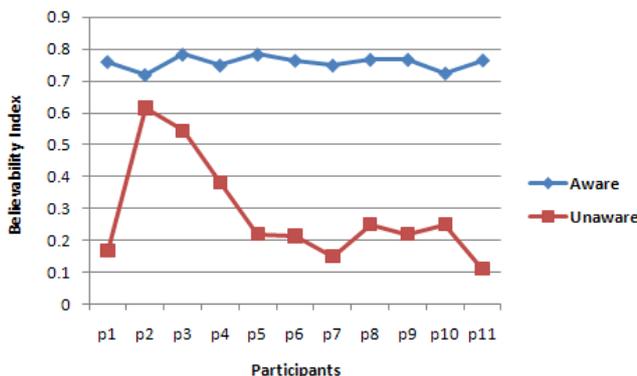


Figure 7: Believability for aware vs unaware agents.

Figure 7 shows the individual believability evaluations, which range within [0.72; 0.78] across participants dealing with aware agents and varied substantially for unaware agents. As our unaware agents were based on AIML rules, on early stages of a conversation they managed to give an impression of a real person, but later in the conversation participants correctly identified them as artificial.

To test believability of aware and unaware

agents’ conversations, the recorded conversations were partitioned into three categories: environment-, interaction- and self-awareness. Figure 8 shows a very high human-likeness (0.6 to 0.8) for each category in our aware agents.

7 Conclusion

This paper has analysed the notion of believability and believability characteristics of embodied conversational agents interacting with a user in a virtual world. The identified features of environment-, self- and interaction-awareness believability have not been addressed in previous works. To show that these features are indeed responsible for improving the overall believability of virtual agents we conducted an experiment where two groups of participants evaluated the believability of two different types of agents: those that incorporate the aforementioned features and those that don’t. The study showed that when embedded, each feature on its own (as well as the combination of all three features) improves the perception of agents as more believable.

References

Bickmore, T. W., Mauer, D. & Brown, T. (2007), Context awareness in mobile relational agents, *in* ‘IVA ’07: Proceedings of the 7th international conference on Intelligent Virtual Agents’, Springer-Verlag, Berlin, Heidelberg, pp. 354–355.

Bogdanovych, A. (2007), Virtual Institutions, PhD thesis, UTS, Sydney, Australia.

Bogdanovych, A., Rodriguez-Aguilar, J. A., Simoff, S. & Cohen, A. (2010), ‘Authentic Interactive Reenactment of Cultural Heritage with 3D Virtual Worlds and Artificial Intelligence’, *Applied Artificial Intelligence* **24**(6), 617–647.

Cunningham, D. W., Kleiner, M., Wallraven, C. & Bülthoff, H. H. (2005), ‘Manipulating video sequences to determine the components of conversational facial expressions’, *ACM Trans. Appl. Percept.* **2**(3), 251–269.

Dautenhahn, K., Ogden, B. & Quick, T. (2003), ‘From embodied to socially embedded agents- implications for interaction-aware robots.’ URL: <http://hdl.handle.net/2299/3057>

Doyle, P. (2002), Believability through context using “knowledge in the world” to create intelligent characters, *in* ‘Proceedings of AAMAS ’02: Conference’, ACM, New York, USA, pp. 342–349.

Elpidorou, A. (2010), ‘Alva noë: Out of our heads: Why you are not your brain, and other lessons from the biology of consciousness’, *Minds Mach.* **20**(1), 155–159.

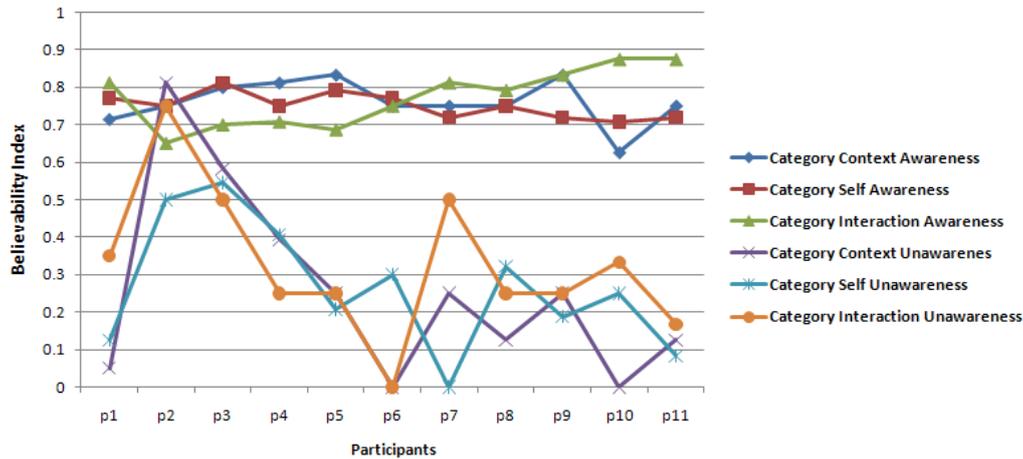


Figure 8: Believability index for each category.

- Esteva, M. (2003), *Electronic Institutions: From Specification to Development*, PhD thesis, Institut d'Investigació en Intel·ligència Artificial (IIIA), Spain.
- Gandhe, S. & Traum, D. (2007), *Creating spoken dialogue characters from corpora without annotations*, in 'Proceedings of Interspeech-07', pp. 2201–2204.
- Gerhard, M., Moore, D. & Hobbs, D. (2004), 'Embodiment and copresence in collaborative interfaces', *Int. J. Hum.-Comput. Stud.* **61**(4), 453–480.
- Gerhard, M., Moore, D. & Hobbs, D. (2005), 'Close encounters of the virtual kind: Agents simulating copresence', *Applied Artificial Intelligence* **19**(3-4), 393–412.
- Gorman, B., Thurau, C., Bauckhage, C. & Humphrys, M. (2006), *Believability Testing and Bayesian Imitation in Interactive Computer Games*, in 'Proceedings of the 9th International Conference on the Simulation of Adaptive Behavior (SAB'06)', Vol. LNAI 4095, Springer, pp. 655–666.
- Hallowell, A. I. (1955), *Culture and Experience*, University of Pennsylvania Press, Philadelphia, USA.
- Hartmann, B., Mancini, M. & Pelachaud, C. (2005), *Implementing expressive gesture synthesis for embodied conversational agents*, in 'Gesture Workshop', pp. 188–199.
- Heylen, D. K. J., van Es, I., Nijholt, A. & van Dijk, E. M. A. G. (2005), *Controlling the gaze of conversational agents*, in J. van Kuppevelt, L. Dybkjaer & N. O. Bernsen, eds, 'Natural, Intelligent and Effective Interaction in Multimodal Dialogue Systems', Kluwer Academic Publishers, pp. 245–262.
- Johnson, W.L., R. J. & Lester, J. (2000), 'Animated pedagogical agents: face-to-face interaction in interactive learning environments', *International Journal of Artificial Intelligence in Education* **11**, 47–78.
- Lee, J., Marsella, S., Traum, D., Gratch, J. & Lance, B. (2007), *The rickel gaze model: A window on the mind of a virtual human*, in C. Pelachaud, J.-C. Martin, E. Andr, G. Chollet, K. Karpouzis & D. Pel, eds, 'Intelligent Virtual Agents', Vol. 4722 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 296–303.
- Lester, J. C., Voerman, J. L., Towns, S. G. & Callaway, C. B. (1999), 'Deictic believability: Coordinating gesture, locomotion, and speech in lifelike pedagogical agents', *Applied Artificial Intelligence* **13**, 383–414.
- Livingstone, D. (2006), 'Turing's test and believable AI in games', *Computer Entertainment* **4**(1), 6.
- Loyall, A. B. (1997), *Believable Agents: Building Interactive Personalities*, PhD thesis, Department of Computer Science, Carnegie Mellon University, Pittsburgh.
- Mateas, M. (1997), *An oz-centric review of interactive drama and believable agents*, Technical report, Carnegie Mellon University, Pittsburgh, PA, USA.
- O'Keefe, J. & Nadel, L. (1978), *The Hippocampus as a cognitive map*, Oxford University Press.
- Pantic, Maja Member, S. & Rothkrantz, L. J. M. (2000), 'Automatic analysis of facial expressions: The state of the art', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(12), 1424–1445.
- Prendinger, H. & Ishizuka, M. (2001), *Social role awareness in animated agents*, in 'AGENTS '01: Proceedings of the fifth international conference on Autonomous agents', ACM, New York, NY, USA, pp. 270–277.
- Riedl, M. O. & Stern, A. (2006), *Failing believably: Toward drama management with autonomous actors in interactive narratives*, in 'TIDSE', pp. 195–206.
- Thiebaux, M., Lance, B. & Marsella, S. (2009), *Real-time expressive gaze animation for virtual humans*, in 'AAMAS (1)', pp. 321–328.
- Thomas, F. & Johnston, O. (1981), *Disney animation: the illusion of life*, 1st ed. edn, Abbeville Press, New York.
- Wallace, R. (2004a), 'The elements of AIML style. AI Foundation.'
- Wallace, R. (2004b), 'The Anatomy of A.L.I.C.E', *A.L.I.C.E. Artificial Intelligence Foundation*.
- Weizenbaum, J. (1966), 'Eliza—a computer program for the study of natural language communication between man and machine', *Communications of the ACM* **9**(1), 36–45.

EvoJava: A Tool for Measuring Evolving Software

Joshua Oosterman¹, Warwick Irwin², Neville Churcher³

Department of Computer Science and Software Engineering
University of Canterbury

Private Bag 4800, Christchurch, New Zealand

jj054@uclive.ac.nz¹, {warwick.irwin², neville.churcher³}@canterbury.ac.nz

Abstract

This paper introduces EVOJAVA, a new tool for extracting static software metrics from a Java source code repository. For each version of a program, EVOJAVA builds a comprehensive model of the semantic features described by Java code (classes, methods, invocations, etc), and tracks the identity of these features as they evolve through sequential versions. This allows traditional software metrics to be recorded over time without losing traceability of software components, and permits calculation of new metrics that characterise the software evolution itself.

Keywords: Software metrics; software evolution; source repository mining; software visualisation.

1 Introduction

Software systems are commonly too large and complex for an individual to fully comprehend. Software product metrics attempt to mitigate the problem by aggregating and abstracting detail in order to expose salient characteristics of programs. Static software product metrics quantify aspects of programs that can be observed in source code (or other static artefacts such as UML diagrams). Examples include measures of size, complexity, coupling, and so on.

Although many software metrics are proposed in the literature, they have seen limited use in the software development industry. Many developers remain sceptical about their claimed benefits (Medha and Carolyn, 2008). Further work is needed to make metrics more useful, and new tools are needed to make measurement easier.

This paper describes EVOJAVA, a new tool that relies on static analysis of Java source code to measure software structure. EVOJAVA builds on earlier tools that could measure a snapshot of source code at some point in time, but which did not specifically support repeated measurement of a program's code as it evolved (Oosterman J. and Churcher, 2010, Irwin, 2007). A particular goal of EVOJAVA is the ability to preserve the identity of semantic software features (classes, methods, statements etc) across versions, despite renaming, moving or refactoring of parts of the code.

By adding the dimension of time to our models, we

hope to enable further research that will clarify the behaviour of metrics over time and lead to improvements in the predictive power of metrics and ultimately increase their usefulness for code comprehension and project management. We would like to investigate, for example, whether source code refactoring – which aims to improve readability, understandability, extensibility or performance of code without modifying its functionality – does in fact lead to improved software quality measurements.

The inclusion of time in our models also enables a family of metrics that quantify aspects of software evolution, such as the lifetime of features or their rate of change. These evolution metrics should help to answer questions such as whether software quality metrics can be used to predict the likelihood that software maintenance will later be required.

Information overload is already a challenge for software engineers, and the inclusion of another dimension of measurement data compounds the problem. Visualisation techniques and software tool support are essential for effectively communicating measurements to developers. A visualisation prototype is included in this paper.

The rest of the report is structured as follows:

Section 2 reviews related technologies in the literature. Section 3 describes the design and implementation of EVOJAVA. Section 4 presents results collected from real world software by the tool, and related visualisations.

2 Background

2.1 Software Metrics

Among the best known metrics for Object-Oriented (OO) programming are the Chidamber and Kemerer (CK) metrics suite (Chidamber and Kemerer, 1994) and the MOOD (and later MOOD2) suite (Abreu and Melo, 1996). Popular CK and MOOD metrics include *Weighted Methods per Class* (WMC), *Depth of Inheritance Tree* (DIT), *Coupling Between Objects* (CBO), and *Attribute Hiding Factor* (MHF).

Many subsequent papers have addressed the CK and MOOD suites, with particular focuses on extending the set or validating their use (Chahal and Singh, 2009, Nagappan et al., 2005). A criticism of these metrics (and others), is that they are underspecified. The few industry applications built to measure metrics differ in the way they compute their values (Lincke et al., 2008). Even counting the number of methods in a class is not trivial; should inherited methods, constructors, or overloaded methods be counted? In order to be reproducible, metrics research must include precise specifications of metrics.

Copyright (c) 2011, Australian Computer Society, Inc. This paper appeared at the Thirty-Fourth Australasian Computer Science Conference (ACSC2011), Perth, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. TBA. M. Reynolds, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

Older complexity metrics such as McCabe's cyclomatic complexity (McCabe, 1976) and NPATH (Nejmeh, 1988) can be run on procedural programs or functions, but remain applicable to OO programs. Both of these metrics measure aspects of a program's control flow graph, an acyclic graph representation of the possible control paths through a program. More nodes or paths in this graph should indicate a higher complexity.

The type of metrics that can be calculated is limited by the richness and completeness of the model used. This is discussed further in Section 2.2.

2.1.1 Automatic OO Design Evaluation

Since the introduction of Object Oriented (OO) programming, many informal OO guidelines have been proposed to increase quality. Examples include OO design heuristics (Riel, 1996) and *Code Smells* (Fowler, 1999). To varying degrees, these guidelines can be measured with quantitative methods using software metrics and rules. For example, the *Large Class Smell* could possibly be detected by counting lines of code, number of methods or number of instance variables. The *Acyclic Dependencies Principle* could be enforced with a topological sort algorithm. Others are impossible to automate, such as Riel's *Model the Real World* heuristic, which requires semantic knowledge of the problem domain.

A system called CODE CRITICK (OOSTERMAN J. AND CHURCHER, 2010) was built in our previous work to automate many of these rules. CRITICK evaluates Java programs, and provides a ranked list of violations. These metrics and rules were built on the JST semantic model (Irwin, 2007), and can be used to quantify aspects of quality in our software evolution research.

2.2 Static Analysis and Semantic Models

Static analysis involves the analysis of software artefacts, such as source code. Most modern Integrated Development Environments (IDEs) analyse source code to provide features such as syntax highlighting, auto completion and automated refactoring. Some compilers such as the Java compiler also provide more complex checking such as dead code detection. Raw source code alone is not suitable for such complex static analysis, so it must be processed first.

A file of source code in its simplest form is simply a string of characters which represents some sentence in a particular language's grammar. A scanner and parser are used to turn this string into a syntactic model, represented as a syntax tree. Metrics such as cyclomatic complexity or NPATH can be calculated from a syntax tree, but others such as coupling and cohesion cannot.

The syntax tree, while useful, does not fully model the high level concepts in OO source code such as packages, classes and relationships. A semantic model of the code can be built from the syntax trees, and this is in essence what a compiler does. Depending on the richness and accuracy of the semantic model, advanced analysis such as relationship metrics and automatic design evaluations can then occur.

We use the Java semantic model provided by JST (Irwin, 2007). The model accurately describes the

relationships between packages, classes and methods in a Java codebase.

2.3 Version Control Systems

The use of a Version Control System (VCS) such as SUBVERSION¹ or MERCURIAL² is common practice in team-based software engineering. These systems enable multiple developers to concurrently read and modify the same code base. The history of source code is maintained in a repository so that developers can revert to an earlier version at any time. Repository storage is usually highly optimized using text compression techniques.

2.4 Source Repository Mining

As the VCS repository provides access to every past version of the source code, it enables retrospective analysis of the evolution of software. The process of extracting and processing information from a repository is known as Source Repository Mining (SRM). It has become a significant area of research in recent years (Robbes, 2007, Wedel et al., 2008). Two of the most popular applications of repository mining are defect prediction (Aversano et al., 2007, Nagappan et al., 2006), and the characterisation of software evolution (Robles et al., 2006). Our tool EVOJAVA builds on technology used in both of these areas, and should subsequently allow us to contribute back to this area, with detailed and accurate data. When mining a repository, it is also possible to extract the metadata about the change between each version change, such as the time, author, or change description (commit message). This can give information such as the size of contributions per developer, or which particular changes fixed a bug.

2.4.1 Defect Prediction

Defect prediction involves using software metrics to find areas of code that are likely to contain defects or bugs. Predictors are built in retrospect, by analysing both the source code and historical defect information. In essence, a predictor takes a collection of metric values from a module and then estimates the number of defects it contains using its internal model and past experience. In order to build an effective predictor, the metrics must be shown to have a strong empirical relationship to defect rate.

Diverse metrics are used in the defect prediction research, but they have common themes of measuring size, quality, and complexity. Metrics from the CK suite have been found to correlate with defect rates, as documented in a summary of the area by (English et al., 2009). Modules with more Lines of Code (LOC) have been strongly correlated to higher defect rates; this is hardly a surprising result.

A large scale study of Microsoft projects such as INTERNET EXPLORER and DIRECTX found that other metrics such as number of functions, fan in, cyclomatic complexity and inheritance depth were correlated to defect rates (Nagappan et al., 2006).

¹ <http://subversion.apache.org/>

² <http://mercurial.selenic.com/>

The EVOJAVA tool will enable us to collect these metrics, to detect correlations not only against defect rate, but refactoring and code rework.

2.4.2 Software Evolution

The other large area of SRM research is the characterisation of software evolution, by means of repository mining (Robles et al., 2006). For example, a tool called SOURCERER was built and then run over 38 million lines of java code to collect many evolution metrics (Bajracharya et al., 2009). Similarly, OHLOH³ is a website which mines repositories of thousands of open source projects to provide information about longevity of projects, and popularity statistics for programming languages. Although the existing work has a large number of projects, the metrics and source code analysis is too superficial for our analysis. These results are perhaps suitable at project manager level, but are not fine-grained enough to perform metrics research at the class, method or line level.

Some researchers have focussed on designing and implementing extensible research frameworks for software evolution, in a similar vein to EVOJAVA. The benefits of such frameworks are that researchers can focus without specific questions without having to invest significant development time on the tools. The ALITHEIA CORE is such an extensible framework designed for software engineering research (Gousios and Spinellis, 2009). In addition to source code repositories, information is extracted from bug tracking systems and email servers. This system is very heavy weight, designed to have a distributed architecture. The KENYON framework (Bevan et al., 2005) was designed for similar reasons, with a focus on fact extraction, fact storage and scalability. Both systems stored rich amounts of metadata and integrated with multiple VCS systems. However, they currently only support evolution measurement at a higher level and plug-ins would need to be developed to accurately collect our target metrics. While relevant, the design goals for EVOJAVA are sufficiently different that we have designed a new system.

2.5 Limitations of a VCS

EVOJAVA was designed to be a general purpose, accurate tool for software evolution metric research. Mining from a VCS repository, as in much previous SRM research, is not the only option for such analysis. In fact, mining from a VCS repository has several downfalls (Robbes, 2007). In this section we discuss these reasons, and argue for our choice to mine a VCS repository.

The primary shortcomings of VCS repositories are that they are file-based and snapshot based. The term ‘file-based’ means that version control systems store information as files and folders the basic building blocks in a file system. While this allows them to store a wide range of information, it means that a significant amount of pre-processing is required to abstract the content into higher level semantic concepts such as classes, methods and relationships. The term snapshot-based refers to the granularity at which information is updated to the

repository. The size of a *changeset* between any two revisions can be arbitrary, and the actual number of code level change actions such as refactorings is unknown. It is not possible given two consecutive revisions (or snapshots) to completely reconstruct the set of actual actions that took place, thus data is lost.

Robbes’ proposed solution was a new technology called a change based repository. This required an IDE plug-in to be used, which would store semantic code level actions in a domain specific repository, at a significantly finer granularity. Although this approach reduced information loss, we still argue for the use of classic VCS repository mining in EVOJAVA for several reasons.

Previous work on static analysis has produced tools for parsing and modelling software systems. We believe that JST is accurate and powerful enough that the file-based aspect of repositories is not a problem.

Also, the implemented change based repository was only a prototype for one language (Smalltalk) and only worked for a specific IDE. This made the assumption that all code modification actions would occur using IDE actions, which is not a universal reality. Real world software may be developed using different tools or environments.

And lastly, VCS repository mining can be done after the fact. Repository mining techniques allow us to analyse the many existing open source projects over multiple years of development.

3 EvoJava

3.1 Requirements

EVOJAVA was designed to meet the following requirements:

1. Integrate with a SUBVERSION repository. The history of code versions is the primary data source for analysis.
2. Construct a temporal semantic model of Java programs. Existing models such as JST describe the static semantic concepts in software; EVOJAVA adds a time dimension.
3. Compare consecutive models and deduce the actual changes that occurred, without losing the identity of software features. This is a necessary prerequisite to building an accurate evolution model from a snapshot based repository.
4. Provide an API that enables configurable calculation of metrics, including traditional software metrics, CODE CRITICK metrics and new evolution metrics.
5. Condense results into comprehensible forms using visualisations.

System performance, in terms of memory usage and processing time, was not a priority, due to the research nature of this project.

3.2 System Architecture

The system was designed in a modular fashion so that it would be extensible and could be used in future work. The main modules of the system directly relate to each requirement identified above: the repository integration module, the evolution model, JSTDIFF, and the metric

³ <http://www.ohloh.net/>

framework. In this section, the system workflow, architecture and interaction between modules are discussed. Important modules are then explained in detail in following sections.

The system requires two inputs items to run, the address of a SUBVERSION repository and an XML query file which specifies the metrics to collect. Once a run has finished, an XML file is produced containing the results. This workflow is that of the XML pipeline (Irwin and Churcher, 2001) – the output could be transformed using XSLT to XML input formats for our existing visualisations. Thus, although the system is internally complex, it can be viewed as a ‘black box’ for collecting evolution data.

The internal architecture of the system is depicted in Figure 1. The grey dotted rectangle represents the EVOJAVA system, and the items outside of this rectangle represent the input or output artefacts discussed above.

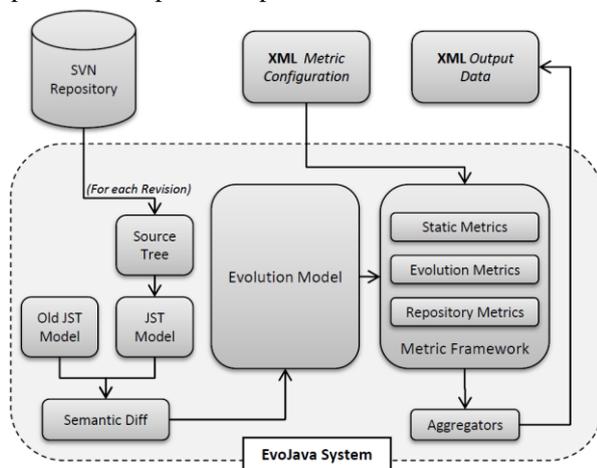


Figure 1: System Architecture

The repository integration module is responsible for several tasks. When a run commences, it will query the repository to fetch a list of version numbers, and branches. It will then incrementally check-out a copy of the code base at each version, allowing the rest of the system to process one version at a time.

The evolution model is a semantic model of a Java code base, modelling the concepts of classes, methods and packages. It is similar to the JST, except that has the dimension of time. The model is updated at each version to reflect the current state of the code base, whilst still remembering critical information from previous versions (such as a historical list of changes).

The metric framework is responsible for querying the evolution model to collect the metrics of interest. When the metrics are collected depends on the category of metric. *PerRevision* metrics, such as the traditional metrics LOC and WMC are run as the model updates to each version. *Evolution* metrics are those run only at the end of processing, such as the *Lifespan* or *ModificationRate* of a java method.

JSTDIFF is the last module of the system. JSTDIFF uses a set of algorithms and heuristics to determine the semantic changes that have occurred between two consecutive code versions. At each version, it used to calculate the *semantic diff*, which is used to update the

evolution model, whilst still preserving identity to modified classes, methods and packages.

The JST is still the backbone to the system. It is richer that the evolution model so it is used by many of the *PerRevision* metrics. It is also used to parse each version of the code base, and build the semantic model required by JSTDIFF.

3.3 Subversion Integration

The EVOJAVA system interacts with a SUBVERSION repository in order to extract data. We chose to support SUBVERSION for the following several reasons. Firstly, SUBVERSION is centralised, so history of versions is stored on a single server. It would be harder to mine a distributed system as there can be no identifiable ‘canon’ repository. Secondly, SUBVERSION is one of the most popular systems, used by open source giants such as the Apache Software Foundation⁴, and GCC⁵. Finally, the University of Canterbury Software Engineering department uses SUBVERSION. This allows us to mine our own projects, and apply context to the collected data. Although we currently only support SUBVERSION, generalising EVOJAVA at a later time should not prove difficult.

An open source, pure Java SUBVERSION library called SVNKIT was used in EVOJAVA as it provides an API for all of the tasks required.

Unfortunately, the code history is complicated by branching and merging in the repository. SUBVERSION allows you to create a branch, which is a clone of the entire code base, with its own parallel history. Typically this is used in order to create an unstable feature branch, or a stable release branch, which can be modified without affecting the main branch, usually known as *trunk*. The opposite, applying the changes from the history on one branch to another branch, is known as merging. Due to branching and merging, there may be many parallel histories for any code object, and not a single linear history.

