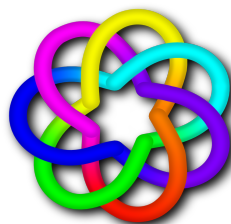


CONFERENCES IN RESEARCH AND PRACTICE IN
INFORMATION TECHNOLOGY

VOLUME 147

COMPUTER SCIENCE 2014

AUSTRALIAN COMPUTER SCIENCE COMMUNICATIONS, VOLUME 36, NUMBER 1



COMPUTER SCIENCE 2014

Proceedings of the
Thirty-Seventh Australasian Computer Science Conference
(ACSC 2014), Auckland, New Zealand,
20 – 23 January 2014

Bruce Thomas and Dave Parry, Eds.

Volume 147 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 2014. Proceedings of the Thirty-Seventh Australasian Computer Science Conference (ACSC 2014), Auckland, New Zealand, 20 – 23 January 2014

Conferences in Research and Practice in Information Technology, Volume 147.

Copyright ©2014, 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:

Bruce Thomas

School of Computer and Information Science
Division of Information Technology, Engineering and the Environment
University of South Australia
Adelaide, SA 5001
Australia
Email: bruce.thomas@unisa.edu.au

Dave Parry

School of Computing and Mathematical Sciences
AUT University
Auckland 1142
New Zealand
Email: dave.parry@aut.ac.nz

Series Editors:

Vladimir Estivill-Castro, Griffith University, Queensland
Simeon J. Simoff, University of Western Sydney, NSW
Email: crpit@scem.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 147.
ISSN 1445-1336.
ISBN 978-1-921770-30-2.

Document engineering, January 2014 by CRPIT
On-line proceedings, January 2014 by the University of Western Sydney
Electronic media production, January 2014 by the AUT University

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-Seventh Australasian Computer Science Conference (ACSC 2014), Auckland, New Zealand, 20 – 23 January 2014

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	xiii
ACSW and ACSC 2014 Sponsors	xv

Contributed Papers

An Adaptive Aggregate Maintenance Approach for Mixed Workloads in Columnar In-Memory Databases <i>Stephan Müller, Lars Butzmann, Stefan Klauck and Hasso Plattner</i>	3
Combining the Shortest Paths and the Bottleneck Paths Problems	13
<i>Tong-Wook Shinn and Tadao Takaoka</i>	
Lazy and Eager Approaches for the Set Cover Problem	19
<i>Ching Lih Lim, Alistair Moffat and Anthony Wirth</i>	
Towards a Vertex and Edge Label Aware Force Directed Layout Algorithm	29
<i>Roman Klapaukh, David J Pearce and Stuart Marshall</i>	
DAC: Database Application Context Analysis applied to Enterprise Applications	39
<i>Johannes Wust, Carsten Meyer and Hasso Plattner</i>	
Shape Predicates Allow Unbounded Verification of Linearizability Using Canonical Abstraction	49
<i>David Friggens and Lindsay Groves</i>	
Document DNA: Content Centric Provenance Data Tracking in Documents	57
<i>Michael Rinck, Annika Hinze and David Bainbridge</i>	
Exploring the applicability of Reservoir methods for Classifying Punctual Sports Activities Using On-body Sensors	67
<i>Doug P. Hunt, Dave Parry and Stefan Schliebs</i>	
A Comparative Study of RFID Technology Measuring Efficiency and Acceptance when Capturing Attendance	75
<i>Steven Tucker, Peter Darcy and Bela Stantic</i>	
A Trigger Counting Mechanism for Ring Topology	81
<i>Sushanta Karmakar and Subhrendu Chattopadhyay</i>	
Formal Approach for Generating Privacy Preserving User Requirements-Based Business Process Frag- ments	89
<i>Mohamed Anis Zemni, Amel Mammar and Nejib Ben Hadj-Alouane</i>	

Poisson Blended Exemplar-based Texture Completion	99
<i>Hoang M. Nguyen, Burkhard C. Wünsche, Patrice Delmas and Christof Lutteroth</i>	
A Replication and Reproduction of Code Clone Detection Studies	105
<i>Xiliang Chen, Alice Yuchen Wang and Ewan Tempero</i>	
Mining Indonesian Cyber Bullying Patterns in Social Networks	115
<i>Hendro Margono, Xun Yi and Gitesh K. Raikundalia</i>	
Current Educational Technology Use for Digital Information Acquisition by Young New Zealand Children	125
<i>Nicholas Vanderschantz, Annika Hinze and Sally Jo Cunningham</i>	
Understanding Saudi Arabian students engagement in E-learning 2.0 in Australian Higher Education	135
<i>Omar Mayan, Judy Sheard and Angela Carbone</i>	
Author Index	145

Preface

The Australasian Computer Science Conference (ACSC) series is an annual meeting, bringing together research sub-disciplines in Computer Science. The conference allows academics and other researchers to discourse research topics as well as progress in the field, and policies to stimulate its growth. This conference is unique in its ability to provide a platform for cross-disciplinary research. This volume comprises papers being presented at the Thirty-Seventh ACSC in Auckland, New Zealand.

ACSC 2014 is part of the Australasian Computer Science Week which runs from January 20th to 23rd, 2014. The ACSC 2014 call for papers solicited 37 submissions from Australia, New Zealand, Czech Republic, France, Germany, India, Indonesia, Malaysia, Poland, Slovenia, Taiwan, Thailand, Tunisia, and United Kingdom.

The topics addressed by the submitted papers illustrate the broadness of the discipline. These included algorithms, databases virtualisation, document retrieval, ubiquitous computing, wearable computing, networking, privacy, image processing, software engineering, social media, and education, to name just a few.

The programme committee consisted of 32 highly regarded academics from Australia, New Zealand, Italy, Japan, China, and Korea. Every paper was reviewed by at least three programme committee members, and, in some cases, external reviewers. Of the 37 papers submitted, 16 were selected for presentation at the conference.

The Programme Committee determined that the “Best Paper Award” should go to Justin Nguyen, Burkhard Wünsche, Patrice Delmas and Christof Lutteroth. For “Poisson Exemplar-based Texture Completion” Congratulations !

We thank all authors who submitted papers and all conference participants for helping to make the conference a success. We also thank the members of the programme committee and the external referees for their expertise in carefully reviewing the papers. We are grateful to Dr. Russell Pears from the Auckland University of Technology, New Zealand for his assistance in the production of the proceedings. I thank Professor John Grundy (President) for his support representing CORE (the Computing Research and Education Association of Australasia).

Last, but not least, we express gratitude to our hosts at the Auckland University of Technology and, in particular the ACSW general chairs Tony Clear and Russel Pears.

Bruce Thomas

University of South Australia

Dave Parry

Auckland University of Technology

ACSC 2014 Programme Chairs

January 2014, Auckland, New Zealand

Programme