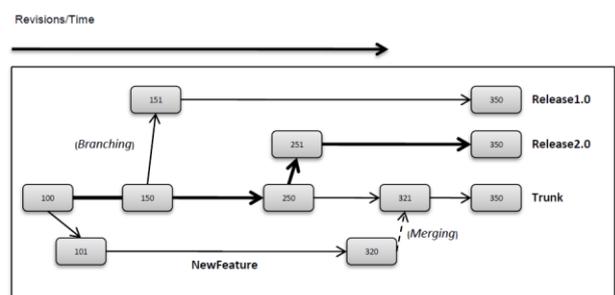


Figure 2: SVN Branching

EVOJAVA addresses this problem by automatically finding the logical path through the version tree from a user-specified endpoint of interest. It will backtrack through the tree, finding the branching revisions, and locate the logical origin of the code. Figure 2 depicts a nonlinear history, and the path of revisions used for analysis, given the endpoint *Release2.0*. Until recently,

⁴ <http://www.apache.org/>

⁵ <http://gcc.gnu.org/>

SUBVERSION would not keep record of merges in the repository. Consequently, when a set of changes from a branch is merged back into the trunk it will appear to EVOJAVA as a single large set of changes. This is largely unavoidable.

3.4 Evolution Model

The evolution model is effectively a lightweight version of the JST which models containment relationships between Java feature nodes such as packages, classes, interfaces and methods. However, each node in the model stores evolution information such as creation and deletion revisions as well as a history of changes. It does not aim to fully replace the rich JST model, but rather augment it with the time dimension.

Both JST and the evolution model are queried in the same fashion. The metrics are able to visit the models using the Visitor design pattern [GoF]. This allows the metrics to touch the nodes of interest, and then call their methods to collect data.

Each node in evolution model supports the concept of identity, even if its name or content changes drastically throughout its lifespan. Nodes are recognized using a unique id, built from the fully qualified Java name at the first version it appeared in.

3.5 JSTDiff

JSTDIFF is perhaps the most complex module in EVOJAVA. Its purpose is to determine the semantic changes that have occurred between two consecutive code versions. It is based on the existing UMLDIFF system, which required a custom semantic model, rather than the JST.

It is important to distinguish a semantic diff from a text diff. A text diff is produced by the VCS to display the text-level changes that have occurred between two versions, in terms of line and file additions and removals. The text diff fails to capture higher level Java semantic concepts. What is identified as a line addition in a text diff, could semantically actually be the addition of a field to a class, or statement to a method body.

JSTDIFF works by walking two JST models simultaneously, comparing the 'before' and 'after' nodes. The difficult part is determining which nodes in each are actually the same, despite having been renamed or modified between versions. When matching nodes, JSTDIFF matches nodes with the following priorities:

1. Completely Identical
2. Different, but have the same name⁶. This corresponds to a *Modified* change.
3. Structurally Similar, different name. This corresponds to a *Renamed* change.
4. Unresolved. The remaining nodes are marked as *additions* and *deletions*.

Several similarity heuristics are used to determine matches. Priorities 1 and 2 use a simple text similarity heuristic to determine name similarity.

Priority 3 uses two heuristics. The first one is the percentage of identical or nearly identical lines in text body. The other one compares the similarity of the relationship sets of the two nodes. The set includes object names, and their relationship type to the node, such as containment, invocation, declaration, dependency etc. The maximum value from these two heuristics is used, for stability against different types of changes.

Once two nodes are matched, the actual changes are determined by comparing the attributes on the before and after node. The changes are combined into a tree model, which is then applied to the evolution model.

3.6 Metrics

Different categories of metrics can be collected using the metric framework in EVOJAVA.

The first category is the *PerRevision* metrics. These metrics run only on a snapshot, and don't account for evolution themselves. However, as they are run on each version, they can be used to describe evolution. The CK and MOOD OO metrics, as well as the CODE CRITICK system, are all *PerRevision* metrics.

The second category is the *Evolution* metrics, which utilise the EVOJAVA evolution model. *Evolution* metrics tell you something about the life span of an object, such as how frequently it was modified.

The last category is *Repository* metrics. These are enabled in EVOJAVA through the SVNKIT API, and could include metrics such as commit frequency and developer contribution size.

4 Results

We have used the EVOJAVA tool to gather data from the repositories of real world software projects. In this section we discuss the preliminary findings, to demonstrate the power and utility of the EVOJAVA tool.

At the University of Canterbury, 3rd year students can elect to take a year-long course, in which they must develop a large scale, real-world software project. The class usually consists of about 6 teams, each containing 6 students. We chose to use the 2010 student projects as a pilot study for our tool, for several reasons:

- The Scale of the projects is suitable for a pilot study. Most have between 200-500 revisions, and about 5,000 LOC.
- The teams are building individual projects, but to solve the same problem. This allows comparisons between repositories.
- Feedback on the development habits of students is useful for the department.
- The developers of the project are easily accessible, which is of mutual benefit to us and the students. Students are interested in the data collected by our tool, and in return we are able to query them about any irregularities in the data collected, to evaluate the tool and refine it.

As mentioned previously, EVOJAVA fits into the XML pipeline. XSLT transformations have been written to transform the XML output into CSV files, which were then graphed using Excel for this report.

⁶ For example, it is a very unlikely a class is deleted and another is added immediately with the same name.

4.1 Evolution Overview

Traditional ‘static’ metrics such as LOC, and the CK suite, can be run automatically on each previous version of the code. This is one of the features of EVOJAVA.

Figure 3 displays the total LOC of a group’s project, over the first 550 revisions. Unsurprisingly, the graph is gradually trending upward, eventually reaching about 15,000. Gathering this data is a trivial task, but it is critical for deeper analysis, thus EVOJAVA is able to collect it. Graphs of this sort are common in both the literature and existing tools such as OHLOH.

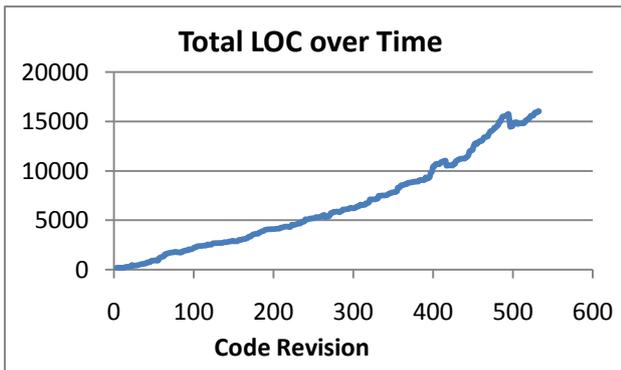


Figure 3: LOC over Time

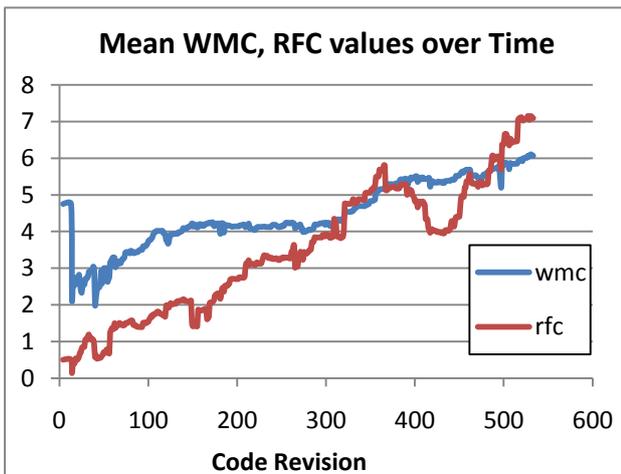


Figure 4: WMC, RFC over Time

Figure 4 shows some more detailed information. Weighted Methods per Class (WMC) and Response for Class (RFC) are two metrics in the CK OO Suite. These

metrics must be measured on a per-class basis, and require a strong semantic model, such as the JST. This graph shows that on average, the number of methods per class, and the number of methods called per method, are both increasing over time.

4.2 Code Critick

The CODE CRITICK System developed in our previous work is now part of the EVOJAVA system, so we are able to characterise software evolution in regards to OO design.

CRITICK returns a ranked list of Violations to OO design rules and heuristics, which are implemented using various metrics and algorithms. Figure 6 displays the quantity of violations found against several CRITICK rules, over a project’s lifetime. Note that this graph is normalised for project size, and the Y axis represents violations per 1,000 lines of code.

The presence of violations is common, and largely unavoidable, due to the nature of conflicting forces in OO design. Encapsulation related rules can also be very conflicting, and tend to make up the majority of violations found in student systems. For this graph, several rules were removed.

The results for the first few revisions are bound to be noisy due to the small, volatile nature of a project at this point.

The overall violation trends for other groups’ projects were quite different to the one depicted above. In particular, another group frequently broke the *LargeClassSmell* and *LongParameterListSmell*, rules, but had no *SwitchStatementSmells* at all.

4.3 Change Metrics

The results discussed so far only measure software metrics against individual snapshots.

The real power of the EVOJAVA system is that it contains an evolution model, and preserves the identity of semantic elements (such as classes, methods, and packages) between versions, for a richer evolution analysis.

In addition to tracking semantic elements, between versions, the JSTDIFF subsystem of EVOJAVA serves to characterise the actual semantic changes that occur during an elements lifetime. Figure 5 shows the number of each detected type of semantic change, over the lifespan of a group’s project. The revisions have been grouped into 5 100 revision buckets, and the quantity of each change type is shown as a bar.

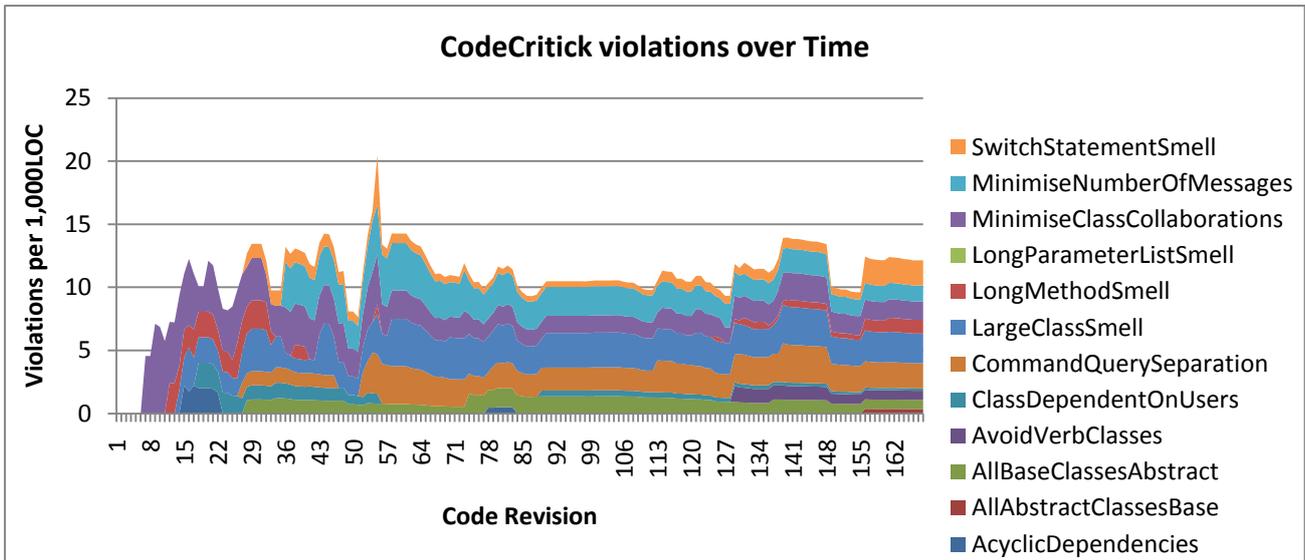


Figure 6: CodeCritic over Time

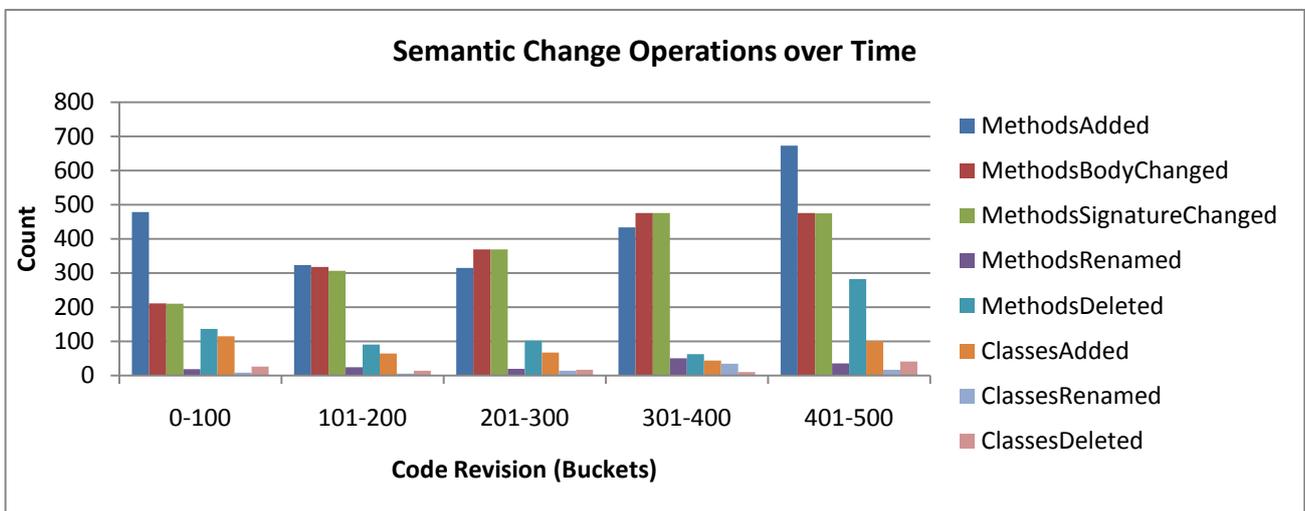


Figure 5: Change Types over Time

In the first and last buckets, the primary type change operation identified is *MethodAdded*. The change type *MethodsBodyChanged* is detected when one or more lines of code in the method have been added, removed, or modified, as detected by a SVN textual diff. *MethodsSignatureChanged* is detected when the name, visibility, return type or parameters of a method are changed. These change types are two of the most common in each bucket, as can be seen by the red and green bars in the graph.

In all buckets, the number of methods added outweighs the methods deleted, and the same is true for classes added and classes deleted. This is sensibly linked to Figure 3, where we see a steady increase in LOC for the same project.

In the last bucket, there were significantly more methods added *and* deleted. This suggests that significant code rework occurred at this point in development. This likely explains the apparent dip in RFC displayed between on Figure 4, during the same revision period.

Another interesting point is that throughout the entire lifespan of the project, relatively few rename operations were detected.

4.4 The Interactive heat map Visualisation

Although overview aggregate measures are useful for characterising software evolution, EVOJAVA is able to measure metrics, and track elements at a much finer granularity. Displaying this amount of information presents information overload issues, which need to be addressed. The interactive heat map, as depicted in Figure 7, is able to display such detailed information in a compact space. It is loosely based on CVSCAN (VOINEA ET AL., 2005).

Each row in the display represents an element in the system, depending on the selected granularity (in this case, each row is a method). Each column in the display represents a revision of the code, as extracted from Subversion. The shaded (non-white) cells in a row represent the versions that the particular element was present in. For example, in the first revision 3 methods

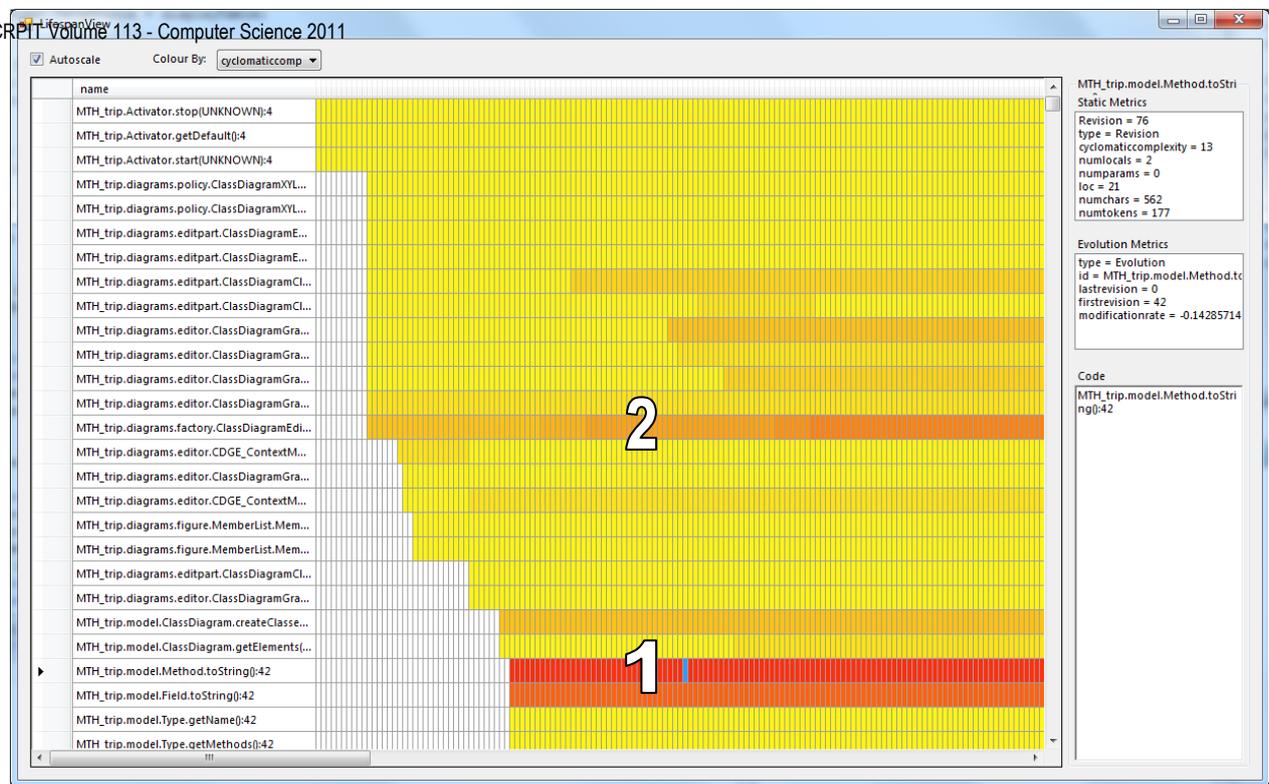


Figure 7: The Interactive heat map

were present. Several revisions later, 11 further methods were added.

The actual colour of each shaded cell represents the heat, or relative metric value, of that element at that particular version. For example, this heat map is shading cells based on the *CyclomaticComplexity* metric. The row marked 1 represents a method that has is theoretically quite complex. The method shown as the row marked 2 was mildly complex when it was added, but was modified several times subsequently to become more complex.

The heat map is interactive, as it allows users to select any cell in the display to view more information. Selecting a cell will display all of the available metric values for the selected element (row) at the selected version (column) in a text box at the top right.

If available, the source code is retrieved and displayed in a textbox to the bottom right.

4.5 Performance Considerations

As mentioned previously, EVOJAVA is primarily a research tool, correctness was valued ahead of system performance, and thus it was only identified as a secondary goal.

For a project with 200 revisions, and 5,000LOC, analysis took under 20 minutes on a modern desktop computer (2.8Ghz Quad-core, with 4GB DDR3 RAM). The same computer processed a 600 revision, 15,000LOC project in just under 3 hours.

The memory footprint of the system is low, as it processes versions incrementally, rather than in parallel. This allows analysis to occur comfortably on a regular desktop computer.

4.6 Future Work

The EVOJAVA system will be extended in our future work to accommodate these features:

1. Extension of JSTDIFF to detect of composite semantic change operations, such as refactorings.
2. Performance enhancements to the underlying JST model, for faster analysis
3. Potential integration with our Process Metrics plugins, to collect code change information at an even finer level.

In addition to these tool features, the system will be used for more in-depth software evolution analysis.

5 Conclusion

We have presented EVOJAVA, a new tool for extracting static software metrics from a Java source code repository. For each version of a program, EVOJAVA builds a comprehensive model of the semantic features described by Java code (classes, methods, invocations, etc), and tracks the identity of these features as they evolve through versions, using the novel JSTDIFF system.

We presented and discussed results collected from real world software projects, developed by student teams at the University of Canterbury. Traditional metrics, OO design metrics, and change metrics were all collected with the tool and discussed.

Finally, we presented a software evolution visualisation called the interactive heat map, and mention our future directions.

6 References

- Abreu, F. B. E. & Melo, W. (1996): Evaluating the Impact of Object-Oriented Design on Software Quality. *METRICS '96: Proceedings of the 3rd International Symposium on Software Metrics*. Washington, DC, USA: IEEE Computer Society.
- Aversano, L., Cerulo, L. & Del Grosso, C. (2007): Learning from bug-introducing changes to prevent fault prone code. *IWPSE '07: Ninth*

- international workshop on Principles of software evolution*. Dubrovnik, Croatia: ACM.
- Bajracharya, S., Ossher, J. & Cristina Lopes (2009): Sourcerer: An internet-scale software repository. *SUITE '09: Proceedings of the 2009 ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation*. Washington, DC, USA: IEEE Computer Society.
- Bevan, J., Whitehead, E. J. J., Kim, S. & Godfrey, M. (Year): Facilitating software evolution research with kenyon. In: *ESEC/FSE-13: Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering*, 2005 Lisbon, Portugal. ACM, 177-186.
- Chahal, K. K. & Singh, H. (2009): Metrics to study symptoms of bad software designs. *SIGSOFT Softw. Eng. Notes*, 34, 1-4.
- Chidamber, S. R. & Kemerer, C. F. (1994): A Metrics Suite for Object Oriented Design. *IEEE Trans. Softw. Eng.*, 20, 476-493.
- English, M., Exton, C., Rigon, I. & Cleary, B. (2009): Fault detection and prediction in an open-source software project. *PROMISE '09: Proceedings of the 5th International Conference on Predictor Models in Software Engineering*. Vancouver, British Columbia, Canada: ACM.
- Fowler, M. (1999): *Refactoring: Improving the Design of Existing Code*, Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc.
- Gousios, G. & Spinellis, D. (2009): Alitheia Core: An extensible software quality monitoring platform. *ICSE '09: Proceedings of the 31st International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society.
- Irwin, W. (2007): *Understanding and Improving Object-Orientated Software Through Static Software Analysis*.
- Irwin, W. & Churcher, N. (Year): XML in the visualisation pipeline. In, 2001. 67.
- Lincke, R. U., Lundberg, J. & L\Owe, W. (2008): Comparing software metrics tools. *ISSTA '08: Proceedings of the 2008 international symposium on Software testing and analysis*. Seattle, WA, USA: ACM.
- Mccabe, T. J. (1976): A complexity measure. *ICSE '76: Proceedings of the 2nd international conference on Software engineering*. San Francisco, California, United States: IEEE Computer Society Press.
- Medha, U. & Carolyn, S. (2008): Why do programmers avoid metrics? *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. Kaiserslautern, Germany: ACM.
- Nagappan, N., Ball, T. & Zeller, A. (2006): Mining metrics to predict component failures. *ICSE '06: Proceedings of the 28th international conference on Software engineering*. Shanghai, China: ACM.
- Nagappan, N., Williams, L., Vouk, M. & Osborne, J. (2005): Early estimation of software quality using in-process testing metrics: a controlled case study. *3-WoSQ: Proceedings of the third workshop on Software quality*. St. Louis, Missouri: ACM.
- Nejmeh, B. A. (1988): NPATH: a measure of execution path complexity and its applications. *Commun. ACM*, 31, 188-200.
- Oosterman J., I. W. & Churcher, N. (2010): Code Critick: Using Metrics to Inform Design. *ASWEC '10*. University of Canterbury, Christchurch, New Zealand.
- Riel, A. J. (1996): *Object-Oriented Design Heuristics*, Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc.
- Robbes, R. (2007): Mining a Change-Based Software Repository. *MSR '07: Proceedings of the Fourth International Workshop on Mining Software Repositories*. Washington, DC, USA: IEEE Computer Society.
- Robles, G., Gonzalez-Barahona, J. M., Michlmayr, M. & Amor, J. J. (2006): Mining large software compilations over time: another perspective of software evolution. *MSR '06: Proceedings of the 2006 international workshop on Mining software repositories*. Shanghai, China: ACM.
- Voinea, L., Telea, A. & Van Wijk, J. J. (2005): CVSScan: visualization of code evolution. *SoftVis '05: Proceedings of the 2005 ACM symposium on Software visualization*. St. Louis, Missouri: ACM.
- Wedel, M., Jensen, U. & G\Ohner, P. (2008): Mining software code repositories and bug databases using survival analysis models. *ESEM '08: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. Kaiserslautern, Germany: ACM.

Dynamic Visualisation of Software State

James Ashford

Neville Churcher

Warwick Irwin

Department of Computer Science and Software Engineering
 University of Canterbury,
 Private Bag 4800, Christchurch 8140, New Zealand,
 Email: jra82@uclive.ac.nz, neville.churcher@canterbury.ac.nz
warwick.irwin@canterbury.ac.nz

Abstract

The size and complexity of software systems presents many challenges to developers. Software visualisation techniques can help make more manageable tasks across the development cycle. In this paper we focus on the dynamic visualisation of software state—an important element in supporting activities such as debugging. We propose Pseudo-Breakpoints as a means of collecting data and making it available to specialised visualisation components. We describe the implementation of this concept in Eclipse and present some example visualisations.

Keywords: Software visualisation, information visualisation, debugging.

1 Introduction

Software engineering, described by Parnas as the “multi-person construction of multi-version programs,” (Parnas 1975) is a dynamic and challenging discipline.

Despite numerous advances in areas such as programming languages, design techniques, tool support and process models developers have struggled to keep pace with the ever-increasing size and complexity of software systems.

The consequences are reflected in the figures for the number of software projects which either fail or are significantly compromised.

Although software engineering is a collaborative undertaking in which individuals play a variety of rôles, the bulk of the “real” work continues to be carried out by individual developers. These people will typically be using individual tools, with IDEs such as Eclipse (<http://www.eclipse.org>) being the norm, and collaboration being managed at the resource level by version control tools such as subversion (<http://subversion.apache.org>).

Development ultimately consists of individuals pitting themselves against a range of tasks including elements of design, comprehension, implementation, fault diagnosis and repair, and testing.

Many of the techniques used by individual software engineers (such as UML, OO design patterns, code smells, complexity metrics, ...) are essentially static. They are applied in the context of a snapshot

of the evolving software artifacts. Such techniques are among the most effective available.

However, there are a number of vital development activities where dynamic techniques have much to offer—either on their own or in combination with static techniques. Examples include:

algorithm visualisation: Understanding and communicating algorithms by enacting them in a way which highlights visually the relevant elements has a long history, dating from systems such as Balsa (Brown & Sedgewick 1984) and Tango (Stasko 1990).

profiling: Implementing a “clean” design and subsequently observing and gathering data about performance issues is likely to direct refactoring effort to productive areas.

debugging: The ability to suspend a running program at “interesting” places and examine its evolving state is a powerful tool for identifying faults.

testing: Intrinsically dynamic activities such as enacting scenarios, use cases or performing acceptance tests occur in software processes.

Information visualisation (Spence 2001, Ware 2004) involves the display of information via computed geometry (a simple example is the representation of a class by a rectangular glyph in UML). Unlike scientific visualisation (electric field strength, temperature profiles, ...), there is no “real” geometry. The computed geometry provides an anchor or metaphor for the user. Software visualisation is the application of information visualisation techniques to problems in the software engineering domain and has been used to address a wide range of issues in software engineering (Eades & Zhang 1996, Stasko et al. 1998).

Effective software visualisation design involves the identification of tasks in the problem domain, the selection of relevant domain data items, and the processing and presentation of derived *information* (Churcher & Irwin 2005). In our previous work, we have applied this approach primarily to visualisations of static software properties (Irwin & Churcher 2003, Churcher et al. 2003, 2007, Irwin et al. 2005, Neate et al. 2006, Harward et al. 2010).

In this paper, we explore the extension of our ideas to the visualisation of dynamic software state. There are two main contributions. Firstly, we propose Pseudo-Breakpoints. The breakpoint is a familiar concept in debuggers: a user-specified point at which execution may be suspended while the user gathers data by exploring the current program state. The Pseudo-Breakpoint is essentially similar, with the

Copyright ©2011, Australian Computer Society, Inc. This paper appeared at the 32nd Australasian Computer Science Conference (ACSC 2011), Perth, Australia, January 2011. Conference in Research and Practice in Information Technology (CRPIT), Vol. 113, Mark Reynolds, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.

exception that execution need not be interrupted—instead the specified information is included in a range of visualisations. Secondly, we propose a range of visualisations which will assist developers with tasks requiring comprehension of dynamic software state. We have implemented Pseudo-Breakpoints and a number of visualisations in Eclipse.

The remainder of the paper is structured as follows. In the next section, we discuss tasks which developers must perform and their information needs. In section 3 we present the Pseudo-Breakpoint concept and describe its implementation in Eclipse. Section 4 includes a number of visualisations and examples of their application. Finally, we present our conclusions and discuss ongoing work.

2 Tasks and issues

For simplicity, we will use the term “debugging” to refer to a wide range of activities which require a developer to acquire, comprehend, maintain, analyse and react to information about the state and behaviour of a running program.

Debugging is challenging, and remains so despite the advent of modern OO languages (Java, C++, C#, ...) and IDEs (Eclipse, Visual Studio, ...). Multi-threaded software, distributed applications and huge APIs are just three of a range of complicating factors which were not commonly faced by C programmers in the 1980s but which have become major issues today.

Carrying out activities such as those mentioned above requires a developer to cope with potentially very large data sets. There are likely not only to be a large number of data items involved but also to be complex interrelationships. This leads to the problem of information overload—which information visualisation aims to address by employing mechanisms for structuring and exploring related data items.

The tools we use are themselves very complex and have the potential to contribute to the information overload experienced during debugging. Figure 1 shows the debug perspective of Eclipse in use. Like its analogues in other IDEs, the display features numerous components, each with a specific purpose. Some relate to threads, some to the location of breakpoints, some to the state at the current execution point, some to corresponding source code and so on. It is common for holoprasting mechanisms to be employed to allow data structures to be expanded as desired. While this is an effective technique, one can have too much of a good thing—the state of the interface in Figure 1 shows several structures expanded in this way. The user can be given the feeling of “You are in a maze of twisty little passages, all alike” in such cases.

In order to solve a problem or complete a task, the developer needs access to a (potentially very large) set of data. However, at the same time, it is also necessary to have a “holistic” overview of the larger-scale situation. This is known as the *focus+detail* problem in information visualisation (Spence 2001, Ware 2004).

Ways of addressing this problem include distortion-oriented (fisheye view) techniques (Furnas 1986, Sarkar & Brown 1994, Leung & Apperley 1994) in which the degree of interest (DOI) of each data item is determined and used to determine quantities such as the amount of the display or degree of highlighting with which to be presented to the user. We explore the application of this idea in our present work.

The mechanics of debugging have not changed substantially for decades, though the tools themselves

are much more sophisticated. Users set breakpoints at “interesting” locations. When program execution is suspended at a breakpoint, information about the program state is gathered: this may involve examining the contents of heap and stack in varying degrees of detail.

Effective debugging involves developers selecting appropriate strategies for isolating problems, identifying their causes and taking corrective action. The process is often initiated by a specific symptom such as a `NullPointerException`. Isolating this may involve activities such as stepping through nested control structures (in case the cause is an initialisation error), following a call chain or considering concurrent thread issues.

In order to support people engaged in such tasks we need an understanding of the various activities and sequences of steps together with their information needs. To do this exhaustively would require extensive empirical work (such as that of Vans et al (Vans et al. 1999)) and the results would reflect a range of individual styles and available technologies. There is a considerable amount of experience-based advice (e.g. http://home.earthlink.net/~patricia_shanahan/debug) about debugging strategies. There are also interesting overlaps with more general areas such as software comprehension as well as more specific ones such as debugging strategies—an example is the work of Lawrance et al. (2007).

However, it is sufficient for our purposes to consider the following (incomplete and unordered) list:

Location: Where am I? This question may be answered in varying degrees of detail—including at this breakpoint, in this scope/block, in this method, at this node, in this collection. Location also has an aspect of “*When* am I?” which might be conveyed by an iteration number or recursion depth.

History: Related questions include “How did I get here?” and “Have I been here before?” Answering these may involve referring to branching (for invocations or jumps) or to iteration controls. A question like “Where have I been?” can be difficult for a conventional debugger to answer fully.

Activity: What’s happening? This might involve the most recent change to a variable in scope, relative frequency of changes to individual variables, activity within collections and so on.

Analysis: Where is the time going? What patterns are evident in the updates to this collection? Addressing questions such as these may require multiple information sources and involve exploration of various parts of the software state and history.

Conventional breakpoints answer some of these (such as corresponding source code location) precisely and associated debugger features such as watch lists indicate aspects of activity. Others are much less accessible via conventional displays.

Additionally, using conventional breakpoints can lead to a somewhat “stop-start” mode of operation. The user is required to make significant cognitive jumps as the context changes from one breakpoint to another, or even to another occurrence of the same breakpoint.

Our approach is based on the concept of using visualisation to obtain a more holistic view of the software state and then to support a focus on more specific elements or information. Our system complements the conventional breakpoint approach and can be used alongside it as desired.

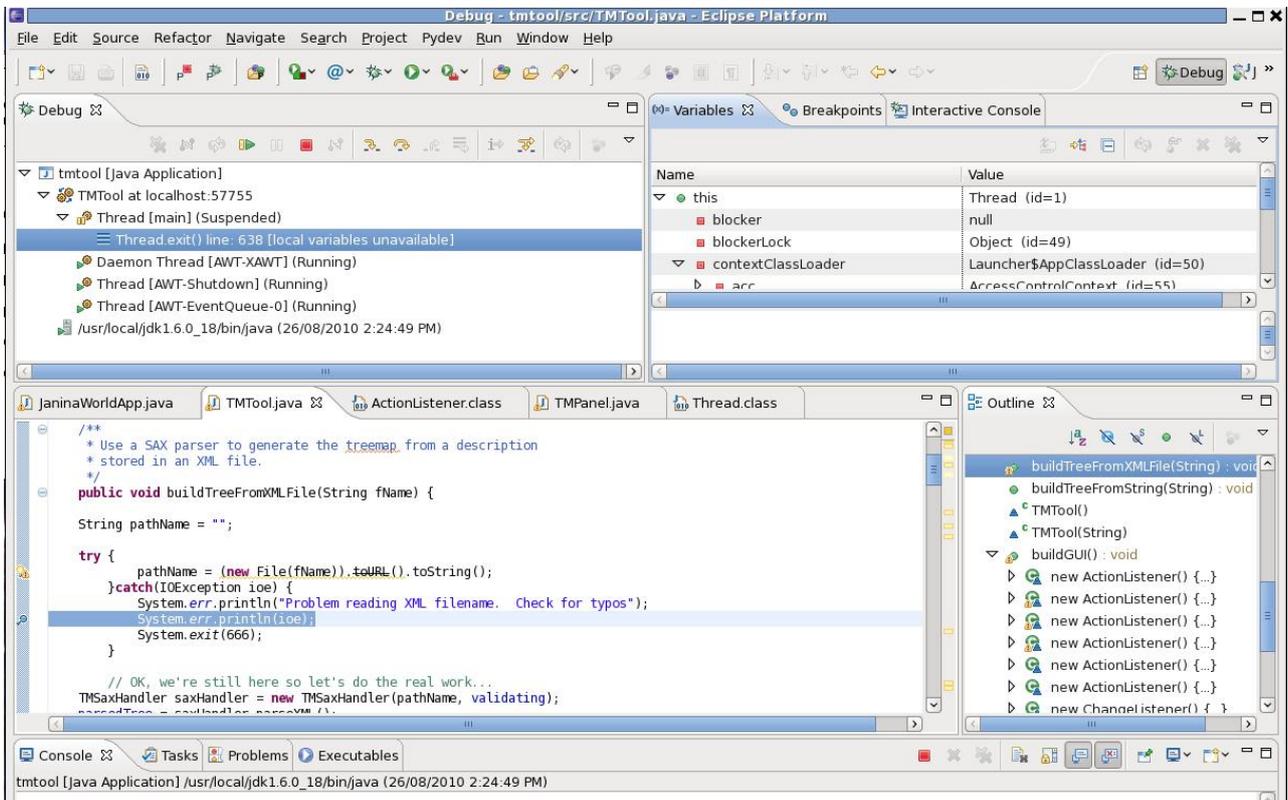


Figure 1: Eclipse debugger perspective

3 PseudoBreakpoints

Much work has been done on support for debugging and other activities requiring access to dynamic software state information and there is a considerable body of research literature on adding visual elements. However, some of these are stand-alone research prototypes, teaching tools or constrained in some way. We have chosen to implement our system as an Eclipse plug-in because we believe it is essential to be able to study the resulting tools in “real” user environments. Similarly, visualisation sometimes appears to be an after-thought—being provided as something to do with the data rather than being an integral element in the overall system design. Our approach involves identifying tasks, together their information needs, and designing visualisations, together with appropriate data capture facilities, to support them.

Mehner (2002) developed a UML-based visualisation similar to the augmented UML class diagrams we have created. It runs as a separate application outside of any IDE and appears to be designed more as a teaching tool than a tools for practitioners. Another stand-alone application sitting outside an IDE is an extensible sequence diagram generator based on execution traces (McGavin et al. 2006).

Linnberg et al. (2004) developed a prototype debugging tool (MVT) which allows for the visual debugging of software. MVT uses the JDI (Java Debug Interface) — again as a stand-alone application.

Jacobs & Musial (2003) make use of the DOI concept in order to increase significantly the number of classes which can be displayed in their UML-based visual debugging application.

Czyz & Jayaraman (2007) is another system which uses the Eclipse interface to generate UML diagrams within the IDE. While their work involves recording

some state information, it does not address the range of aggregation and integrated visualisation techniques that we attempt.

In order to fulfil our visualisation goals we need a means of obtaining relevant data and directing it to appropriate GUI elements. Our initial vision was essentially a “watch list” including the entire system but this would be impracticable on systems of realistic size.

The Pseudo-Breakpoint concept represents what seems to us to be the best of both worlds: it allows specific locations to be selected and de-selected in the same way as conventional breakpoints but does not require execution to be suspended in order to obtain information.

Using Eclipse extensions, we have created a new breakpoint type, PseudoBreakpoint, which extends `IBreakpoint` in the Eclipse breakpoint model. The standard Eclipse editor has been extended to include a new annotation (the red dot to indicate where a pseudo breakpoint is located) and menu item to allow the user to toggle the new breakpoint on and off. In this way, Pseudo-Breakpoints appear to the user in a natural manner for Eclipse users.

When a new Pseudo-Breakpoint is added, the `PseudoBreakpointCreator` class determines which breakpoint should be created. Two types, `PseudoBreakpointJava` and `PseudoBreakpointPython`, are defined in the example shown in Figure 2. Each class contains language-specific breakpoint code (`PseudoBreakpointJava` actually extends the normal `JavaBreakpoint` to provide this functionality).

Data collection occurs when the program is run in debug mode. Our Pseudo-Breakpoints are inserted into the virtual machine in exactly the same way as ordinary breakpoints.

We have created an event handler to capture every Eclipse Debug event — including whenever a break-

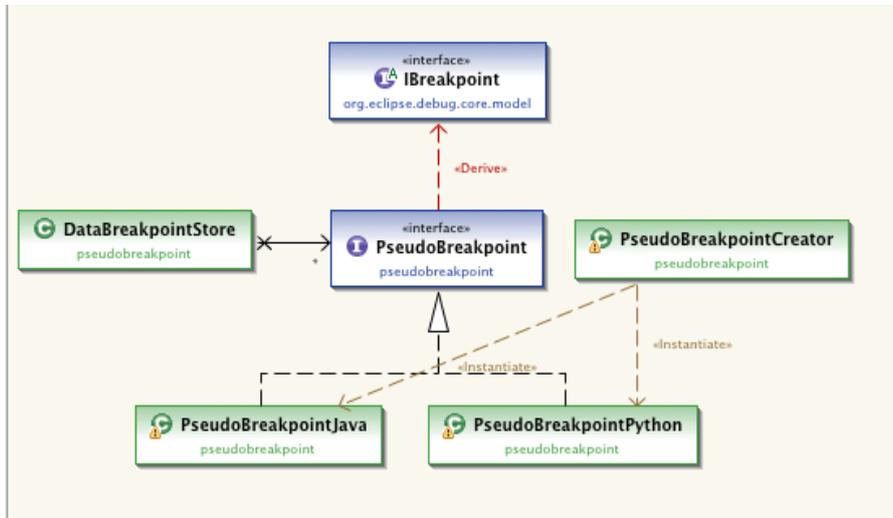


Figure 2: Pseudo-Breakpoint structure

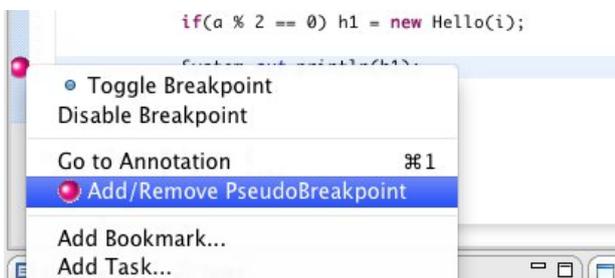


Figure 3: Managing Pseudo-Breakpoints

point is hit (i.e. the corresponding statement is about to be executed). The process is shown schematically in Figure 4. Whenever a breakpoint has been hit, the handler first determines if it was a pseudo breakpoint. If so, then it will walk the object tree, taking a copy of all variables it encounters. It will also record the time the breakpoint was hit and the name of the executing thread. Each variable has a unique object ID allowing individual objects, and the relationships between them, to be identified.

Any necessary comparisons between objects (to determine whether/how the variables have changed) occurs during the generation of the visualisation rather than during the program runtime.

The model design is very simple, and contains two main elements: BreakpointEvent and Variable. Each execution of a program will result in a list of BreakpointEvents (one for each time a PseudoBreakpoint is hit), and each BreakpointEvent will contain a set of Variables — all the variables that were in-scope at the breakpoint event (including local, global, and instance variables). Each variable contains a set of all references it contains (if it is an object), or the value (if it's a primitive). This is done recursively to ensure that all variables and sub-variables are recorded.

Our system is intended to be extensible, so that new visualisations may be added as required. Adding a new visualisation requires development of two main components: a VisualisationDisplayType and a Visualisation (see Figure 5). A VisualisationDisplayType contains all the code to display an Eclipse ViewType. A ViewType is an Eclipse-specific view which is used within the IDE to create a new panel (similar to AWT Panel or Swing JPanel). To create your own custom visualisation, you must extend the VisualisationDis-

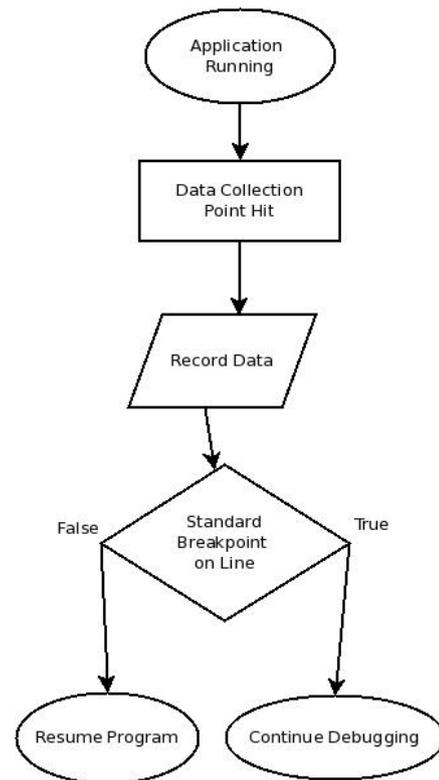
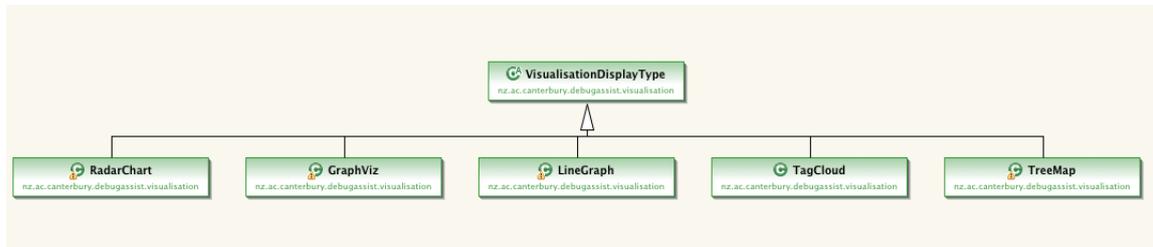
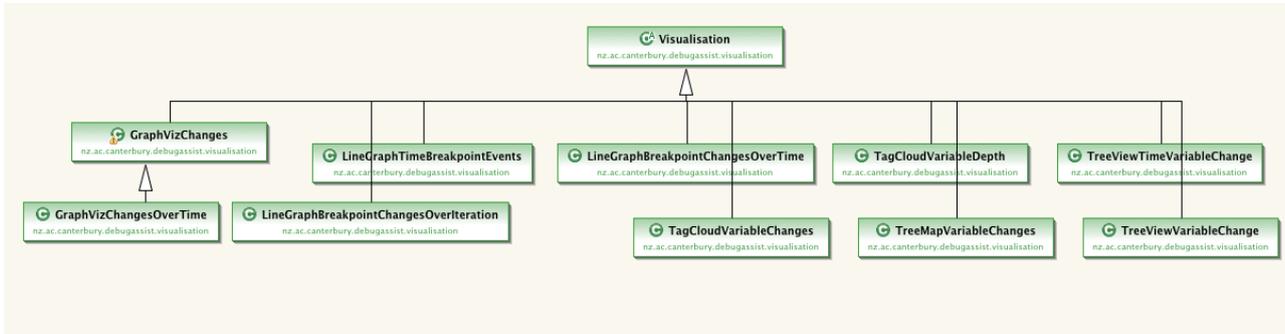


Figure 4: Handling Pseudo-Breakpoint



(a)



(b)

Figure 5: Visualisation structure

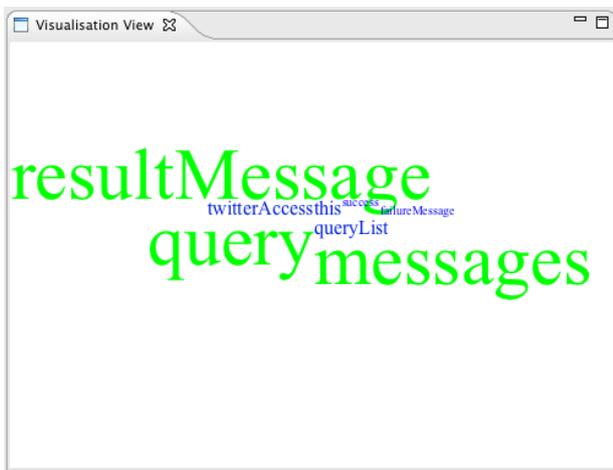


Figure 6: Tag cloud corresponding to state shown in Figure 7

playType class, and implement the createPartControl method to generate the ViewType’s content.

A Visualisation class contains the code to process the breakpoint and variable information for a VisualisationDisplayType to use.

This means that a given VisualisationDisplayType can have multiple Visualisations, and multiple Visualisations can use different VisualisationDisplayTypes.

Currently, each Visualisation and VisualisationDisplay must be integrated manually within the plug-in via the VisualisationHandler class. It is intended that future versions will support the use of an XML configuration file to allow new visualisations to be added without the need to manually alter the code.

To date, we have created a number of visualisations using this framework. An example is our LineGraph (see Section 4.6). We created a generic LineGraph DisplayType using JFreeChart (<http://www.jfree.org/jfreechart>) that was able to generate a ViewPoint to display a number of points on

a Line Graph. A number of visualisation variants are then able to supply data to be plotted on the LineGraphs (such as number of variable changes per breakpoint and time between breakpoint hits). The ability to make use of existing visualisation tools such as JFreeChart, while supporting the development of specialised extensions, makes it relatively straightforward to generate simple, yet useful, visualisations.

Performance has not been the primary focus of our work to date but our experience thus far has been encouraging. Any debugging or monitoring environment will inevitably experience some overheads when compared to the production environment.

In our case, there are two potential sources. Firstly, the number of Pseudo-Breakpoints, the number of times each is activated, and the number of variables to be monitored will contribute to the data collection costs. The cost of processing the data for visualisation is the other potential source of performance issues. We have not found either to be problematic. We ran our system on a typical Java project with 200 Pseudo-Breakpoint activations we found $\mu = 125ms$, $\sigma = 26ms$ for the time cost of data collection — well within acceptable limits,

This cost depends on the number and complexity of in-scope variables, since these are copied when a Pseudo-Breakpoint is hit, so having sufficient client memory available will be important as scope size and activation count increase.

Ongoing and future work includes improving the overhead required with the data collection process and the potential for serialising large data sets for subsequent analysis and replay.

One way to manage the volume of information is to filter out “uninteresting” data. Basic filtering has been implemented. Users can remove primitives from data collections and limit the variable depth. String objects are converted to a primitive type (rather than having all their internals shown) to save space on any visualisations.

Future work would include more extensive and finely-controllable (class and variable based) filtering to help suppress irrelevant information and decrease

memory usage.

4 Visualisations

In this section we present a range of visualisations which illustrate the application of our approach. Having considered the range of tasks to be supported, we have identified a number of common themes and designed individual visualisations around them. Examples include drilling down through hierarchical structures to reveal more detail and determining the differences between the state at successive breakpoint visits.

Figure 7 shows Eclipse in action with a selection of our visualisations in use. Note the significant contrast with Figure 1. Figures 1 and 7 represent two extremes and the user is free to customise the interface as her information needs evolve.

4.1 Tag clouds for context

Tag clouds have become commonplace on blogs and in information retrieval contexts where the holistic contextual overview of the content of a document or document collection is of interest. A recent example is the analysis of Barack Obama's inauguration speech (<http://www.telegraph.co.uk/news/worldnews/northamerica/usa/barackobama/4299886/Barack-Obamas-inauguration-speech-as-a-tag-cloud.html>) In their simplest form, tag clouds consist of a number of words which are terms in a document. The font size used to display each word indicates the frequency with which it occurs in the document. We have extended the concept for use in software visualisation (Deaker et al. 2010). In our approach, visual attributes such as the size, font family, colour and transparency of individual words may each be used to display a variable of interest.

This approach has potential benefits in comprehension of the evolving state of a running program. The tag text is directly mapped to identifiers in the program so no separate translation is required. Properties such as size and colour can be mapped to quantities such as number of invocations or time in scope.

Figure 6 shows an example: it corresponds to the state shown in Figure 7 and indicated by a number of other visualisations there. The font size for each identifier indicates the number of assignments to the corresponding variable. This allows the user to identify rapidly the most active data items which can then be scrutinised more closely.

4.2 Treemaps for hierarchy

Hierarchical structures are common in software engineering. Static examples in Java include the inheritance relationships of classes and the nesting of packages.

In the dynamic context, activities such as following nested scopes, call graphs or drilling into data structures involve navigation and comprehension of hierarchical structures.

In practice, “nearly hierarchical” structures are also common: adding interface implementation to pure inheritance breaks the strict hierarchy and real call graphs aren't necessarily trees. However, representing such situations with visualisations designed primarily for hierarchical contexts is often satisfactory.

Displaying large trees can be awkward. One appealing solution is treemaps, a space-saving representation which can fit neatly within the constraints of a

pane in a GUI component (Johnson & Shneiderman 1991).

We have made use of treemaps in previous work (Churcher et al. 1999, Irwin & Churcher 2002) and they are widely-used in information visualisation applications.

A treemap is visible at the lower right of Figure 7. The rectangular nodes correspond to data items in the scope. The rectangle size indicates the number of children: in this case the larger node (messages) at left is an array while the other two are primitive types. The colour of the rectangle is currently configured to indicate the number of updates which have occurred to the children. Expanding and collapsing the nodes allows the user to explore potentially complex structures without encroaching onto the screen real estate for other concurrent visualisations.

4.3 Augmentation

A common strategy is to implement individual visualisations separately: each has its own primary purpose and is located in a well-defined dedicated part of the IDEs interface. Rather than providing a separate visualisation is it sometimes helpful to “piggyback” by overlaying additional information onto existing display elements.

This approach can be effective when it can take advantage of a diagram or other structure which is familiar to users. UML is arguably the *lingua franca* of software engineering and our users are likely to be familiar with object diagrams, class and sequence diagrams.

Visual attributes such as border thickness and colour can be used to show the quantities such as the number of updates at a particular breakpoint.

The lower left pane visible in Figure 7 shows an augmented UML object diagram. Colour is used to indicate “freshness” (e.g. blue for an object seen for the first time)

4.4 DOI & focus+context

The Degree of Interest of a point (i.e. data item) x , given a focus at x_f is given by

$$DOI(x|_ = x_f) = API(x) - D(x, x_f) \quad (1)$$

API is the *a priori* interest of x . For example we might consider that methods are more important than fields, and that concrete elements are more important than abstract ones. D represents a (conceptual) distance between x and x_f . Thus we might consider length of a call chain or number of generations of inheritance to indicate relative distance between elements.

The Functional form is application-dependent and we are free to allow this to be configurable by users to better support specific tasks.

The augmented object diagram at the lower left of Figure 7 also illustrates the use of DOI-based visualisation. The font size indicates the “distance” of the corresponding identifier from the neighbourhood of the visualisation's focus.

4.5 Augmented tree view

Another effective visualisation technique involves overlaying additional information *in situ* to augment an existing display without diverting the user's focus of attention (Harward et al. 2010).

Figure 8 shows colour being used to augment a breakpoint tree view with information about activity

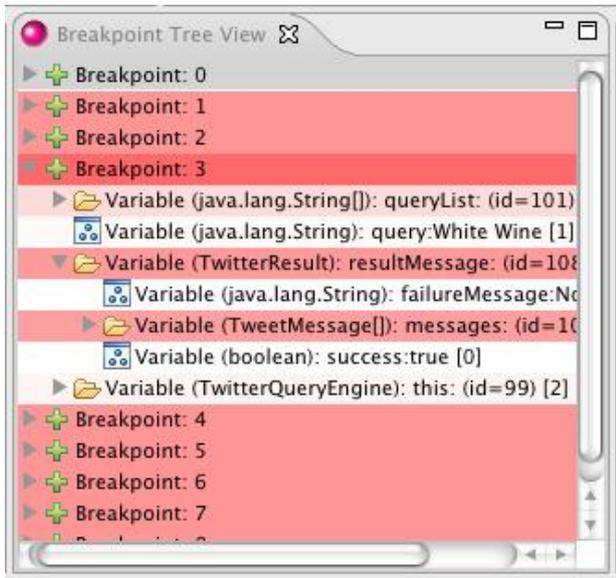


Figure 8: *In situ* augmentation

in the scopes corresponding to children. Expanding the nodes in the usual way reveals finer grained detail.

4.6 Conventional elements

Effective visualisation need not involve “fancy” graphics or 3D effects. Our system supports such conventional diagrams as line graphs. The line graph at the upper right of Figure 7 shows the time intervals between successive activations of a Pseudo-Breakpoint. In this case, the peak corresponds to the creation of an object which caused network authentication to occur, leading to a slight delay.

5 Conclusions

We have proposed Pseudo-Breakpoints as a mechanism for supporting visualisation-driven debugging an other activities requiring understanding of dynamic software state. We have implemented Pseudo-Breakpoints and a range of visualisations in Eclipse and these are available in the form of a plug-in.

Our approach allows visualisations to be integrated into the GUI of an industrial strength IDE and for users to use them alongside conventional breakpoints and their interface in a natural way.

We are encouraged by our results thus far and this work is continuing.

One important activity is the evaluation of the effectiveness of our approach. To date, this has consisted of some anecdotal feedback and “eating our own dog food.” Our intention is to conduct a heuristic evaluation (Nielsen 1992) to guide the ongoing development of the system. A parallel project is gathering usage data and we hope that this will shed light on debugging practices among our target users.

Another possible direction is the ability to record and play back dynamic state information as this is useful in a range of contexts.

References

Brown, M. H. & Sedgewick, R. (1984), ‘A system for algorithm animation’, *SIGGRAPH Comput. Graph.* **18**(3), 177–186.

Churcher, N. & Irwin, W. (2005), Informing the design of pipeline-based software visualisations, in S.-H. Hong, ed., ‘APVIS2005: Asia-Pacific Symposium on Information Visualisation’, Vol. 45 of *Conferences in Research and Practice in Information Technology*, ACS, Sydney, Australia, pp. 59–68.

Churcher, N., Frater, S., Huynh, C. P. & Irwin, W. (2007), Supporting OO design heuristics, in J. Grundy & J. Han, eds, ‘ASWEC2007: Australian Software Engineering Conference’, IEEE, Melbourne, Australia, pp. 101–110.

Churcher, N., Irwin, W. & Kriz, R. (2003), Visualising class cohesion with virtual worlds, in T. Paterson & B. Thomas, eds, ‘Australasian Symposium on Information Visualisation, (invis.au’03)’, Vol. 24 of *Conferences in Research and Practice in Information Technology*, ACS, Adelaide, Australia, pp. 89–97.

Churcher, N., Keown, L. & Irwin, W. (1999), Virtual worlds for software visualisation, in A. Quigley, ed., ‘SoftVis99 Software Visualisation Workshop’, University of Technology, Sydney, Australia, pp. 9–16.

Czyz, J. & Jayaraman, B. (2007), Declarative and visual debugging in Eclipse, in ‘Proceedings of the 2007 OOPSLA workshop on eclipse technology exchange’, ACM, pp. 31–35.

Deaker, C., Pettigrew, L., Churcher, N. & Irwin, W. (2010), Software visualisation with tag clouds, in J. Hosking & B. Long, eds, ‘ASWEC 2010 Industry Track Proceedings’, Auckland, New Zealand, pp. 129–133.

Eades, P. & Zhang, K., eds (1996), *Software Visualisation*, Vol. 7 of *Series on Software Engineering and Knowledge Engineering*, World Scientific.

Furnas, G. (1986), Generalised fisheye views, in ‘Proc ACM SIGCHI ’86 Conference on Human Factors in Computing Systems’, pp. 16–23.

Harward, M., Irwin, W. & Churcher, N. (2010), In situ software visualisation, in J. Noble & C. Fidge, eds, ‘ASWEC 2010’, IEEE, Auckland, New Zealand, pp. 171–180.

Irwin, W. & Churcher, N. (2002), XML in the visualisation pipeline, in D. D. Feng, J. Jin, P. Eades & H. Yan, eds, ‘Visualisation 2001’, Vol. 11 of *Conferences in Research and Practice in Information Technology*, ACS, Sydney, Australia, pp. 59–68. Selected papers from 2001 Pan-Sydney Workshop on Visual Information Processing.

Irwin, W. & Churcher, N. (2003), Object oriented metrics: Precision tools and configurable visualisations, in ‘METRICS2003: 9th IEEE Symposium on Software Metrics’, IEEE Press, Sydney, Australia, pp. 112–123.

Irwin, W., Cook, C. & Churcher, N. (2005), Parsing and semantic modelling for software engineering applications, in P. Strooper, ed., ‘Australian Software Engineering Conference’, IEEE Press, Brisbane, Australia, pp. 180–189.

Jacobs, T. & Musial, B. (2003), Interactive visual debugging with uml, in ‘Proceedings of the 2003 ACM symposium on Software visualization’, ACM, San Diego, California, pp. 115–122.

- Johnson, B. & Shneiderman, B. (1991), Tree-maps: A space-filling approach to the visualization of hierarchical information structures, *in* G. Nielson & L. Rosenblum, eds, 'proc. Visualization '91', IEEE Computer Society Press, Los Alamitos, CA, pp. 284–291.
- Lawrance, J., Bellamy, R. & Burnett, M. (2007), Scents in programs: does information foraging theory apply to program maintenance?, *in* 'Visual Languages and Human-Centric Computing, 2007. VL/HCC 2007. IEEE Symposium on', pp. 15–22.
- Leung, Y. K. & Apperley, M. D. (1994), 'A review and taxonomy of distortion-oriented presentation techniques', *ACM Transactions on Computer-Human Interaction* **1**(2), 126–160.
- Linnberg, J., Korhonen, A. & Malmi, L. (2004), Mvt: a system for visual testing of software, *in* 'Proceedings of the working conference on Advanced visual interfaces', ACM, Gallipoli, Italy, pp. 385–388.
- McGavin, M., Wright, T. & Marshall, S. (2006), Visualisations of execution traces (vet): an interactive plugin-based visualisation tool, *in* 'Proceedings of the 7th Australasian User interface conference - Volume 50', Australian Computer Society, Inc., Hobart, Australia, pp. 153–160.
- Mehner, K. (2002), Jarvis: A uml-based visualization and debugging environment for concurrent java programs, *in* 'Revised Lectures on Software Visualization, International Seminar', Springer-Verlag, pp. 163–175–.
- Neate, B., Irwin, W. & Churcher, N. (2006), Coderank: A new family of software metrics, *in* J. Han & M. Staples, eds, 'ASWEC2006: Australian Software Engineering Conference', IEEE, Sydney, pp. 369–378.
- Nielsen, J. (1992), Finding usability problems through heuristic evaluation, *in* 'Proceedings of the SIGCHI conference on Human factors in computing systems', ACM Press, pp. 373–380.
- Parnas, D. L. (1975), Software engineering or methods for the multi-person construction of multi-version programs, *in* C. E. Hackl, ed., 'Programming Methodology, 4th Informatik Symposium', Vol. 23 of *Lecture Notes in Computer Science*, Springer-Verlag, Wildbad, Germany, pp. 225–235.
- Sarkar, M. & Brown, M. (1994), 'Graphical fisheye views', *Communications of the ACM* **37**(12), 73–84.
- Spence, R. (2001), *Information Visualisation*, Addison-Wesley.
- Stasko, J. (1990), 'Tango: a framework and system for algorithm animation', *IEEE Computer* **23**(9), 27–39.
- Stasko, J., Domingue, J., Brown, M. & Price, B., eds (1998), *Software Visualization: Programming as a Multimedia Experience*, MIT Press.
- Vans, A. M., von Mayrhauser, A. & Somlo, G. (1999), 'Program understanding behaviour during corrective maintenance of large-scale software', *Int. J. Human-Computer Studies* **51**(1), 37–70.
- Ware, C. (2004), *Information Visualization: Perception for Design*, 2nd edn, Morgan Kaufman.

Analysis of Key Installation Protection using Computerized Red Teaming

Tirtha R. Ranjeet, Philip Hingston, Chiou-Peng Lam and Martin Masek

School of Computer and Security Science,
Edith Cowan University
2 Bradford Street, Mount Lawley, Western Australia, 6050.

Abstract

This paper describes the use of genetic algorithms (GAs) for computerized red teaming applications, to explore options for military plans in specific scenarios. A tool called Optimized Red Teaming (ORT) is developed and we illustrate how it may be utilized to assist the red teaming process in security organizations, such as military forces. The developed technique incorporates a genetic algorithm in conjunction with an agent-based simulation system (ABS) called MANA (Map Aware Non-uniform Automata). Both enemy forces (the red team) and friendly forces (the blue team) are modelled as intelligent agents in a multi-agent system and many computer simulations of a scenario are run, pitting the red team plan against the blue team plan.

The paper contains two major sections. First, we present a description of the ORT tool, including its various components. Second, experimental results obtained using ORT on a specific military scenario known as Key Installation Protection, developed at DSO National Laboratories in Singapore, are presented. The aim of these experiments is to explore the red tactics to penetrate a fixed blue patrolling strategy.

Keywords: Red Teaming, Evolutionary algorithm, Key Installation Protection

1 Introduction

This paper presents a tool, ORT (Optimised Red Teaming), which provides automated support for red teaming. We illustrate the use of the tool by exploring potential attack plans to defeat a defensive coastline patrolling strategy designed to protect a key installation.

Red teaming is a process that assists in finding vulnerabilities in a system, whereby the organization itself takes on the role of an “attacker” to test the system. In military organizations, the red teaming concept has long been used at various levels, including organizational and tactical. Traditionally, it is a manual process using humans as actors, resulting in a process that can be expensive, time-consuming, and limited from the perspective of humans “thinking inside the box” (Andrews, 2005, DoD, 2003, Meehan, 2007).

As a possible solution to the “human” limitations in manual red teaming, computerized red teaming uses agent-based simulation (ABS) in which autonomous agents taking on the roles of attacker and defender. Using such a system, where humans are not an intrinsic part of the simulation loop, allows many iterations of the problem to be simulated in a short space of time. This allows for the exploration of a wider range of possible attack/defence strategies, in a shorter time, utilising less real personnel than the traditional manual red teaming. Promising results from the automated simulation can then be checked for legitimacy and evaluated by the expert.

2 Related Work

Upton and McDonald (2003) first suggested using evolutionary algorithms and agent-based simulation for automated red teaming, in their case for testing proposed security procedures. They used an Evolutionary Programming algorithm to evolve the parameters of a red team strategy to defeat a fixed blue team strategy for defence of a fixed structure. The idea of combining agent-based simulation with an evolutionary algorithm has been further developed into the ART framework by researchers at Singapore’s DSO and Nanyang Technological University (Choo et al., 2007, Chua et al., 2008, Xu et al., 2009). Further work on computerised red teaming has also been done at The University of New South Wales’ Australian Defence Force Academy (Ang, 2006) using a simple (1+1) Evolution Strategy algorithm, coupled with WISDOM, a low-resolution simulation model for military simulations.

The ART framework integrates an optimisation algorithm with an agent-based simulation. It currently supports particle swarm optimisation and a multi-objective evolutionary algorithm as the optimiser, and several simulations models, chiefly MANA (Map Aware Non-uniform Automata) (Lauren, 2002). ART has been used in a series of data farming workshops (Lee et al., 2006, Sim et al., 2006, Wong et al., 2007) for applications including urban operations, maritime defence and anchorage protection, and is claimed to be able to discover non-intuitive tactics that are superior to those obtained by manual red teaming.

The aim of (Ang et al., 2006) study was to investigate the nature of the fitness landscape taking into account the personalities of the red and blue teams. The early part of the paper provides a useful survey of computational tools and techniques that are available for defence games.

Recently, (Hingston et al., 2010) proposed RedTNet, a network based modelling framework intended to support

red teaming studies for critical infrastructure protection and strategy games.

3 Optimized Red Teaming (ORT)

ORT is an automated tool developed to assist the red teaming process in finding vulnerabilities in a security plan. The tool is used to explore a situation and identify potentials penetration strategies to 'break blue'. This helps subject matter experts to recognize weaknesses in their plan, which provides the opportunity to take action to address those weaknesses. MANA, an agent-based simulation application, is used to run simulations of the scenario, and a genetic algorithm (GA) is utilized to optimize the combatants' behaviours.

In this section, we describe the design of ORT and its components. The main components of ORT are shown in Figure 1 and are discussed below:

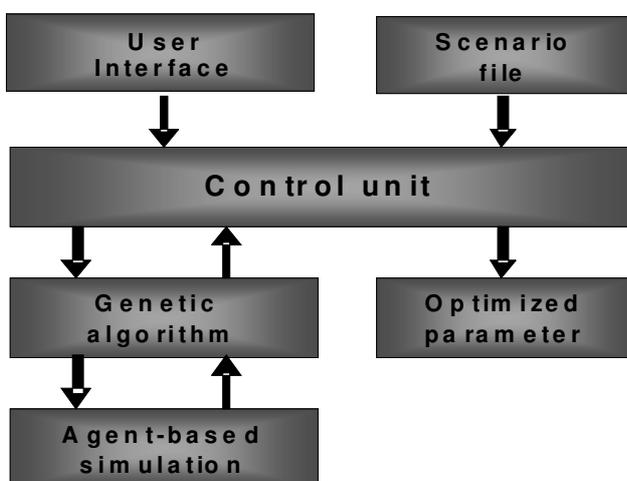


Figure 1: The framework of ORT

- a. **User interface:** This is a graphical user interface which allows the user to supply required information to the program, including genetic algorithm parameters and agent personality parameters for each squad, and to select which squads to optimize.
- b. **Scenario file:** This is a description of the particular scenario, in XML format, which contains details of the environment and at least two military squads with different intentions and targets. The scenario allows for different tactics to achieve the goal.
- c. **Control unit:** This component controls the overall execution of the program, taking parameters as specified via the user interface. These are used to configure and execute the genetic algorithm, running simulations as needed to calculate fitness values.
- d. **Genetic algorithm:** This takes parameter values from the control unit and executes a genetic algorithm, using fitness values calculated using agent-based simulations.
- e. **Optimized parameters:** This is the optimized parameter values for the agents' personalities.
- f. **Agent-based simulation:** The framework uses MANA as the agent-based simulation that runs scenarios in order to evaluate different parameter choices.

To use ORT to analyse a particular scenario, the user selects agent personality parameters via the user interface, which determines the structure of the chromosome to be used for the optimization process. The agent personality parameters are divided into three categories, agent situational awareness (SA), squad SA and inorganic SA. These represent personal, internal group and external group activities. Users have the option to choose which squads to optimize - the red, blue or others. The user also selects GA parameters such as population size, generation number, and crossover and mutation rate, and also the scenario file containing the details of the scenario to be examined. The optimization process can then be initiated.

The genetic algorithm is initialised using the specified GA parameters, and is executed as described in Section 3.1 below. When fitness values are required, sets of simulations are run using parameters values specified by the genome being evaluated. When the genetic algorithm terminates, the optimised parameter values are available to the user, who can then run further simulations to examine and understand the behaviours of the optimised squads.

In this way, the user may be able to identify and address weaknesses in the blue defensive strategy. The improved strategy can then be further tested in a similar way. An alternative is to use ORT to optimise the blue team against the optimised red plan. We illustrate this alternative in later sections (but note that there are potential difficulties, which are also discussed).

ORT makes use of both genetic algorithms and the MANA agent-based simulation. These are described briefly in the next subsections.

3.1 Genetic algorithm (GA)

In this subsection, we briefly explain the process of a typical genetic algorithm. The foundation of evolutionary algorithms (EAs) is evolutionary theory, which suggests that solutions to an optimisation problem can be derived by an evolutionary process that selects a best solution from a population (Abbass et al., 2001, Alcalá et al., 2007, Veldhuizen, 1999, Zitzler, 1999). According to (Coello et al., 2007, Deb, 1999), EAs are adaptive heuristic search algorithms that derive the high quality solutions by using the principles of natural selection: each solution gets a chance to reproduce a certain number of times depending on its performance. Thus, quality results are achieved by selecting among the best solutions. Genetic algorithms are a specific kind of EA. The process of a typical GA may be described in pseudo code:

1. Generate an initial population.
2. Do until the termination condition is satisfied:
 - a. Calculate the fitness of every individual.
 - b. Start a new population.
 - c. Do until new population is complete.
 - i. Select two parents from the old population according to their fitness
 - ii. Perform crossover and mutation to obtain two new offspring
 - iii. Add the new offspring to the new generation.
3. Output optimized parameter values from the best individual in the population.

There are a number of design choices that must be made before applying a genetic algorithm to solve a specific problem of interest. Table 1 lists the GA design choices that are utilized in ORT:

Features	Name
Crossover	Simulated binary crossover (SBX)
Mutation	Polynomial mutation (PM)
Selection method	Stochastic universal selection
Elite individual	Only the best one
Initial population	Each genome is a sequence of randomly generated values from 0 to 100, representing parameter values for each agent's personality.

Table 1: GA features incorporated in ORT

3.2 Map Aware Non-uniform Automata (MANA)

MANA is a cellular automaton combat simulation model, designed at the New Zealand Defence Technology Agency (DTA). It includes a graphic user interface (GUI) that allows users to create new scenarios or loads external scenario files. The features of MANA include agents with situational awareness (SA), a terrain map, event driven personality and flexible waypoints. These features allow a rich set of parameters to be explored when running a scenario. SA influences agent behaviours in MANA. For example, an agent in a squad may detect enemy approaching near to the squad, the information they share among other agents to alert from the situation. Terrain maps are coded using colours to indicate traversability of the terrain. Event driven behaviours help agents to change their activity according to changes in the situation. MANA also contains its own analytical tools including a

genetic algorithm (GA). These analytical tools can be used to find the suitable tactics in order to penetrate an opponents' strategy (Lauren, 2002, McIntosh et al., 2007).

4 Scenario description

The Key Installation (KIN) protection scenario was developed using MANA at DSO National Laboratories, Singapore. Basically, the scenario demonstrates the threats to KIN protection from non-military boats which try to penetrate the regular surveillance of the three blue boats. The fairly low speed blue boats patrol a specific area of the coastline with low level weapons. Conversely, the red boats without weapons try to penetrate the blue patrol to get into the land using different escaping tactics and routes (Chua et al., 2008).

In the original scenario, there are three KINs and three blue patrolling boats. Each blue boat has their patrolling route in which they constantly move to resist any penetrator. The blue surveillance route and KINs along with the initial positions of the red boats are depicted in Figure 2. The red boats are penetrators whose objective is to reach into the land by escaping from the blue patrol and destroy KINs.

4.1 Parameters

Each squad's behaviour is determined by a number of parameters. Table 2 lists these parameters. The default parameters used for the blue agents in the scenarios are depicted in Table 3.

The parameter values for the red team are evolved using the genetic algorithm. Thus the genome is a sequence of real parameter values in the ranges indicated in Table 2.

Characteristics Considered	Description	Values in Range
Movement Speed	The value determine the number of cells agents move in a given time step. Its range is 0 to 1000; however normalized to 100 so that an agent can move one cell per time step.	0 to 100
Agent SA – Agents take actions on the basis of the information available from its own sensors. Negative and positive value indicates repulsion and attraction respectively.		
Enemy	Attraction or repulsion with the agent with enemy allegiance	-100 to 100
Enemy Threat 3	Attraction or repulsion with the agent with enemy allegiance Threat Level 3	-100 to 100
Uninjured Friends	Attraction or repulsion with the agent with same allegiance	-100 to 100
Cover	Determine the distance of shooting by direct fire weapons in the terrain.	-100 to 100
Concealment	Determine the visibility of agents in the terrain.	-100 to 100
Squad SA - Agent stake actions on the basis of the information available on the squad's SA map. Negative and positive value indicates repulsion and attraction respectively.		
Enemy Threat 3	Attraction or repulsion with the agent with enemy allegiance Threat Level 3	-100 to 100
Friends	Attraction or repulsion with agents of the same squad	-100 to 100

Table 2: Selected agent personality parameters

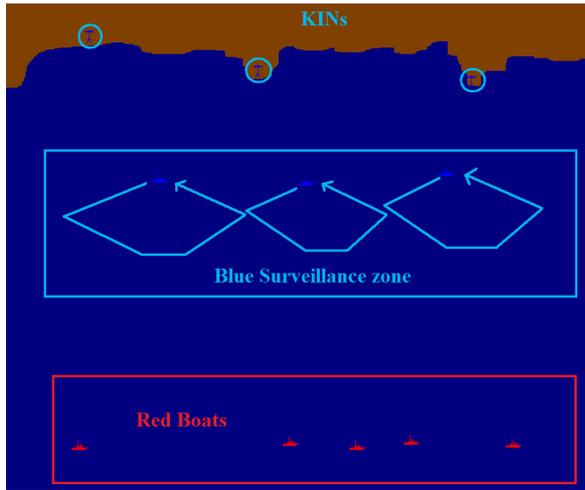


Figure 2. Scenario for Key Installation protection

Personalities			Normal Behaviour	Enemy Contact
Distance from Enemy	Agent SA	Enemy		
		Enemy Threat 3	0	0
	Squad SA	Enemy Threat 3	0	0
Distance from Friends	Agent SA	Uninjured Friend	0	0
	Squad SA	Friend	0	0
Movement Speed			60	80

Table 3: Default characteristics of the blue agents

Each blue boat follows its specified route unless one of the red boats comes into their contact area. The values given under ‘normal behaviour’ section in Table 3 are all 0, meaning that they are neither aggressive to enemies nor affected by friendly boats. They circle their route at normal speed (movement speed is 60). When any blue boat finds a red boat within its sensor range, it switches to the ‘enemy contact’ parameters values, and its behaviour will become aggressive (the value of 100 indicates that it will chase after the enemy, and will do so with the greater speed of 80).

4.2 Measures of Effectiveness (MOE)

Two factors are considered as measures of effectiveness (MOEs) to evaluate the performance of the red team:

1. Maximizing the goal achievement – that is, breaking the blue boat patrolling tactics by getting at least one boat to the land.
2. Minimizing red casualties

These are combined to define the fitness function to guide selection in the genetic algorithm, using the formula:

$$\text{Fitness} = \text{Red Goal Success Proportion} * (\text{Number of red agents})^2 - \text{Mean red casualties} + \text{Number of red agents}.$$

5 Initial experimentation

In order to explore the strategies available to each side in this scenario, we consider variations with different numbers of attacking boats. Every scenario has the same number of the blue agents, patrolling strategy and mission (which is to prevent the red boats attacking the key installation). The numbers of red boats is varied between two and five. In the first variation of the scenario, two red boats try to penetrate against the three blue patrolling boats. Subsequently, the second, third and fourth scenarios have three, four and five red boats respectively.

For these experiments, following some preliminary testing, we set the GA parameter values as listed in Table 4 below:

Properties	Values
Agent-based simulation	MANA
Evolutionary algorithm	GA
Simulations per individual	20
Population size	20
Generations	50
Crossover Rate	60%
Mutation rate	1/population size
Number of experiments	20

Table 4: GA parameter values

In MANA, the simulation termination condition was set to 1000 simulation steps, or all red agents destroyed, or achieving the goal by any red agent (reaching the land). ORT executed 20000 (= 20 x 50 x 20) simulation runs in each experiment. It takes less than 1 second to evaluate each individual on a standard personal computer with 1GB RAM and 1.6 GHz CPU.

6 Discussion

Convergence graphs showing the fitness values over generations in the scenarios with the red agents ranging from two to five are depict in Figure 3, Figure 4, Figure 5 and Figure 6 respectively. Each graph shows the minimum, maximum and median values of best fitness values for each generation over 20 repeats of the genetic algorithm. Not unexpectedly, the results demonstrate that there is a direct relationship between the number of penetrators involved in the battle and the likelihood of them achieving their goal.

In all experiments, we can see that the search converges quickly, in less than 20 generations. With five attackers, the genetic algorithm reliably converges to a solution with a fitness value close to 30. However, with only two attackers, convergence is much less reliable, with a range of final fitness values between 5 and 6. This may indicate that it is more difficult to find good solutions for the red team when there are fewer agents available. There is non-

linear relationship between the agent numbers and the number of red casualties, as casualties increase when there are many agents involved in the penetration process.

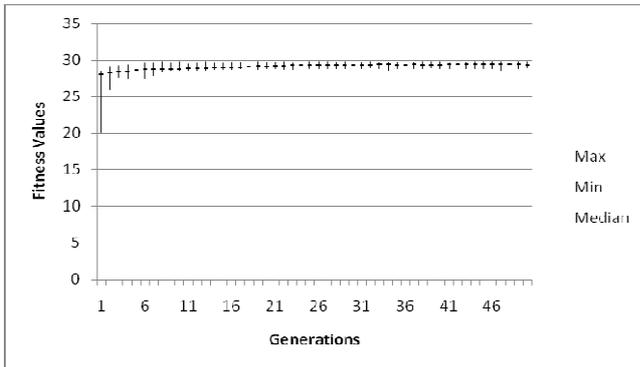


Figure 3. Minimum, maximum and median fitness values of the red team while considering five red boats.

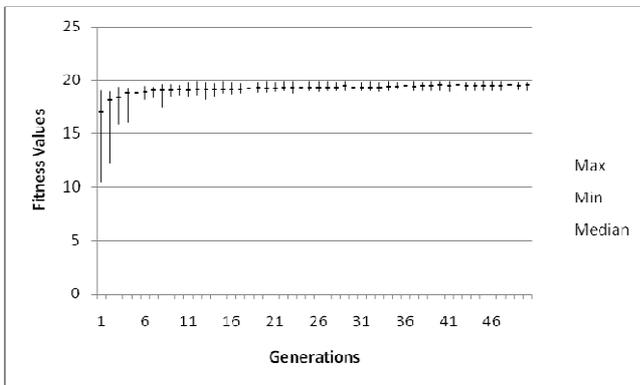


Figure 4. Minimum, maximum and median fitness values of the red team while considering four red boats.

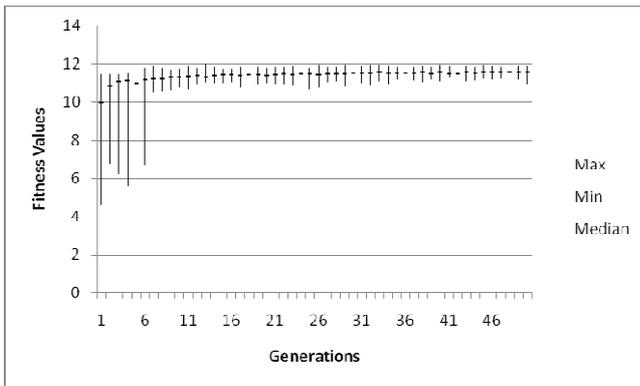


Figure 5. Minimum, maximum and median fitness values of the red team while considering three red boats.

The experiments show that the red teams alter their tactics and behaviours as the number of penetrator boats changes. Example tactics incorporated by the red team when different numbers of red boats are involved in blue penetration are depicted in Figure 7, Figure 8, Figure 9 and Figure 10 below.

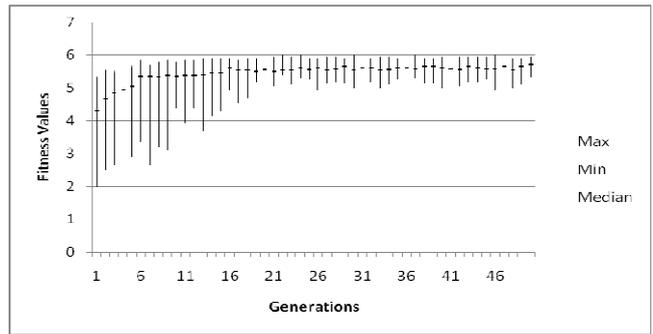


Figure 6. Minimum, maximum and median fitness values of the red team while considering two red boats.

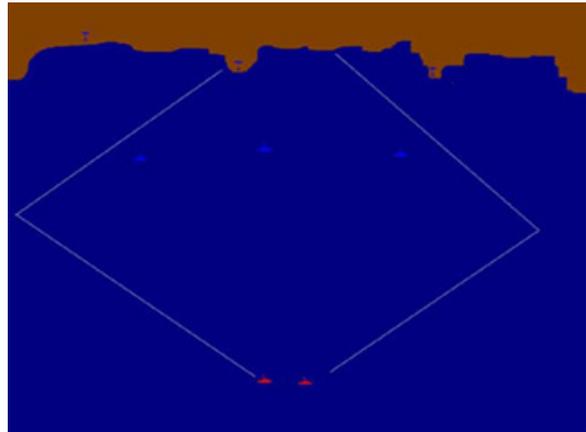


Figure 7. The route suggested by ORT for two red boats to penetrate the three blue patrolling boats.

In Figure 7, the red boats avoid confrontation with the blue boat and find a secure way to the land. When there are three red boats as in Figure 8, the tactics use one boat as a distraction so that other two can easily pass through the blue patrol formation.

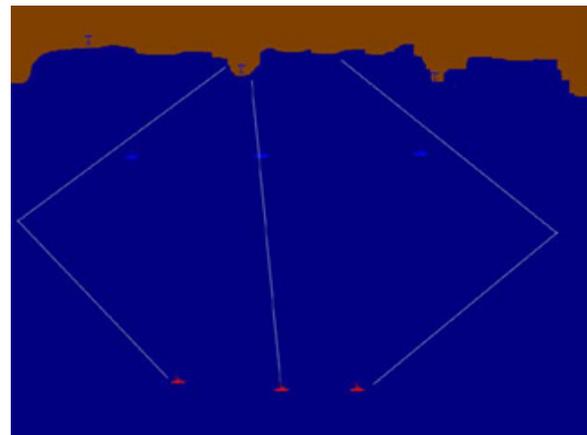


Figure 8. The route suggested by ORT for three red boats to penetrate the three blue patrolling boats.

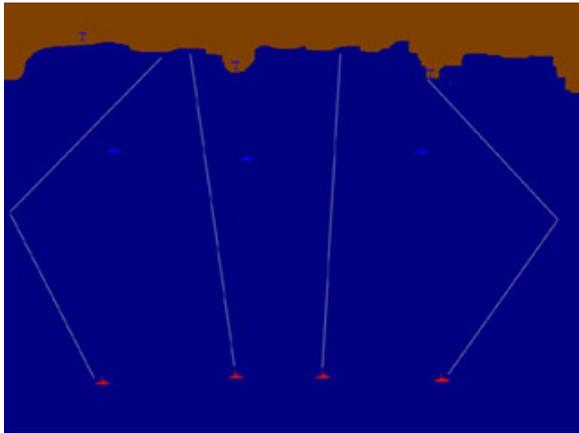


Figure 9. The route suggested by ORT for four red boats to penetrate the three blue patrolling boats.

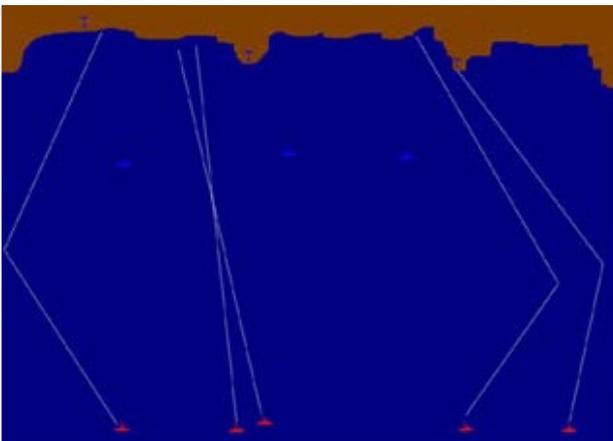


Figure 10. The route suggested by ORT for five red boats to penetrate the three blue patrolling boats.

Figure 9 and Figure 10 show a mixed strategy. The boats at the corner avoid confrontation whereas the others move towards the patrolling area by maintaining distance with friendly boats. Similar to the scenario with three red boats, the tactics use some distraction boats in order to allow the rest of the boat to achieve the goal.

The tactics show that with a smaller number of red boats, the red team should follow flanking tactics to achieve the goal. The result demonstrates the behaviour of internal cooperation among the red agents when the red boats are varied. The cooperation among the red boats is strong when a large number of red agents are involved in the penetration process. Conversely, they maintain distance if there is a smaller number of agents involved, which leads them to follow flanking strategies.

As the number of the red agents increases, their tactics change from flanking to direct confrontation. However, they avoid conflict and try to find a narrow escape between the blue patrolling routes to avoid casualties.

The personality values for the red team with two, three, four and five agents as suggested by ORT are depicted in Table 5, which indicate that the red agents stay away from the blue boats and they maintain distance between friendly agents also. The flanking tactics and increased speed help the red agents to avoid confrontation with the blue agents and reach the goal. The red teams with the given characteristics succeed almost 100% to

achieve the goal while minimizing their casualties. The negative value under 'Enemy' shows they fear of the blue and stay away from their contact. The positive and negative value in 'Friend' rows show closeness and distance with the friendly boats.

Personalities/ Boat No.		2	3	4	5	
Distance from Enemy	Agent SA	Enemy	-90	-60	-83	-93
		Enemy Threat 3	-95	-98	-99	-98
	Squad SA	Enemy Threat 3	-85	-75	-90	-87
Distance from Friends	Agent SA	Uninjured Friend	-96	-35	30	50
		Squad SA	Friend	-65	-20	22
	Inorganic SA		0	0	0	0
Movement Speed		100	100	100	100	

Table 5: Personality of the red suggested by ORT for a red team with two agents

Scenarios with agent personalities as listed in Table 5 were further analysed to evaluate their effectiveness. For this, an additional 50 repetitions of each scenario were run in MANA. Table 6 tabulates the mean MOE and fitness values for different numbers of red agents.

Red agents	Mean Casualties	Std. Dev. (+/-)	Mean Success Rate	Std. Dev. (+/-)	Fitness
2	0.38	0.07	0.95	0.02	5.54
3	0.65	0.19	0.96	0.05	11.03
4	0.7	0.10	0.97	0.02	19.04
5	1.24	0.14	0.98	0.02	28.26

Table 6: Mean casualties and success rate of optimized red team

The results in Table 6 indicate that there is a direct relation between the number of agents involved in penetration and their success rate. Conversely, there is negative relation between the number of agents and their attrition.

7 Further Experimentation

To further explore the strategy options, in response to the evolved red team, another experiment was devised to consider the blue agents to be optimized against the optimized red agents. For this, only the scenario with two red boats is considered. The default personality values for the red boats are shown in the second column of Table 5. GA parameters were the same as in the previous experiments, as depicted in Table 4.

Two factors are considered as MOEs, to evaluate the individuals: maximizing the red casualties and stopping the red boats to pass through the patrolling area. The formula used in fitness function is:

Fitness = Mean red casualties – Red goal success proportion + number of blue agents

ORT suggests characteristics for the blue team to stop the red boats as depicts in Table 7. The emerged tactics for the blue boats to respond the optimized red boats alter the blue behaviours make them more aggressive and active. Despite the use of flanking tactics by the optimized red boats, the later optimized blue boats are capable of taking action against them.

Against the default blue strategy, optimized red boats would reach destination almost 100% of the time; however, when the blue team are optimized their winning ratio is reduced by one third. The fall of the red winning ratio after blue optimization indicates that improved tactics can address the weaknesses of the plan if they are identified in advance.

Personalities			Normal Behaviour	Enemy Contact
Distance with Enemy	Agent SA	Enemy	0	100
		Enemy Threat 3	0	100
	Squad SA	Enemy Threat 3	0	25
Distance with Friends	Agent SA	Uninjured Friend	0	-84
	Squad SA	Friend	0	-71
Movement Speed			60	100

Table 7: Optimized personality of the blue agents

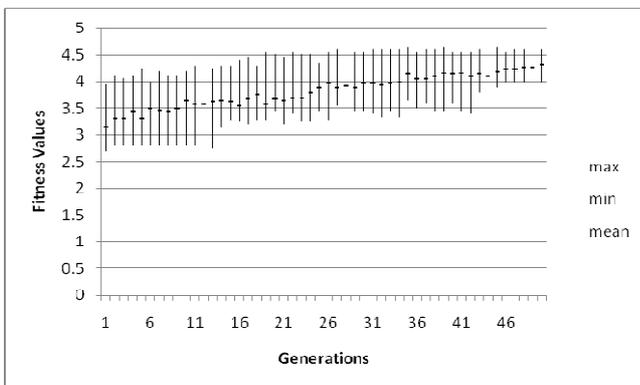


Figure 11. Maximum, minimum and mean fitness values of the blue team while considering two red boats trying to penetrate three blue patrolling boats in the scenario.

In order to monitor the progress of the GA, Figure 11 depicts the fitness values of the blue teams in each generation. The graph indicates that the gaps between maximum and minimum values are wide in every generation and convergence is hard to acquire when optimizing the blue team against already optimized the red team.

A word of warning is in order here – it would be wrong for the blue side to assume that its plans will now be effective against red attacks. It may be that different red tactics would defeat these blue tactics, which are only optimised against one specific type of red tactic. A

comprehensive analysis would have to consider the range of possible red tactics and their likelihood.

Red agents	Mean Casualties	Std. Dev. (+/-)	Mean Success Rate	Std. Dev. (+/-)	Fitness
2	1.46	0.09	0.32	0.07	4.14

Table 8: Mean casualties and success rate of red boats after optimizing the blue boats.

8 Conclusion

In this paper we demonstrated the use of ORT, as a tool to assist the red teaming process for detecting weaknesses in tactical security plans. We have seen different tactics emerge in response to the blue patrolling boats in different scenarios, and shown that we can develop blue tactics to respond to optimised red tactics. While the simple approach illustrated here can be used to gain valuable insights into a scenario, in general, the situation is very complicated and ventures into the realms of game theory. We intend to explore this in future work using co-evolutionary algorithms.

Acknowledgment

The authors would like to thank Defence Technology Agency (DTA) New Zealand for providing MANA and DSO National Laboratories, Singapore for sharing the scenarios.

References

- ABBASS, H. A., SARKER, R. & NEWTON, C. 2001. PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems. *Proceedings of the Congress on Evolutionary Computation 2001*.
- ALCALA, R., ALCALA-FDEZ, J., GACTO, M. J. & HERRERA, F. 2007. A Multi-Objective Evolutionary Algorithm for Rule Selection and Tuning on Fuzzy Rule-Based Systems. *Proc. of the 16th IEEE Int. Conf. on Fuzzy Systems, FUZZ-IEEE'07*, 1367-1372, London.
- ANDREWS, P. 2005. Red Teams: Toward radical innovation. *IBM Advanced Business Institute in Palisades, New York*.
- ANG, Y. 2006. A Networked Multi-Agent Combat Model: Emergency Explained. *University of New South Wales, New South Wales, Australia*.
- ANG, Y., ABBASS, H. A. & SARKER, R. 2006. Characterizing warfare in red teaming *Systems, Man and Cybernetics, Part B: Cybernetics, IEEE Transactions*, 36, 268-285.
- CHOO, C. S., CHUA, C. L. & TAY, S. V. Year. Automated Red Teaming: A Proposed Framework for Military Application. *In: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, 2007 London, England, United Kingdom. 1936-1942.
- CHUA, C. L., SIM, C. W. C., CHOO, C. S. & TAY, V. 2008. Automated Red Teaming: An Objective-based Data Farming Approach for Red Teaming.

- Proceedings of the 2008 Winter Simulation Conference.*
- COELLO, C. A. C., LAMONT, G. B. & VELDHUIZEN, D. A. V. 2007. Evolutionary Algorithm for Solving Multi-Objective Problems, Kluwer, Boston.
- DEB, K. 1999. Multi-objective Genetic Algorithms: problem Difficulties and Construction of Test Problems. *Evolutionary Computation*, 7, 205-230.
- DOD 2003. The Role and Status of DoD Red Teaming Activities. *Defense Science Board Task Force.*
- HINGSTON, P., PREUSS, M. & SPIERLING, D. 2010. RedTNet: A Network Model for Strategy Games. *IEEE Congress on Evolutionary Computation, Barcelona.*
- LAUREN, M. K. 2002. A Metamodel for Describing the Outcomes of the MANA Cellular Automaton Combat Model Based on Lauren's Attrition Equation. *DTA Report 205.*
- LEE, M., ANG, D. & HUNG, L. F. 2006. Applying automated red teaming in an urban ops scenario *In scythe 1: Proceedings and Bulletin of the International Data Farming Community, Monterey, CA, USA, Naval Postgraduate School, 2430.*
- MCINTOSH, G. C., GALLIGAN, D. P., ANDERSON, M. A. & LAUREN, M. K. 2007. MANA: Map Aware Non-Uniform Automata Version 4 User Manual.
- MEEHAN, M. K. 2007. Red Teaming for Law Enforcement. *Police Chief*, 74.
- SIM, W. C., CHOO, C. S., M-TIBURCIO, F., LIN, K. & SHEE, M. Year. Applying automated red teaming in a maritime scenario. . *In: In Scythe 2: Proceedings and Bulletin of the International Data Farming Community 2006 Monterey, CA, USA, NavalPostgraduate School. 2629.*
- UPTON, S. C. & MCDONALD, M. J. 2003. Automated Red Teaming Using Evolutionary Algorithm. *WG31-Computing Advances in Military and Security Applications of Evolutionray Computation, GECCO.*
- VELDHUIZEN, D. A. V. 1999. Multi-Objective Evolutionary Algorithms: Classifications, Analysis and New Innovations. Ph.D. thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Tech., Wright-Patterson AFB, Ohio,
- WONG, A. C. H., CHUA, C. L., LIM, Y. K., KANG, S. C., TEO, C. L. J., LAMPE, T. & ABBOTT, P. H. B. 2007. Applying automated red teaming in a maritime scenario. *In Scythe 3: Proceedings and Bulletin of the International Data Farming Community, Monterey, CA, USA, NavalPostgraduate School. 35.*
- XU, Y. L., LOW, M. & CHOO, C. S. Year. Enhancing automated red teaming with evolvable simulation. *In: GEC '09: Proceedings of the first ACM/SIEGVO Summit on Genetic and Evolutionary Computation, ACM, 2009 New York, USA. 687-694.*
- ZITZLER, E. 1999. Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Ph.D. thesis, Swiss Federal Institute of Technology, Zurich.

A New Term Ranking Method Based on Relation Extraction and Graph Model for Text Classification

Dat Huynh

Dat Tran

Wanli Ma

Dharmendra Sharma

Faculty of Information Sciences and Engineering
University of Canberra
ACT 2601, Australia,
Email: {dat.huynh, dat.tran, wanli.ma,
dharmendra.sharma}@canberra.edu.au

Abstract

Term frequency and document frequency are currently used to measure term significance in text classification. However, these measures cannot provide sufficient information to differentiate important terms. Thus, in this research, a new term ranking (weighting) approach for text classification will be proposed. The approach firstly is based on relations among terms to estimate the important levels of terms in a document. Secondly, the proposed approach provides a considerable representation for the text documents. The results from experiment show that with the same data in Wikipedia corpus the term weighting approach provides higher accuracy in comparison to the popular approaches based on term frequency.

Keywords: Text Representation, Term Weighting Approach, Relation Extraction, Graph Building Model, Graph Weighting Model, Text Classification.

1 Introduction

The task of text classification is to automatically assign single or multiple category labels to a new text document based on category models created after learning a set of training documents with correct category labels. Current text classification methods convert a text document into a relational tuple using the popular vector-space model to obtain a list of terms with corresponding frequencies. A term-by-frequency matrix, interpreted as a relational table, will be obtained to represent a collection of documents (Wang et al. 2005).

Term frequency (*tf*) has been used to measure term significance in a specific context (Robertson & Jones 1997) and to estimate the probabilistic distribution of features using maximum likelihood estimates. The more a term is encountered in a certain context, the more it contributes to the meaning of the context. Other approaches based on the combination between *tf* and inverse document frequency (*idf*) (Yang & Pedersen 1997, Joachims 1998, Yang & Liu 1999, Yu & Zhang 2009) have also been proposed to solve the problems of classifying text documents. However, with some abstract and complex corpus where the contents of text documents are much more equivalent in term of statistic information, those approaches

cannot differentiate documents and achieve high classification results (Joachims 1998, Yang & Pedersen 1997).

To overcome this shortcoming, some approaches have been recently proposed to discover more relationships among terms that represent for a given document. Most of those approaches used graph model to connect the relationships and applied centrality algorithms to identify significant terms that represent the document context. Relationships could be extracted from particular collections such as WordNet or Wikipedia, in which connections between words are based on characteristics of word-senses or Wikipedia links (Wang et al. 2007, Gabrilovich & Markovitch 2007, Strube & Ponzetto 2006, Yeh et al. 2009, Hu et al. 2008).

Term co-occurrence (*tco*) is the most popular method to model relationships among terms. The relations are considered as parts of a graph, and a random walk algorithm is applied to rank important terms based on characteristics of the connections (Hassan & Banea 2006, Wang et al. 2007, 2005). Our investigation indicates that *tco*-based methods can provide more relationships among terms. In a recent work (Hassan & Banea 2006), any combinations of single nouns within a certain window are accepted as relationships. However, the contribution of those relations to the document in term of semantic aspects is still an open question. In order words, those combinations are more referable to the statistical relationships rather than the relationship in document contexts. The relations not only contribute statistical information to document representation, but also more importantly they contribute noises to the contexts of document representation in terms of semantic aspects.

Recognising the deficiency of those approaches, in this research, an alternative term weighting method is proposed, which not only discovers more meaningful relationships among terms from a short context, but also weights the importance of terms based on their participations on the global contexts.

The remaining of this paper is organised as follows. Section 2 presents the framework of term weighting approach. Section 3 explains in details the methodology for extracting relations between terms from a given text document. Section 4 introduces the procedures of taking advantages of relations to return list of term representatives. Section 5 shows how to apply the document representations to text categorisation tasks. Section 6 describes our experimental results. The conclusion and future work will be discussed in section 7.

2 The Proposed Term Weighting Framework

The proposed term weighting framework includes the following main phases, (figure 1):

1. *Relation Extraction*: Each document in a text corpus is processed to extract a set of relations representing the contents of that document.
2. *Graph Ranking*: A weighted and directed graph is constructed by connecting all extracted relations from the previous phase. A random walk algorithm is applied to estimate the levels of importance of terms as their weightings. A weighted list of terms is returned and is regarded as the representation of the given document.

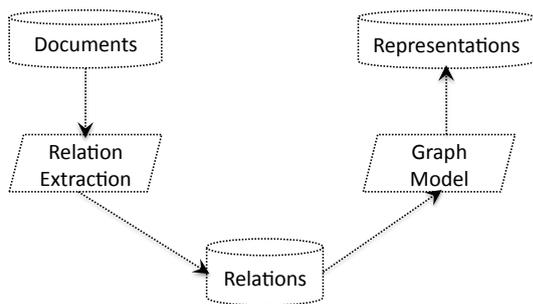


Figure 1: The proposed framework of extracting document representations from a text corpus

3 Relation Extraction

Relation extraction is an important research in text mining, which aims at extracting relationships between entities from text documents. The unsupervised relation extraction method (Banko et al. 2007) is used to extract relationships from web documents. Other methods extract relations based on particular kinds of patterns or seeding examples (Agichtein et al. 2000, Etzioni et al. 2004, Brin 1998). However, those methods extract relations from unstructured documents, where their efficiency can be considered in a case of working with a large number of input documents. Thus, to commit the task of extracting relations among terms, it is necessary to consider an alternative approach to extract relations for text classification.

In this section, a method of extracting relations from an input document based on syntactic analysis is presented. The approach considers roles of words in a sentence to take into account their built-in and hidden relations. For instance, from the sentence “Antibiotics kill bacteria and are helpful in treating infections caused by these organisms”, a list of relations will be obtained such as (Antibiotic, kill, bacteria), (Antibiotic, treat, infection), (Antibiotic, treat infection cause, organism), and (infection, cause, organism).

A relation is considered as a tuple $t = (e_i, r_{ij}, e_j)$, where e_i and e_j are strings denoted as terms, and r_{ij} is a string denoted as the relationship between them. Figure 2 gives an example of tuples.

```

(Garlic, kill , bacteria)
(Garlic , inhibit , bacteria)
(Garlic, kill, virus)
(Garlic , prevent , disease)
  
```

Figure 2: Examples of tuples

3.1 A Framework for Extracting Relations

The relation extraction stage consists of three main steps: pre-processing documents, extracting tuples and optimising relations as shown in figure 3.

- *Pre-processing documents*: documents from the input corpus are pre-processed to extract sentences.
- *Extracting tuples*: an extractor applies a linguistic parser to analyse the syntactic structure of an input sentence and outputs a graph of linkages. The extractor then walks along the graph and applies an heuristic algorithm to extract raw tuples.
- *Optimising relations*: an optimiser receives all the raw tuples as its input and converts every single word into its simple form. Some non-essential words such as stop-words from tuples will be eliminated. The remaining tuples are regarded as a set of relations.

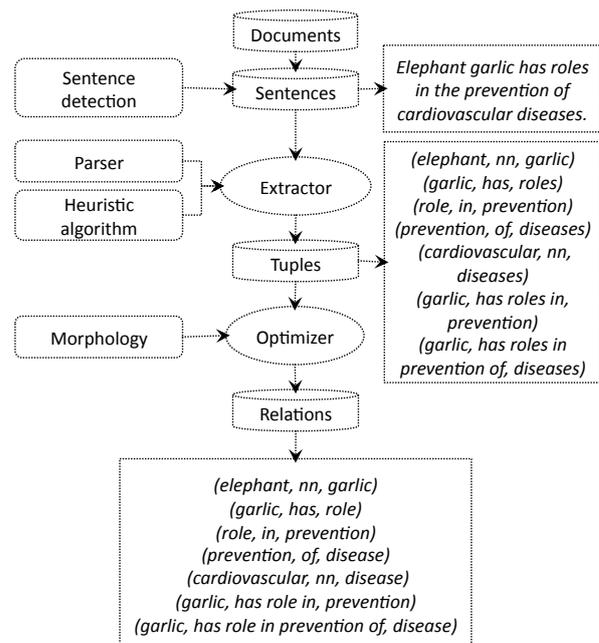


Figure 3: The proposed framework of extracting relations

3.2 An heuristic algorithm to extract relations

A linguistic parser¹ is used to analyse a sentence to produce a graph of linkages, in which relations among words are discovered. Figure 4 shows the graph of linkages extracted from the following sentence “Elephant garlic has a role in the prevention of cardiovascular disease”.

¹The Stanford parser <http://nlp.stanford.edu/software/lex-parser.shtml>

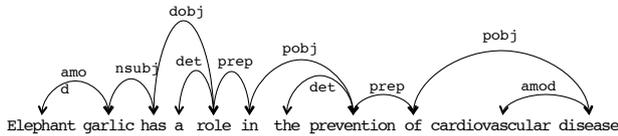


Figure 4: A linkage structure of a sentence

Moreover, the statistic information from Downey et al. (2005) has demonstrated that the majority sentences from English documents are categorised into certain kinds of sentence structures. Thus, the heuristic algorithm is designed to walk on the graph of linkages and extract raw tuples, based on those kinds of sentence structures.

The algorithm firstly scans a graph of the linkages from a sentence and identifies every single pairs of base noun² (e_i, e_j) with $i < j$. From each pair of noun phrases, if there is a shortest path (Bunescu & Mooney 2005) connecting from e_i to e_j , the algorithm will go along the path and identify a sequence of words between e_i and e_j . These words are considered as a potential relation r_{ij} to form the raw tuple $t = (e_i, r_{ij}, e_j)$. In the case that e_i and e_j are located next to each other, they should be connected by a direct connection. So, the relation r_{ij} is regarded as the name of the linkage (kinds of syntactic connection).

In short, if the raw tuples are passed through the following list of constrains, they will be retained to the next processing step. Here are some constraints to test the raw tuples

- e_i has to be a base noun
- e_j has to be a base noun
- r_{ij} has to be in a shortest path connecting e_i to e_j
- r_{ij} has to contain a verb or a preposition.
- w_k belonged to r_{ij} needs to match $i < k < j$.

After extracting the set of raw tuples from each document, components in each tuple is then optimised. First of all, all non-essential words will be eliminated such as adverbs, relative clause marker (who, whom, which, that, etc.), and stop-words from e_i and e_j . Then the morphology technique is used to convert all words in their simple forms such as the plural nouns into its single form, any kind of verb forms into its “root” word (Minnen et al. 2001). For instance, the noun phrase “developing countries” is converted to “develop country”, the verb phrase “have been working” is converted to “have be work”. Once all wired tuples are eliminated, the remaining tuples are considered as a set of relations representing the document.

4 Graph Construction: Constructing, Weighting and Ranking Graph

Graph model is an alternative way to model information, which shows clearly the relationships among its vertices. It also groups the related information in a certain way, in which a centrality algorithm can take their best advantages. Recent approaches have used the model to select representations of a given text document. Firstly, the co-occurrence between

²Base noun is considered as a single noun or a main noun in a noun phrase. With topic/theme text classification, noun information is more informative than other kinds of words

words from the document is regarded as the relationships on the graph (Rada & Paul 2004). The graph model considers words as its vertices, and the term co-occurrence is measured as the weight of the relation. Secondly, taking the advantages of the universal corpus, Wikipedia, some authors have constructed the graph based on Wikipedia links. The graph connects all the concepts among Wikipedia documents. Under the light of the success of graph model to extract the representations of documents, we also propose a method to extract the document representation by taking the advantages of the extracted relations as well as the graph model.

4.1 Conducting Graph

A graph model is built to connect all extracted relations. Given a relation $t = (e_i, r_{ij}, e_j)$, where e_i and e_j are considered as vertices in the graph and r_{ij} is considered as the edge connecting between e_i and e_j . The weight of the edge w_{ij} is calculated based on the importance of r_{ij} in its documents or the relatedness between e_i and e_j in the corpus. In this paper, we are using the former one to measure the weightings of relations.

4.2 Weighting Graph

The weight w_{ij} is calculated based on two factors. Firstly, it is based on the frequency of a relation t in the document d . The higher redundancy of relation t is, the more important it is in the document d . Secondly, w_{ij} is based on the redundancy of relation t in the corpus. The redundancy of a tuple determines how valuable of that information from its document (Downey et al. 2005). So, w_{ij} is calculated as follows. Let $t = (e_i, r_{ij}, e_j)$ be a relation of d , and $e = (e_i, w_{ij}, e_j)$ be an edge of the graph. So w_{ij} is calculated as

$$w(r_{ij}) = freq(t, C) * rf(t, d) \quad (1)$$

$$rf(t, d) = \frac{freq(t, d)}{\sum_{i=1}^{|t:t \in d|} freq(t_i, d)} \quad (2)$$

where $freq(t, C)$ is the frequency of tuple t in the corpus C , $freq(t, d)$ is the frequency of tuple t in the document d , and $rf(t, d)$ is the normalised relation frequency value of the relation t in the document d .

A document from the corpus is represented as a directed weighted multi-graph, in which every single term is considered as a vertex of the graph. In order to weight the important levels of terms, we are using a centrality algorithm PageRank (Lawrence et al. 1998). The first reason of using PageRank for weighting terms is based on its original intuition. A page will have a high rank if there are many pages in the web pointing to it, or if there are some pages with high ranks pointing to it. Adapting the ideas for weighting important terms, a term is considered as importance if it participates the majority relations with other terms of the document, or if it has the relations to other important terms. Secondly, from a larger number of centrality weighting algorithms, PageRank is considered an outperformed method in evaluating importance information from the graph (Ravi & Rada 2007).

As a result, in order to use PageRank algorithm, every vertex from the graph needs to be treated as a webpage, the graph of relations needs to be

converted into a directed and weighted multi-graph. Thus, every undirected edge $e = (e_i, w_{ij}, e_j)$ is converted to two directed edges $\vec{e} = (e_i, w_{ij}, e_j)$ and $\overleftarrow{e} = (e_j, w_{ij}, e_i)$. Then, the directed graph is passed through the PageRank as its input data and the output is returned as a set of vertices with their ranking scores. Thus, every vertex e_i has its ranking score pr_i , which is considered as the degree of significance of term e_i in the document d . As a result, a list of terms with their ranking values is the representative for the given document.

5 Applying Graph-based Ranking Approach to Text Classification

In the previous sections, we have discussed the novel method of extracting the representatives of a given document from a text corpus. Given a text document d_j from a corpus C , the list of n terms representatives of d_j is

$$d = \left\{ (w_1, pr(w_1, d)), \dots, (w_n, pr(w_n, d)) \right\} \quad (3)$$

where w_i is the text value of term i in the document d whereas $pr(w_i, d)$ is the weight of term w_i . A list of categories of the corpus C is

$$C = \{c_1, c_2, \dots, c_m\} \quad (4)$$

5.1 Graph-based Ranking Approach and Inverse Document Frequency Measure (*pr.idf*)

The popular method based on term frequency is *tf.idf*, in which the weighting of a term w_i in a document d_j from the corpus C is calculated as:

$$tf.idf(w_i, d_j) = tf(w_i, d_j) * idf(w_i) \quad (5)$$

$$tf(w_i, d_j) = \frac{freq(w_i, d_j)}{\sum_{k=1}^n freq(w_k, d_j)} \quad (6)$$

$$idf(w_i) = \log \left(\frac{|C|}{|d : w_i \in d|} \right) \quad (7)$$

The idea of *tf.idf* is that term frequency is represented for the importance of a term in a document, and the inverse document frequency is represented for the importance of the term in its corpus. Adapting with the idea of *tf.idf*, we propose *pr.idf* term weighting measure, which takes the advantages of *idf* and *pr*. Instead of using *tf*, we have been calculating the weighting values based on graph model *pr* as the importance of the terms in the document.

Thus, the formula of weighting a term w_i of the document d_j in the corpus C as below:

$$pr.idf(w_i, d_j) = pr(w_i, d_j) * idf(w_i) \quad (8)$$

where $pr(w_i, d_j)$ is a importance value of w_i from the document d_j . The *pr.idf* value of each term will be filled to the feature vector for classifying task.

5.2 Graph-based Ranking Approach and Term Category Dependence Measure (*pr.tcd*)

The idea of term category dependence measure (*tcd*) is that terms represented for a document are dependence to its categories. Recognising the benefits of

term category dependency, a measure *tcd* has been proposed, which takes into account the degree of belonging of a term to a particular category. If a word occurs frequently in many documents of one class, and never or infrequently occurs in other classes, it is considered as a representative of the class if its ranking value from the document is also comparable. We suggest a measure of degree of belonging of the term w_i to the category c_i

$$tcd(w_i, c_j) = \frac{tf(w_i, c_j) * df(w_i, c_j)}{\sum_{k=1}^m (tf(w_i, c_k) * df(w_i, c_k))} \quad (9)$$

c_k is a categories of the corpus C

The combination of term ranking on documents (*pr*) and term category dependence (*tcd*) presents a new method for weighting a term w_i from a document d_j that belongs to a original category or predicted category c_k as follows:

$$pr.tcd(w_i, d_j) = pr(w_i, d_j) * tcd(w_i, c_k) \quad (10)$$

the *pr.tcd* value of each term will be added to the feature vector for classification task.

6 Experiments

6.1 Comparison Term weighting methods

From the previous sections, we have presented our proposed term-weighting methods *pr.tcd* and *pr.idf*. It can be seen that *pr.idf* is a combination of our calculation based on the graph *pr* and *idf*. The purpose of *pr.idf* is to show how effective between *tf* and *pr* measures when making the comparison between *tf.idf* and *pr.idf*. Moreover, when making the comparison between *pr.tcd* and *pr.idf*, we can clearly see how effective among *tcd* and *idf* measures.

Similarly, the second combination *tf.tcd* presented below gives another aspect between *tf* and *pr* when making the comparison between *tf.tcd* and *pr.tcd*.

$$tf.tcd(w_i, d_j) = tf(w_i, d_j) * tcd(w_i, c_k) \quad (11)$$

With those methods based on the dependencies between terms and categories (*pr.tcd*, *tf.tcd*), the testing data does not have information of categories. Thus, a strategy is suggested to obtain the initial categories for calculating *tcd* values from the testing set. The initial categories are a predicted label set returned by using a text classifier to predict the labels of *tf.idf*-based feature vectors from the testing set. The classifier is enriched by *tf.idf*-based training model from the training set. Once the *tcd* values are identified for every single term of the testing set, the formulas (10) or (11) will be used to estimate the weighting of features.

6.2 Data set

The evaluation data set is the collection Wikipedia XML Corpus compiled by Ludovic & Patrick (2006). The reason to choose Wikipedia corpus as the evaluation data is that it is considered not only as universal unstructured text corpus containing very large numbers of multi-theme documents but also as a high standard grammar corpus.

There are many sub-collections of this corpus. We have chosen the English Single-Label Categorisation Collection, which provides each document that belongs to one single category. With the purpose of

demonstrating the effects of our approach, we randomly select part of the corpus for our experiment. As our approach is based on the context of sentences to extract the desirable information, we ignore documents containing uncompleted sentences. All documents are selected from the original Wikipedia corpus if their sizes are greater than *1kb*. As a result, after eliminating some categories that contain very small number of documents (less than 10 documents), we obtain a total of 8502 documents assigned to 53 categories. These documents are divided randomly and equally to form a training data set and a test data set including 4251 documents and 53 categories in each set. The number of documents for each larger category is 100, and the number of documents for each smaller category is 12.

The pre-processing procedures for the raw text from the training and test data are performed in the same way. From those methods based on term frequency (*tf.idf*, *tf.tcd*), the raw input text is firstly tokenised and eliminated stop-words. The remaining information is pushed to these term weighting methods to calculate the ranking of representation of documents. The outcome of these methods is the list of term representatives of documents. Before starting to evaluate the effectiveness of these methods by classification techniques, it is necessary to reduce the number of features. We are using the simplest method to eliminate the number of features: document frequency. We try not to use those terms as features if they appear in less than 3 different documents in the corpus³. By doing all the aforementioned eliminations, the number features of the term-based methods have been reduced from 103,408 to 24,075 for training set and 85,575 from 16,191 for test set, whereas the numbers of features from the graph-based methods were decreased from 73,529 to 17,277 for the training set and from 61,261 to 11,346 for the test set.

6.3 Classifiers

Support Vector Machines (Vapnik 1995) is a state-of-the-art machine learning approach based on decision plans. The algorithm defines the best hyper-plan, which separates set of points associated with different class labels with a maximum-margin. The unlabelled examples are then classified by deciding in which side of the hyper-surface they reside. The hyper-plan can be a simple linear plan or a non-linear plan such as polynomial, radial, or sigmoid. In our evaluation we used the linear kernel since it was proved to be as powerful as the other kernels when tested on text classification data sets (Yang & Liu 1999).

6.4 Performance & Evaluation measures

To evaluate the classification system we use the popular accuracy measure defined as the number of correct predictions divided with the number of evaluated examples. Four weighting models that need to be tested are *tf.idf*, *tf.tcd*, *pr.idf* and *pr.tcd*.

Corpus	<i>tf.idf</i>	<i>tf.tcd</i>	<i>pr.idf</i>	<i>pr.tcd</i>
Wikipedia	72.7%	77.2%	73.8%	81.8%

Table 1: SVM results on Wikipedia corpus

By examining the SVM text classification results from *table 1* with four different term weighting mod-

³based on experiment reported by (Joachims 1998)

els, it can be seen clearly that *pr.tcd* model provides the highest accuracy and the *tf.idf* model achieves the lowest accuracy when testing in the same Wikipedia data.

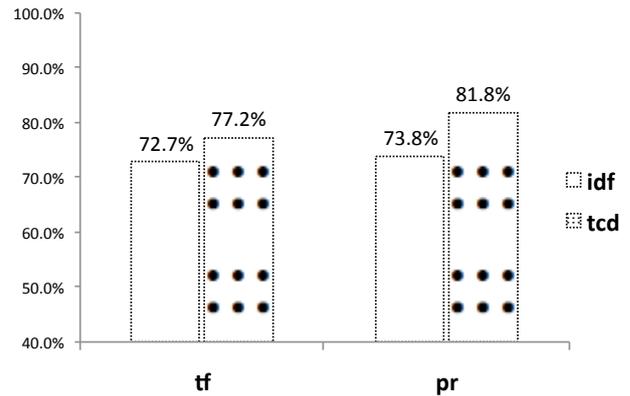


Figure 6: The chart shows the accuracy comparison of four different weight model *tf.idf*, *tf.tcd*, *pr.idf*, *pr.tcd*

6.4.1 Term Graph Weighting versus Term Frequency Weighting

The graph from *figure 6* shows the comparison the effects of graph-based methods and term-based methods.

It is possible to conclude the graph model produces the more reliable text representation for the given text document than the traditional term frequency. Firstly, from the view of *tcd*, we are considering every pair methods *pr.tcd* and *tf.tcd* from the chart, the weighting method based on the graph model (*pr.tcd*) achieves 81.8%, which outperforms the one based on term frequency (*tf.tcd*) archiving 77.2%. Secondly, from the view of *idf*, when making the comparison between *tf.idf* and *pr.idf*, it can be seen clearly that the graph-based method (*pr.idf*) dominates at 73.8%, which is higher at least 1% than the *tf.idf* method (72.7%).

6.4.2 Inverse Document Frequency versus Term Category Dependency

The information from the chart of *figure 6* shows another view of information. It presents the comparison between the contribution of *idf*-based method and *tcd*-based methods to the performance of TC tasks.

The accuracy results have confirmed that all models taking the consideration of the dependency among terms and categories (*tf.tcd*, *pr.tcd*) yield the higher accuracy results than others based on document frequency (*tf.idf*, *pr.idf*) 77.2% vs. 72.2% and 81.8% vs 73.8%, respectively. It is also possible to conclude the *tcd*-based methods are more effective than the *idf*-based methods in text classification.

Moreover, the graph from *figure 5* has also reflected the correlations among the proposed methods and the bottom-line (*tf.idf*) method for text classification tasks. There are 53 classes from the training and testing data for all tests. Every single line from the graph represents the accuracy of a particular weighing method in comparison with the others.

In short, the overall classification results have persuaded that the proposed methods provide outstanding results in comparison to the popular method (*tf.idf*) and can be considered as potential methods for further investigations.

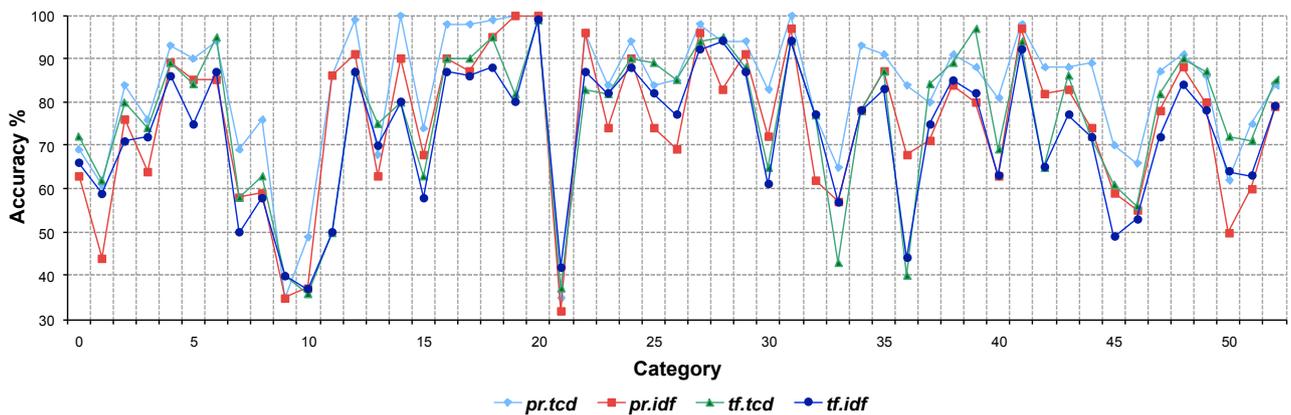


Figure 5: The chart shows correlation descriptions of four different term weighting methods *tf.idf*, *tf.tcd*, *pr.idf*, *pr.tcd* from the view of categories

7 Conclusion and future work

The paper has presented a method for document representation based on relation extraction and graph model, which improve accuracy of text classification in comparison to the popular term weighting methods. Our approach overcomes the lack of frequency information by self-creating the frequency based on the structure of text content. This is also the motivation for our further investigation on the benefits of relations on text classification as well as text mining.

We notice that although our approach uses the concept of relations, we still do not take the closed consideration on its semantic aspect, we only use it as a first attempt for getting more statistical information. For further investigation, we are more focusing on taking the semantic information from tuples and its connection from the graph to form representations of given documents. The expectation approaches can be used as objectives of semantic classifications.

References

- Agichtein, E., Eskin, E. & Gravano, L. (2000), Combining strategies for extracting relations from text collections, in 'Proc. of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery'.
- Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M. & Etzioni, O. (2007), Open information extraction from the web, in 'Proc. of IJCAI', pp. 2670–2676.
- Brin, S. (1998), Extracting patterns and relations from the world wide web, in 'Proc. of the 1998 International Workshop on the Web and Databases', pp. 172–183.
- Bunescu, R. C. & Mooney, R. J. (2005), A shortest path dependency kernel for relation extraction, in 'Proc. of Human Language Technology Conference and Conference on Empirical Methods in Natural Language', pp. 724–731.
- Downey, D., Etzioni, O. & Soderland, S. (2005), A probabilistic model of redundancy in information extraction, in 'Proc. of the 19th IJCAI', pp. 1034–1041.
- Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A., Shaked, T., Soderland, S., Weld, D. & Yates, A. (2004), Web-scale information extraction in knowitall:(preliminary results), in 'Proceedings of the 13th international conference on World Wide Web', ACM, pp. 100–110.
- Gabrilovich, E. & Markovitch, S. (2007), Computing semantic relatedness using wikipedia-based explicit semantic analysis, in 'Proc. of the 20th IJCAI', pp. 1606–1611.
- Hassan, S. & Banea, C. (2006), Random-walk term weighting for improved text classification, in 'Proc. of TextGraphs', pp. 53–60.
- Hu, J., Fang, L., Cao, Y., Zeng, H.-J., Li, H., Yang, Q. & Chen, Z. (2008), Enhancing text clustering by leveraging wikipedia semantics, in 'Proc. of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', pp. 179–186.
- Joachims, T. (1998), Text categorisation with support vector machines: Learning with many relevant features, in 'Proc. of the 10th ECML', pp. 137–142.
- Lawrence, P., Sergey, B., Rajeev, M. & Terry, W. (1998), The pagerank citation ranking: Bringing order to the web, in 'Stanford Digital Library Technologies Project'.
- Ludovic, D. & Patrick, G. (2006), The wikipedia xml corpus, in 'ACM SIGIR Forum', pp. 64–69.
- Minnen, G., Carroll, J. & Pearce, D. (2001), Morphological processing of english, in 'Natural Language Engineering', pp. 207–223.
- Rada, M. & Paul, T. (2004), Textrank: Bringing order into texts, in 'Proc. of the EMNLP'.
- Ravi, S. & Rada, M. (2007), Unsupervised graph-based word sense disambiguation using measures of word semantic similarity, in 'Proc. of the ICSC', pp. 363–369.
- Robertson, S. & Jones, K. S. (1997), Simple, proven approaches to text retrieval, Technical report, University of Cambridge.
- Strube, M. & Ponzetto, S. P. (2006), Wikirelate! computing semantic relatedness using wikipedia, in 'Proc. of the 21st AAAI', pp. 1419–1424.

- Vapnik, V. N. (1995), The nature of statistical learning theory, *in* 'Springer'.
- Wang, P., Hu, J., Zeng, H.-J., Chen, L. & Chen, Z. (2007), Improving text classification by using encyclopaedia knowledge, *in* 'The Seventh IEEE ICDM', pp. 332–341.
- Wang, W., Do, D. B. & Lin, X. (2005), Term graph model for text classification, *in* 'Proc. of ADMA', pp. 19–30.
- Yang, Y. & Liu, X. (1999), A re-examination of text categorisation methods, *in* 'Proc. of the 22nd Annual International ACM SIGIR conference on Research and Development in Information Retrieval', pp. 42–49.
- Yang, Y. & Pedersen, J. O. (1997), A comparative study on feature selection in text categorisation, *in* 'Proc. of the 14th ICML', pp. 412–420.
- Yeh, E., Ramage, D., Manning, C. D., Agirre, E. & Soroa, A. (2009), Wikiwalk: random walks on wikipedia for semantic relatedness, *in* 'Proc. of TextGraph-4', pp. 41–49.
- Yu, S. & Zhang, J. (2009), A class core extraction method for text categorisation, *in* 'Proc. of the 6th FSKD', pp. 3–7.

Comparison of Binary and Multi-Variate Hybrid Decision Diagram Algorithms for K -Terminal Reliability

Johannes U. Herrmann and Sieteng Soh

Department of Computing
Curtin University of Technology
GPO Box U1987, Perth 6845, West Australia

jherrmann@ieee.org

Abstract

The Ordered Binary Decision Diagram (OBDD) has been efficiently used to compute the communication network (CN) reliability (REL). The boundary set method (BS) is used to improve the efficiency of the OBDD approach, while the augmented OBDD diagrams (OBDD-A) that stores CN information in its nodes has been proposed to solve other CN performance metrics in addition to REL. The hybrid OBDD (OBDD-H) combines the BS and OBDD-A features to further improve the performance of the OBDD method. However, both BS and OBDD-H address only CN with communication link (edge) failure. In this paper, we generalize OBDD-H for networks with edge and/or device (vertex) failures. We also present a hybrid ordered multi-variate decision diagram (OMDD-H) to compute the performance metrics of CN with vertex and link failures. This paper examines the time and space complexities of OBDD-H, and shows that OMDD-H can compute the REL in CN with fallible vertices and edges in the same order of complexities as BS or OBDD-H computing the same network with only vertex failure.

Keywords: boundary set, decision diagram, network reliability, time and space complexity.

Acronyms

ALL-REL – All terminal reliability – As reliability ($q.v.$) but with the requirement that all devices in the network be connected.

BS – Boundary Set – A way to store information on the state of a network; in this work BS refers to the method of using a boundary set and an OBDD $q.v.$ to compute network reliability.

CN – Communication Network – A wired or wireless network where communication occurs between source and target devices.

DD – Decision Diagram – A structure used to compute network reliability.

K -REL - K terminal reliability – As reliability ($q.v.$) but with the requirement that a set K of devices in the network be mutually connected.

OBDD – Ordered Binary Decision Diagram - A DD whose binary variables are decided in a fixed order.

OBDD-A – Augmented OBDD – An OBDD with information on network paths stored in each node.

OBDD-H – Hybrid OBDD – An OBDD with information on boundary set connectivity stored in each node; a hybrid of the OBDD-A and BS methods.

OMDD – Ordered Multi-variate DD - As OBDD but without the requirement that all variables be binary. There are also multi-variate versions of the OBDD-A and OBDD-H.

REL – Reliability – The probability that the source and target vertices are connected by active devices and communication links.

WSN – Wireless Sensor Network – A communication network in which the devices are sensor devices which communicate with each other in a wireless manner and co-operate to route transmissions to target devices.

1 Introduction

As the use of both wired and wireless communication networks (CNs) increases, their reliability is of importance for both their design and analysis. When components of a network fail, parts of the network may become unavailable and/or performance can degrade. The reliability (REL) of networks has been studied extensively (Hardy *et al.*, 2005, Hardy *et al.*, 2007, Herrmann and Soh, 2009, Herrmann *et al.*, 2009, Herrmann, 2010, Herrmann *et al.*, 2007, Yeh *et al.*, 2002, Carlier and Lucet, 1996) and methods utilizing ordered binary decision diagrams (OBDD) have been shown to be superior to other methods in general (Hardy *et al.*, 2007).

The boundary set OBDD method (BS) introduced by Hardy, *et al.* (2005) uses edge contraction to construct an OBDD which can be used to compute all terminal reliability (ALL-REL) (Hardy *et al.*, 2005) and K -terminal reliability (K -REL) (Hardy *et al.*, 2007) for the network. BS was shown (Hardy *et al.*, 2005) (Hardy *et al.*, 2007) to be more efficient than other methods available at the time, such as the EED_BFS method proposed by Yeh, *et al.* (2002).

A disadvantage of BS is that it first generates the entire OBDD and then uses this to generate REL, which means that all OBDD nodes must be stored in memory. For large networks this can require that millions of nodes must be stored (Hardy *et al.*, 2007). In addition, the partition numbers used in BS can quickly become larger than native data types in programming languages (*e.g.*, C) can store, requiring the use of computationally expensive libraries that permit the use of numbers of arbitrary size.

Further, BS can be used only for computing the REL of undirected networks whose communication links can fail but whose devices (vertices) are not susceptible to failure.

A network model with fallible communication links but perfect devices is not always appropriate. Wireless networks may suffer both from device failure and the interruption of communication signals (Akyildiz *et al.*, 2002); indeed this holds true for wired networks as well. For example, a mobile telephone communication can fail if the phone battery runs out (device failure) or the phone is taken into a tunnel (link failure).

The augmented ordered decision diagram (OBDD-A) was proposed by Herrmann, *et al.* (2007) to solve K -REL and a number of other metrics (e.g., the expected hop count – EHC). The OBDD-A stores additional information (e.g., the state probability) in diagram nodes, allowing nodes to be discarded after use and the metrics to be computed without traversing the diagram a second time. Information was stored by recording which devices were connected to the source device(s) and tracking any existing paths that had not yet been utilized.

Both the BS and OBDD-A methods decrease greatly in performance when computing REL for networks with both link and device failure as compared to device (or link) failure only. The augmented ordered multi-variate decision diagram (OMDD-A) was introduced (Herrmann *et al.*, 2009) as an extension to OBDD-A. It was shown that computing REL and EHC using OMDD-A is comparable in runtime performance to the same computation for a network with only device failure using OBDD-A on networks with only vertex failures.

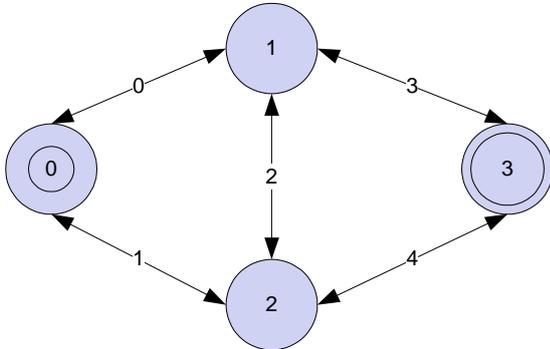


Figure 1: Sample Network

While the augmented diagram methods are able to solve more general networks (e.g., directed networks or those with edge and/or vertex failure) its execution time was far inferior to that of the BS method.

The OBDD-A and BS methods were combined (Herrmann and Soh, 2009) into the hybrid ordered binary decision diagram (OBDD-H) which stored the partition numbers of BS partitioning in diagram nodes. This method was shown empirically to have comparable execution time performance to BS while requiring far less memory (Herrmann and Soh, 2009). However the OBDD-H is also restricted to undirected networks and does not account for vertex failure. In addition, early versions of OBDD-H stored partition numbers, requiring the same libraries as BS for large numbers. This was later modified (Herrmann, 2010) to store partition information directly instead of using partition numbers. This

modification decreases running times while maintaining the low space usage of the earlier version.

Our contributions in this paper are twofold. First, we generalize the OBDD-H method to solve networks with vertex and/or edge failures. Then, we propose a multi-variate hybrid decision diagram (OMDD-H) to improve the performances of both OBDD-H and OMDD-A. Our OMDD-A can be used in networks with vertex and link failures.

The rest of this paper is organized as follows. Section 2 presents the necessary network models and terminology, and reviews the BS and OBDD-H methods. The OBDD-H is modified for vertex and edge failure in Section 3 and its performance analysed. We introduce the OMDD-H in Section 4 and summarize the work in Section 5.

2 Background

2.1 Network Model

An undirected network η is modelled using an undirected graph $G = (V, E)$ whose vertices, V , represent the communication devices of η and whose edges, E , represent the communication links of η . If devices are fallible, each vertex $v_i \in V$ has probability $P(v_i)$ of being available. Similarly if communication links are fallible, each edge $e_i \in E$ has probability $P(e_i)$ of being active. If these probabilities are not known they can be estimated as described in (Colbourn, 1987). Each edge between vertices v_x and v_y is written as (v_x, v_y) or simply (x, y) where $x < y^1$.

A network whose devices do not fail is modelled by a graph with perfect vertices (i.e., $\Pr(v) = 1.0 \forall v \in V$). Similarly a network whose communication links do not fail is modelled by a graph with perfect edges (i.e., $\Pr(e) = 1.0 \forall e \in E$). For this work, when a device can fail we assume it does so with a probability of 0.1 (i.e. it has a probability of 0.9 of being active).²

Depending on the connectivity model in use, each network has one or more source vertices as well as one or more target vertices. For the case where we require a set K of vertices to be connected we choose one of these vertices as the source and the others as targets.

For ordered decision diagrams, networks must be ordered. We use a breadth-first ordering for the graph representing each network which orders vertices in increasing distance from the source(s) (except that at least one of the vertices in K must be the last vertex in the ordering) and edges in increasing order of lower and then higher endpoints. As an example, consider the network given in Figure 1, with the source vertex v_0 and target v_3 . The vertices have been labelled using the ordering given above. Note that switching the labelling of vertices v_1 and v_2 would have been equally correct since both are the same distance from the source vertex. Given the vertex

¹ Strictly speaking this should be $x \leq y$ to allow for self loops. However such loops do not affect the reliability of the network and can hence be ignored.

² The algorithm functions with other component probabilities; however this is the standard assumption in the literature.

ordering there is only one correct ordering of edges, which is shown in their labelling.

A network state $\Omega=(V_\Omega, E_\Omega)$ of network $G=(V, E)$ is a partition of G such that all vertices in $V_\Omega \subseteq V$ and edges in $E_\Omega \subseteq E$ are available and all other vertices in V and edges in E are unavailable. The probability of state Ω is computed as:

$$\Pr(\Omega) = \sum_{e_i \in E_\Omega} \Pr(e_i) \sum_{e_j \notin E_\Omega} (1 - \Pr(e_j)) \sum_{v_i \in V_\Omega} \Pr(v_i) \sum_{v_j \notin V_\Omega} (1 - \Pr(v_j))$$

For K -terminal reliability (K -REL), Ω is a successful state if $K \subseteq V_\Omega$ and each vertex in K is connected to all vertices in K by a path of vertices in V_Ω and edges in E_Ω . REL can be calculated by summing the probabilities of the success states of the network.

Since each edge and vertex can be in one of two states (available or unavailable), there are $2^{|\mathcal{E}|+|\mathcal{V}|}$ states for network G , and therefore K -REL cannot be solved for large networks through state enumeration.

2.2 Boundary Set Algorithm

Boundary sets were introduced for network reliability by Carlier and Lucet (1996). This method was combined with an OBDD into the BS method by Hardy *et al.* to efficiently solve ALL-REL (2005) and K -REL (2007).

BS first builds an OBDD and then traverses this diagram to compute REL. Each level of the OBDD represents the evaluation of edge e_k of the network. Each node N_i has two children; a positive child representing e_k being available and a negative child representing e_k being unavailable. We refer to the variable (in this case e_k) being decided on level k of the diagram as the *decision variable* of that level.

BS utilizes the boundary set (F_k) of each level k of the diagram; that is the set of vertices that are of importance to the algorithm at each stage. F_k (deciding edge e_k) is defined as:

$$F_k = \{v_x: v_x \text{ is an endpoint of edges } e_y \text{ and } e_z \text{ with } y \leq k \text{ and } z \geq k\}$$

Each node of the OBDD represents a state of the network, and is encoded by a partitioning of the boundary set. Each partition represents a subset of F_k that is connected; in other words each vertex in the partition is connected to each other vertex by some path.

When computing K -REL, we want all vertices in K to end up in one partition, showing that they are connected. A node that has such a partition is a success node and a node in which such a partition can no longer exist is a failure node. The connection to K is tracked by marking the partition that is connected to the source vertex with an asterisk. If a vertex in K is disconnected from the marked partition the node is failed.

The BS algorithm is shown in Figure 2. The algorithm presented here is the one presented by Herrmann and Soh (2009). The algorithm proposed by Hardy *et al.* (2007) does not allow for the failure of positive child nodes (lines 11-12).

The creation of child nodes uses *edge contraction* and *edge deletion*. For a positive child, edge e_k is contracted, merging its endpoints. For a negative child, e_k is deleted,

possibly leaving the endpoints disconnected. Vertices that are in F_k but not in F_{k+1} are removed from both child

```

1. Create root node  $N_2$ 
2.  $F_0 = \{v_0\}$ , and  $REL \leftarrow 0$ .
3. for  $k = 1$  to  $|\mathcal{E}|$  do
4.   compute  $F_{k+1}$ .
5.   for each  $N_i$  on level  $k$  do
6.     translate partition number  $i$  into parti.
7.     Create negative partition part0.
8.     Create positive partition part1.
9.     if part1 is successful then
10.      positive child of  $N_i$  is 1.
11.     else if part1 is failed then
12.      positive child of  $N_i$  is 0.
13.     else
14.      translate part1 to number  $j$ .
15.      if  $N_j$  is not in hash table then
16.        create  $N_j$  and insert into hash table.
17.      positive child of  $N_i$  is  $N_j$ .
18.     if part0 is failed then
19.      negative child of  $N_i$  is 0.
20.     else
21.      translate part0 to number  $j$ .
22.      if  $N_j$  is not in hash table then
23.        create  $N_j$  and insert into hash table.
24.      negative child of  $N_i$  is  $N_j$ .
    
```

Figure 2: BS Algorithm

nodes. If such a vertex removal results in an empty partition, this is also removed; when such a partition is removed the child node is *failed* because one or more of the K vertices are disconnected from others in K (lines 11-12 and 18-19).

This partitioning is itself encoded into a partition number, which is stored and can be transformed back into the partitioning when needed. The process makes use of Stirling numbers of the second kind (A_{ij}). Details of this process can be found in (Hardy *et al.*, 2005) and (Hardy *et al.*, 2007) for ALL-REL³ and in (Herrmann and Soh, 2009) for K -REL.

One of the strengths of the ordered decision diagram is that isomorphic nodes (those that have identical subtrees) are merged; reducing the number of nodes that need to be stored and processed. This can be seen in Figure 3 where any node with multiple arrows entering is a merged node; each arrow represents one merged node.

The use of boundary sets makes detecting isomorphic nodes simple. Any two nodes on the same level that have an identical partitioning of the boundary set (and hence identical partition number) are isomorphic.

The OBDD created by BS working on the sample network is shown in Figure 3 with the assumption that $K = \{0, 3\}$. Solid arrows indicate the positive child and dashed arrows the negative. The shaded non-terminal nodes represent states which are neither succeeded nor

³ Hardy *et al.* (2007) do give an algorithm to compute the partition numbers for K -REL but this is erroneously identical to the ALL-REL version and does not consider marked partitions.

failed. Each such state includes the partitioning that it represents along with the associated partition number (in parentheses) and the probability of being in this state. Note that the partition itself is not stored by BS and the partition number is stored in a separate hash table instead of in the OBDD. Finally the probabilities are not stored in the node but also stored in the hash table when they are computed. The boundary set for each level is shown on the left and the edge being decided is shown on the right.

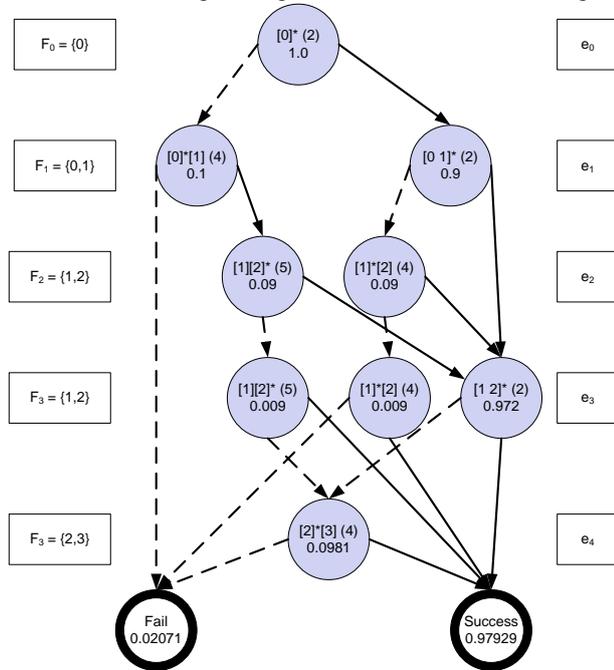


Figure 3: OBDD for Sample Network

The performance of BS was shown to be closely related to the size of the boundary sets; in particular the size of the maximum boundary set (F_{max}). The time and space complexity of the BS method were both shown to be bounded by

$$O(|E| \times (F_{max})^3 \times B_{F_{max}})$$

where $B_{F_{max}}$ is the Bell number defined by

$$B_{|F_{max}|} = \sum_{j=1}^{|F_{max}|} A_{|F_{max}|,j}$$

2.3 The OBDD-He Algorithm

The OBDD-H was introduced by Herrmann and Soh (2009) for undirected graphs for which edges can fail but vertices are perfect. These conditions are identical to those for BS because the algorithms are closely related. Let us refer to this version of OBDD-H the OBDD-He. Note, in Section 3.1, we present OBDD-Hv for undirected graphs with failed vertices and perfect edges, and in Section 3.2 we describe OBDD-Hve that generalizes OBDD-He and OBDD-Hv for use in networks with failed vertices and/or edges.

Like BS, the OBDD-He uses partitions of F_k to encode the network state. While the OBDD-He (Herrmann and Soh, 2009) initially translated these into partition numbers it was shown empirically (Herrmann, 2010) that it is more efficient to store the partitions directly as structures. In either case, the information is stored directly in each OBDD-He node, either as partition

numbers or structures. The OBDD-He for the sample network is shown in Figure 5 and the algorithm is shown in Figure 4.

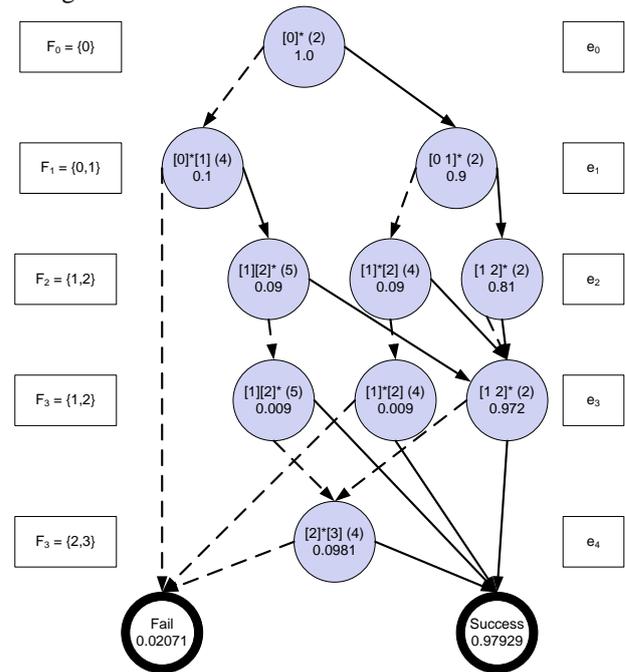


Figure 5: OBDD-He for Sample Network

Also stored in each OBDD-He node is the probability of being in the state represented by that node (and hence the partitioning of F_k stored in that node). BS instead stores this probability and the partition number in a separate hash. The root node of the OBDD-He has probability 1.0 and each child node's probability is computed from that of its parent by multiplying by $P(e_k)$ is available (for the positive child) or unavailable (for the negative).

Most nodes in the OBDD-He are exactly equivalent to

1. Create root node N_0
2. $Q_C \leftarrow \{N_0\}$, $Q_N \leftarrow \{ \}$, $k \leftarrow 0$, and $REL \leftarrow 0$.
3. **while** ($Q_C \neq \{ \}$ **or** $Q_N \neq \{ \}$)
4. **if** $Q_C = \{ \}$ **then**
5. $Q_C \leftarrow Q_N$, $Q_N \leftarrow \{ \}$ and $k \leftarrow k + 1$.
6. $F_k \leftarrow F_{k+1}$, Compute F_{k+1}
7. **for each** N_i on Q_C .
8. Create negative child
9. Create positive child for e_k
10. **for each child** N_j
11. **if** N_j is non-terminal **then**
12. **for each** $N_q \in Q_N$ **do**
13. **if** N_j is isomorphic to N_q **then**
14. merge N_j into N_q
15. **break.**
16. **if no** N_q was isomorphic to N_j **then**
17. add N_j to Q_N .
18. **else if** N_j is a success node **then**
19. $REL \leftarrow REL + Pr(N_j)$.
20. delete N_i

Figure 4: OBDD-H for Edge Failure

the comparable node in BS since they have the same partition information. Children of each node are

equivalent since these are created through the same process. When two nodes in the OBDD-He are isomorphic and are merged, the equivalent nodes in BS are also isomorphic and merged.

The difference between the algorithms is that the OBDD-He nodes are not actually linked into a diagram; they are instead stored in queues representing levels k and $k+1$ of the diagram. Nodes on level k are processed and non-terminal child nodes are either merged with an isomorphic node on level $k+1$ or added to that queue if no isomorphic node exists. Once the child nodes are created the parent node is discarded.

This difference means that the OBDD can be optimized after being created; in particular if both children of a node are isomorphic (and hence merge) the parent node is redundant. This process does not occur for the OBDD-He, as can be seen by the extra node on level 2 of the diagram. It should be noted that this missing node is still created by OBDD and processed, and thus affects time complexity. Because it is subsequently merged it does not affect the space complexity, however.

2.4 The Complexity of OBDD-He

The OBDD-He uses the same methods as BS for generating and merging diagram nodes. For this reason, the structure and number of nodes of both the OBDD generated with BS and the OBDD-He are identical.⁴

Since the time complexity of both algorithms is directly related to the number of diagram nodes generated, the OBDD-He has the same order of time complexity as BS. However the space complexity is not identical.

The OBDD-He discards nodes once they have been processed. This means that nodes from no more than two levels are stored in memory at any one time. This indicates that the space complexity for OBDD-He is bounded by

$$O((F_{max})^3 \times B_{F_{max}}).$$

When a series of networks increase in size but retain a constant inter-connectivity, it was found (Herrmann and Soh, 2009) that the space complexity had a constant upper bound. The constant inter-connectivity of the networks means that F_{max} is bounded by some constant Γ . Hence the bound for the space complexity becomes $O(\Gamma^3 \times B_{\Gamma})$, which is constant and thus verifies the experimental results.

3 The Generalized OBDD-H

3.1 OBDD-H for Vertex Failure (OBDD-Hv)

If vertices fail instead of edges, each level of the decision diagram represents the evaluation of a vertex, which could lead to a number of edges being contracted. For the worst case, each level will represent only one edge being contracted but in general there will be more than one. The

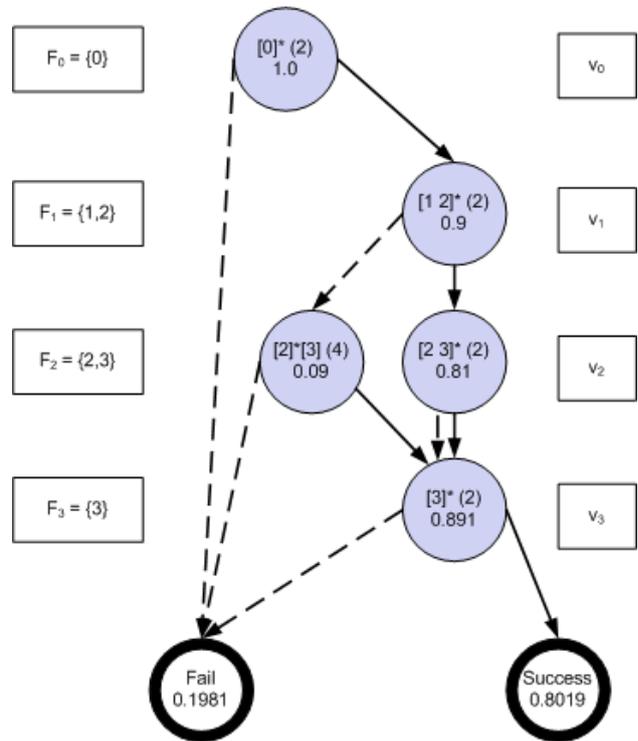


Figure 6: OBDD-Hv of Sample Network

resulting nodes will be equivalent to having performed the contraction of each of the edges associated with the vertex, in turn.

The OBDD-Hv for the sample network is shown in Figure 6, with F_k and the decision variable for each level k shown on the left and right of the diagram respectively. Note that this is smaller than the diagram created by OBDD-He (Figure 5) and BS (Figure 3). Indeed the diagram for OBDD-Hv is a subgraph of the larger diagrams.

The process for constructing the OBDD-Hv is similar to that for the OBDD-He except that we effectively perform multiple edge contractions or deletions at every level; one for each edge adjacent to the vertex being decided. Hence the child nodes for OBDD-Hv are equivalent⁵ to the nodes in BS where either all adjacent vertices are available (the positive child) or unavailable (the negative child). Hence each level of the OBDD-Hv has the same complexity constraints as for an OBDD-He. However the number of levels of the OBDD-Hv is equal to the number of vertices instead of the number of edges. We deduce that the time complexity of the OBDD-Hv is:

$$O((F_{max})^3 \times B_{F_{max}}).$$

The space complexity of OBDD-Hv is identical to OBDD-He since there are still at most two levels of nodes stored in memory at any one time.

⁴ Given the constraints that certain nodes are removed from the BS OBDD after being processed. These nodes do not affect the worst-case space complexity since the worst case assumes that there are a maximal number of nodes at each level.

⁵ The partitions are equivalent for both nodes, but the probabilities of being in the state represented by the nodes are obviously not equal.

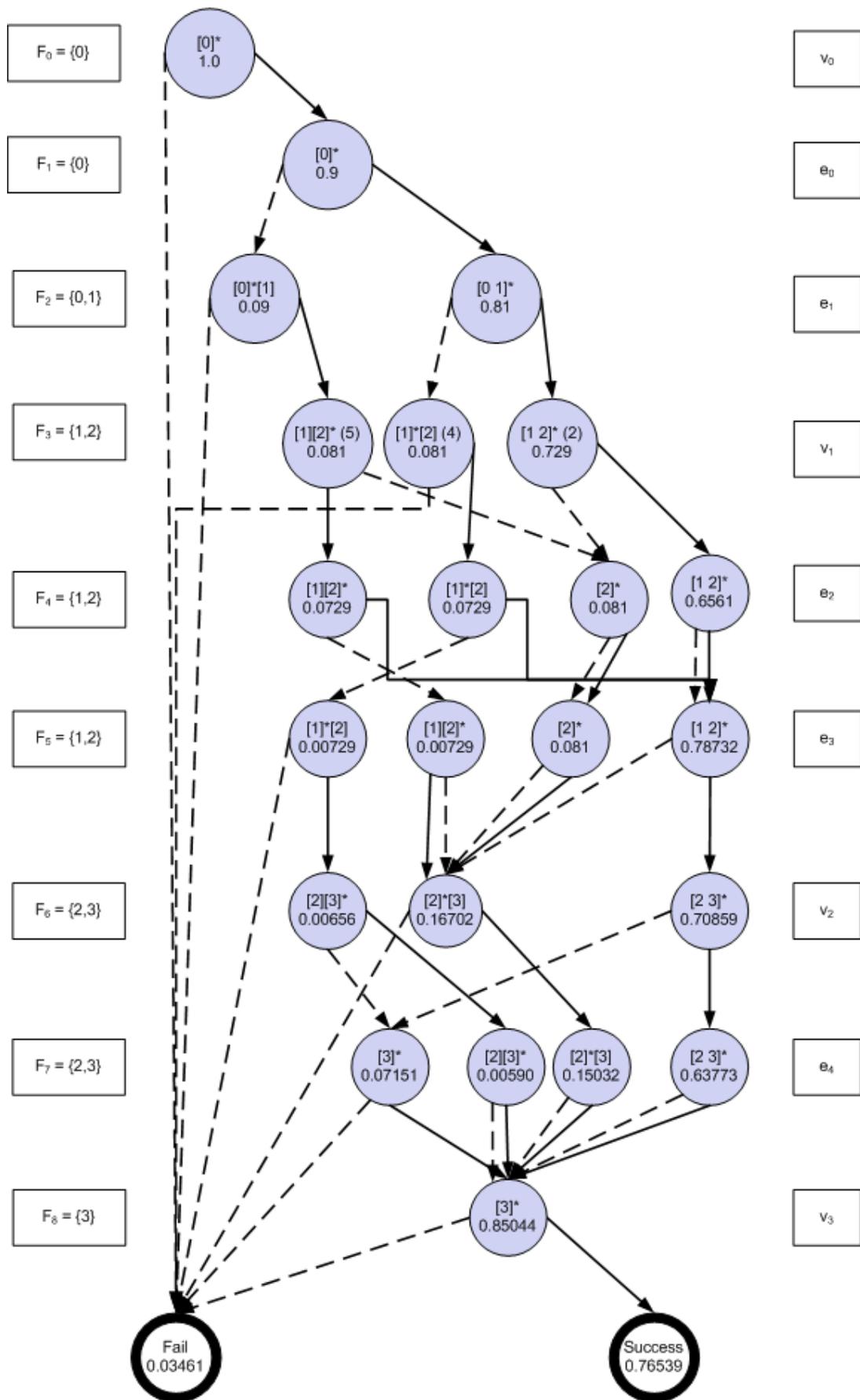


Figure 7: OBDD-Hve for Sample Network

3.2 OBDD-H for Vertex and Edge Failure

When both vertices and edges are fallible each must be decided in turn. We decide each vertex in turn, and decide each edge immediately after the lower endpoint. So for the sample network shown in Figure 1 the variable ordering is $v_0, e_0, e_1, v_1, e_2, e_3, v_2, e_4, v_3$. The resulting OBDD-Hve is shown in Figure 7. Again, F_k for each level is shown on the left of the diagram and the decision variable is shown on the right. Note that the probabilities shown are rounded to five decimal places where needed.

Note that for the negative child when deciding a vertex we remove the unavailable vertex from its partition. While this means that the result is, strictly speaking, not a complete partitioning of F_k anymore it saves unnecessary computation in the following levels. This means that partition numbers are no longer appropriate, but since OBDD-H does not use them this is not an issue.

Note that the last level of both OBDD-Hv and OBDD-Hve will be a single node containing the partition of the last vertex in the ordering. In some applications it is assumed that the target vertex is always available, in which case the ordering is modified to ensure that the target is last and this level can be omitted. We do not make this assumption in this work, and hence retain the final level.

The depth of the OBDD-Hve means that the number of non-terminal nodes are greater than for both the OBDD-He and OBDD-Hv combined.

The time and space complexities for OBDD-Hve can be similarly deduced to be

$$O((|V| + |E|) \times (F_{max})^3 \times B_{F_{max}})$$

and

$$O((F_{max})^3 \times B_{F_{max}}),$$

respectively, when each vertex and edge is decided in turn. It can be seen that the time required to compute OBDD-Hve is considerably larger than either OBDD-Hv and OBDD-He. The amount of memory required is not greatly affected, however.

4 The Multivariate Hybrid Decision Diagram Algorithm

4.1 The OMDD-A

The augmented ordered multi-variate decision diagram (OMDD-A) was introduced (Herrmann *et al.*, 2009) for solving both REL and the Expected Hop Count problems for networks with both device and link failure. The OMDD-A groups each vertex with any adjacent edges that have not yet been grouped with another vertex. Each level of the diagram decides one grouping. The grouping of variables in a MDD has been shown to affect its performance (Nagayama and Sasao, 2005)

The diagram is further optimized by automatically creating a merged node for the case when all edges in the group are unavailable together with all cases where the vertex is unavailable. For this reason a grouping of m edges with a vertex results in 2^m children for each node before merging, instead of 2^{m+1} . This number is further reduced by merging isomorphic nodes.

As with the OBDD-A, the OMDD-A tracks paths in the graph that have not yet been reached. This number of

paths increases rapidly for large networks, degrading the performance. The OBDD-H does not suffer from this issue, since it uses the boundary set system which is independent of the number of paths of the network and depends only of F_{max} and $B_{F_{max}}$.

```

1. Create root node  $N_0$ 
2.  $Q_C \leftarrow \{N_0\}$ ,  $Q_N \leftarrow \{ \}$ ,  $k \leftarrow 0$ , and  $REL \leftarrow 0$ .
3. while ( $Q_C \neq \{ \}$  or  $Q_N \neq \{ \}$ )
4.   if  $Q_C = \{ \}$  then
5.      $Q_C \leftarrow Q_N$ ,  $Q_N \leftarrow \{ \}$  and  $k \leftarrow k + 1$ .
6.    $F_k \leftarrow F_{k+1}$ , Compute  $F_{k+1}$ 
7.   for each  $N_i$  on  $Q_C$ .
8.     Create negative child
9.     for each combination of edges  $(v_k, v_x)$ :
10.      create positive child for these edges.
11.     for each child  $N_j$ 
12.       if  $N_j$  is non-terminal then
13.         for each  $N_q \in Q_N$  do
14.           if  $N_j$  is isomorphic to  $N_q$  then
15.             merge  $N_j$  into  $N_q$ 
16.             break.
17.           if no  $N_q$  was isomorphic to  $N_j$  then
18.             add  $N_j$  to  $Q_N$ .
19.           else if  $N_j$  is a success node then
20.              $REL \leftarrow REL + Pr(N_j)$ .
21.   delete  $N_i$ 
    
```

Figure 8: OMDD-H Algorithm

4.2 The OMDD-H

The hybrid ordered multi-variate decision diagram (OMDD-H) groups variables in the same way as the OMDD-A but uses partitions of F_k to track connectivity information. Like the OBDD-H it is restricted to undirected networks.

For this work we only consider OMDD-H for networks with both vertex and edge failure and don't consider the task of grouping vertices or edges together for other failure conditions (such as fallible vertices and perfect edges).

The OMDD-H for the sample network is shown in Figure 9 with the variable groupings shown on the right and probabilities rounded to five decimal places where required. It has 8 non-terminal nodes compared to the 4 nodes of the OBDD-Hv, the 10 nodes of the OBDD-He and the 24 nodes of the OBDD-Hve. Each link between nodes is labelled with a comma-separated list of the combination of edges that are available, with an X representing the negative child where either no edges are available or the vertex is unavailable. For example the label 3,23 means that this link is followed if edge e_3 is available and all other edges (in this case only e_2) are unavailable, and also if both e_2 and e_3 are available. The vertex in each grouping is available for each link not marked X.

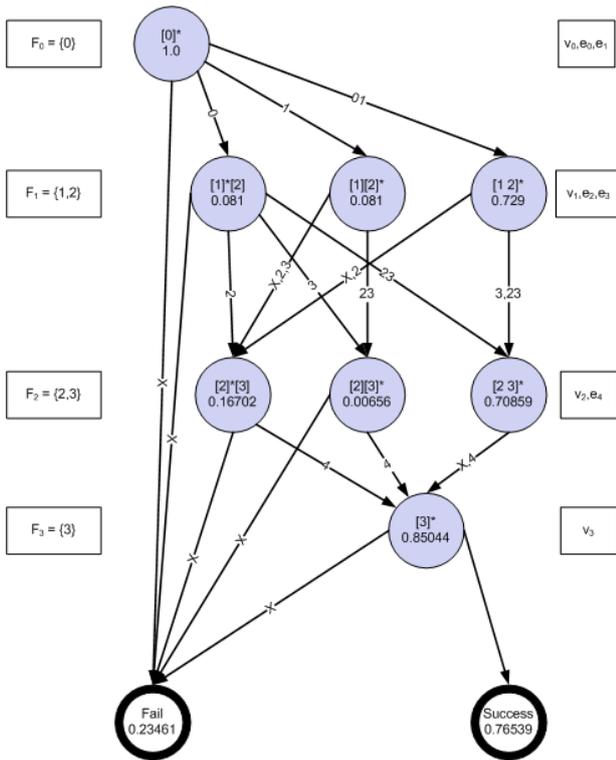


Figure 9: OMDD-H for Sample Network

It can be seen that the nodes of the OMDD-H are equivalent to nodes of the OBDD-Hve on levels that decide vertices. The intermediate nodes on levels of the OBDD-Hve that decide edges are subsumed by the processing of multiple edges on each level of the OMDD-H. Both methods give identical results, but the OMDD-H has fewer nodes.

4.3 The OMDD-H Algorithm

The OMDD-H algorithm given in Figure 8 is closely related to the OBDD-Hv algorithm (Herrmann and Soh, 2009). However instead of one variable being decided per level, an entire grouping of variables is decided.

First the negative child (representing the vertex or all edges in the grouping being unavailable) is created (line 8) by deleting every edge in the grouping. Then the algorithm loops through each combination of available edges and creates the corresponding positive node by contracting every available edge and deleting every unavailable edge (lines 9-10). Note that the combination in which all edges are unavailable is part of the negative child and hence is not considered for the positive loop.

Each non-terminal child created, whether positive or negative, is then compared to the nodes in Q_N . If an isomorphic node is found both are merged. If not the child node is added to Q_N^6 (lines 12-18).

Terminal nodes are never added to the queue. Failure nodes are ignored while success nodes have their probabilities added to REL. When the algorithm terminates, the variable REL contains the appropriate network reliability.

⁶ In implementations of OMDD-H it is best to keep Q_N sorted in order to reduce the number of comparisons made. This does not affect the worst-case complexity but does reduce the average processing time.

4.4 The Complexity of OMDD-H

Since the OMDD-H uses partitions of boundary sets, each level is subject to the same bounds as the OBDD-H. Like the OBDD-H, only two levels of nodes are kept in memory at any one time. Hence the time complexity of the OMDD-H is

$$O(|V| \times (F_{max})^3 \times B_{F_{max}})$$

and the space complexity is

$$O((F_{max})^3 \times B_{F_{max}}).$$

It can be seen that these complexities are identical to those of OBDD-Hv.

While this seems to coincide with the experimental results finding that the performance of the OMDD-A is comparable to that of the OBDD-A (Herrmann *et al.*, 2009) it should be noted that the nodes of the OMDD-A may be larger than those of the OBDD-A. By contrast, each OMDD-H node is identical to the corresponding OBDD-H node except that it can have multiple children. Since nodes are never explicitly linked the number of children has no effect on the memory requirements for a node. Hence the sizes of the OMDD-H and OBDD-H nodes are identical.

While the space complexities of OBDD-Hv and OMDD-H are closely related, the time complexities are somewhat misleading. Although the time complexities show that both diagrams process a comparable number of nodes, many OMDD-H nodes processed requires that multiple positive children are created. Many of these will be found to be isomorphic, but the process of creating them and checking for isomorphism must be carried out first. Hence the processing overhead of an OMDD-H node will be greater than that of an OBDD-H node, even if the number of nodes after isomorphism are comparable.

It should be noted that the time complexity for OMDD-H is slightly misleading since it does not take into account that slightly more processing is needed for each positive child compared to OBDD-He, OBDD-Hv and OBDD-Hve.

5 Conclusion

We have described a generalized hybrid ordered binary decision diagram (OBDD-H) method that can be used on networks with edge and/or vertex failures. OBDD-H is shown analytically more efficient than the BS method which is extremely efficient at computing REL for undirected networks with only edge failures. We also proposed the Hybrid Ordered Multi-variate Decision Diagram (OMDD-H) which combines the best features of both the augmented ordered multi-variate decision diagram (OMDD-A) and the boundary set method (BS). The resulting OMDD-H has been shown to have better time complexity than the OBDD-H for networks with fallible devices and links and to have comparable space efficiency. It is thus an extremely appropriate tool for analysing these types of networks.

All of the methods using partitioning of the boundary set (*i.e.*, BS, OBDD-H, OMDD-H) require that the networks be undirected. The less efficient augmented diagrams (*i.e.*, OBDD-A, OMDD-A) have the advantage in that directed networks can be analysed. It would be useful to extend the hybrid diagrams to allow analysis of

directed networks as well, if this can be done without sacrificing performance.

Finally, the OBDD-A and OMDD-A have been shown to be capable of computing metrics other than REL. Experiments will be undertaken to test whether the OBDD-H and OMDD-H can be extended for this purpose.

6 References

- Akyildiz, W. S., Su, W., Sankarasubramaniam, Y. & Cayirci, R. (2002). A Survey on Sensor Networks. *IEEE Communications Magazine*, **August**: 102-114.
- Carlier, J. & Lucet, C. (1996). A Decomposition Algorithm for Network Reliability Evaluation. *Discrete Applied Mathematics*, **65**: 141-156.
- Colbourn, C. J. 1987. *The Combinatorics of Network Reliability*, New York, Oxford University Press.
- Hardy, G., Lucet, C. & Limnios, N. (2005). Computing all-terminal reliability of stochastic networks with Binary Decision Diagrams. *In: 11th International Symposium on Applied Stochastic Models*, 2005.
- Hardy, G., Lucet, C. & Limnios, N. (2007). K-Terminal Network Reliability Measures With Binary Decision Diagrams. *IEEE Trans. Reliability*, **56**: 506 - 515.
- Herrmann, J. (2010). Improving Reliability Calculation with Augmented Binary Decision Diagrams. *In: AINA*, Apr. 2010 Perth, Australia. IEEE.
- Herrmann, J., Soh, S. & West, G. (2007). An OBDD Approach for Computing Expected Hop Count of Communication Networks. *In: PEECS 2007*, 2007 Perth, Australia. Curtin University of Technology.
- Herrmann, J. U. & Soh, S. (2009). A Space Efficient Algorithm for Network Reliability. *In: 15th Asia-Pacific Conf. Communications (APCC2009)*, Oct. 2009.
- Herrmann, J. U., Soh, S., West, G. & Rai, S. (2009). Using Multi-valued Decision Diagrams to Solve the Expected Hop Count Problem. *In: IEEE 23rd Int. Conf. Advanced Information Networking and Applications Workshops*, 2009 Bradford, UK. 419-424.
- Nagayama, S. & Sasao, T. (2005). On the optimization of heterogeneous MDDs. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, **24**: 1645-1659.
- Yeh, F.-M., Lu, S.-K. & Kuo, S.-Y. (2002). OBDD-Based Evaluation of k-Terminal Network Reliability. *IEEE Trans. Reliability*, **51**: 443-451.

Author Index

- Andreae, Peter, 63
Anslow, Craig, 63
Ashford, James, 127
- Bogdanovych, Anton, 107
Buyya, Rajkumar, 25
- Cameron, Linda, 53
Cassell, Keith, 63
Churcher, Neville, 117, 127
- Fischer, Susanne, 53
- Groves, Lindsay, 63
- Herrmann, Johannes U., 153
Hingston, Philip, 137
Hsieh, Chaur-Heh, 11
Hu, Shangfeng, 3
Huang, Pingsheng, 11
Huda, Shamsul, 43
Huynh, Dat, 145
- Ijaz, Kiran, 107
Irwin, Warwick, 117, 127
- Jago, Lana, 53
Javadi, Bahman, 25
- Kowalkiewicz, Marek, 3
Kuruppu, Shanika, 91
- Lam, Chiou, 137
Li, Xiaodong, 83
Liu, Chengfei, 3
- Müller, Stefan, 53
Ma, Wanli, 145
Marshall, Stuart, 63
Masek, Martin, 137
Morunga, Eva Rose, 53
Murshed, Manzur M., 17
- Oosterman, Joshua, 117
- Parikh, Umang, 53
Puglisi, Simon, 91
- Reid, Robert, 35
Reynolds, Mark, iii
- Salehi, Mohsen Amini, 25
Sharna, Shusmita Anwar, 17
Shen, Haifeng, 73
Simoff, Simeon, 107
Soh, Sieteng, 153
Straneieri, Andrew, 43
- Tang, Ming-Da, 11
Tirtha, Ranjeet, 137
Tran, Dat, 145
- Wüensche, Burkhard, 53
- Yan, Yongyao, 73
Yearwood, John, 43
- Zhai, Zhaolin, 83
Zhang, Ji, 99
Zhao, Xiaohui, 3
Zobel, Justin, 91

Recent Volumes in the CRPIT Series

ISSN 1445-1336

Listed below are some of the latest volumes published in the ACS Series *Conferences in Research and Practice in Information Technology*. The full text of most papers (in either PDF or Postscript format) is available at the series website <http://crpit.com>.

- Volume 91 - Computer Science 2009**
 Edited by Bernard Mans Macquarie University.
 January, 2009. 978-1-920682-72-9.
 Contains the proceedings of the Thirty-Second Australasian Computer Science Conference (ACSC2009), Wellington, New Zealand, January 2009.
- Volume 92 - Database Technologies 2009**
 Edited by Xuemin Lin, University of New South Wales and Athman Bouguettaya, CSIRO.
 January, 2009. 978-1-920682-73-6.
 Contains the proceedings of the Twentieth Australasian Database Conference (ADC2009), Wellington, New Zealand, January 2009.
- Volume 93 - User Interfaces 2009**
 Edited by Paul Calder Flinders University and Gerald Weber University of Auckland.
 January, 2009. 978-1-920682-74-3.
 Contains the proceedings of the Tenth Australasian User Interface Conference (AUI2009), Wellington, New Zealand, January 2009.
- Volume 94 - Theory of Computing 2009**
 Edited by Prabhu Manyam, University of Ballarat and Rod Downey, Victoria University of Wellington.
 January, 2009. 978-1-920682-75-0.
 Contains the proceedings of the Fifteenth Computing: The Australasian Theory Symposium (CATS2009), Wellington, New Zealand, January 2009.
- Volume 95 - Computing Education 2009**
 Edited by Margaret Hamilton, RMIT University and Tony Clear, Auckland University of Technology.
 January, 2009. 978-1-920682-76-7.
 Contains the proceedings of the Eleventh Australasian Computing Education Conference (ACE2009), Wellington, New Zealand, January 2009.
- Volume 96 - Conceptual Modelling 2009**
 Edited by Markus Kirchberg, Institute for Infocomm Research, A*STAR, Singapore and Sebastian Link, Victoria University of Wellington, New Zealand.
 January, 2009. 978-1-920682-77-4.
 Contains the proceedings of the Fifth Asia-Pacific Conference on Conceptual Modelling (APCCM2008), Wollongong, NSW, Australia, January 2008.
- Volume 97 - Health Data and Knowledge Management 2009**
 Edited by James R. Warren, University of Auckland.
 January, 2009. 978-1-920682-78-1.
 Contains the proceedings of the Third Australasian Workshop on Health Data and Knowledge Management (HDKM 2009), Wellington, New Zealand, January 2009.
- Volume 98 - Information Security 2009**
 Edited by Ljiljana Brankovic, University of Newcastle and Willy Susilo, University of Wollongong.
 January, 2009. 978-1-920682-79-8.
 Contains the proceedings of the Australasian Information Security Conference (AISC 2009), Wellington, New Zealand, January 2009.
- Volume 99 - Grid Computing and e-Research 2009**
 Edited by Paul Roe and Wayne Kelly, QUT.
 January, 2009. 978-1-920682-80-4.
 Contains the proceedings of the Australasian Workshop on Grid Computing and e-Research (AusGrid 2009), Wellington, New Zealand, January 2009.
- Volume 100 - Safety Critical Systems and Software 2007**
 Edited by Tony Cant, Defence Science and Technology Organisation, Australia.
 December, 2008. 978-1-920682-81-1.
 Contains the proceedings of the 13th Australian Conference on Safety Critical Systems and Software, Canberra, Australia, December, 2008.
- Volume 101 - Data Mining and Analytics 2009**
 Edited by Paul J. Kennedy, University of Technology, Sydney, Kok-Leong Ong, Deakin University and Peter Christen, The Australian National University.
 November, 2009. 978-1-920682-82-8.
 Contains the proceedings of the 8th Australasian Data Mining Conference (AusDM 2009), Melbourne, Victoria, Australia, November, 2009.
- Volume 102 - Computer Science 2010**
 Edited by Bernard Mans, Macquarie University, Australia and Mark Reynolds, University of Western Australia, Australia.
 January, 2010. 978-1-920682-83-5.
 Contains the proceedings of the Thirty-Third Australasian Computer Science Conference (ACSC 2010), Brisbane, Queensland, Australia, January 2010.
- Volume 103 - Computing Education 2010**
 Edited by Tony Clear, Auckland University of Technology, New Zealand and John Hamer, University of Auckland, New Zealand.
 January, 2010. 978-1-920682-84-2.
 Contains the proceedings of the Twelfth Australasian Computing Education Conference (ACE 2010), Brisbane, Queensland, Australia, January 2010.
- Volume 104 - Database Technologies 2010**
 Edited by Heng Tao Shen, University of Queensland, Australia and Athman Bouguettaya, CSIRO ICT Centre, Australia.
 January, 2010. 978-1-920682-85-9.
 Contains the proceedings of the Twenty-First Australasian Database Conference (ADC 2010), Brisbane, Queensland, Australia, January 2010.
- Volume 105 - Information Security 2010**
 Edited by Colin Boyd, Queensland University of Technology, Australia and Willy Susilo, University of Wollongong, Australia.
 January, 2010. 978-1-920682-86-6.
 Contains the proceedings of the Eight Australasian Information Security Conference (AISC 2010), Brisbane, Queensland, Australia, January 2010.
- Volume 106 - User Interfaces 2010**
 Edited by Christof Lutteroth, University of Auckland, New Zealand and Paul Calder Flinders University, Australia.
 January, 2010. 978-1-920682-87-3.
 Contains the proceedings of the Eleventh Australasian User Interface Conference (AUI2010), Brisbane, Queensland, Australia, January 2010.
- Volume 107 - Parallel and Distributed Computing 2010 2010**
 Edited by Jinjun Chen, Swinburne University of Technology, Australia and Rajiv Ranjan, University of New South Wales, Australia.
 January, 2010. 978-1-920682-88-0.
 Contains the proceedings of the Eighth Australasian Symposium on Parallel and Distributed Computing (AusPDC 2010), Brisbane, Queensland, Australia, January 2010.
- Volume 108 - Health Informatics and Knowledge Management 2010**
 Edited by Anthony Maeder, University of Western Sydney, Australia and David Hansen, CSIRO Australian e-Health Research Centre, Australia.
 January, 2010. 978-1-920682-89-7.
 Contains the proceedings of the Fourth Australasian Workshop on Health Informatics and Knowledge Management (HIKM 2010), Brisbane, Queensland, Australia, January 2010.

Volume 109 - Theory of Computing 2010

Edited by Taso Viglas, University of Sydney, Australia and Alex Potanin, Victoria University of Wellington, New Zealand. January, 2010. 978-1-920682-90-3.

Contains the proceedings of the Sixteenth Computing: The Australasian Theory Symposium (CATS 2010), Brisbane, Queensland, Australia, January 2010.

Volume 110 - Conceptual Modelling 2010

Edited by Sebastian Link, Victoria University of Wellington, New Zealand and Aditya Ghose, University of Wollongong, Australia. January, 2010. 978-1-920682-92-7.

Contains the proceedings of the Seventh Asia-Pacific Conference on Conceptual Modelling (APCCM2010), Brisbane, Queensland, Australia, January 2010.

Volume 112 - Advances in Ontologies 2009

Edited by Edited by Thomas Meyer, Meraka Institute, South Africa and Kerry Taylor, CSIRO ICT Centre, Australia. December, 2009. 978-1-920682-91-0.

Contains the proceedings of the Australasian Ontology Workshop 2009 (AOW 2009), Melbourne, Australia, December, 2009.

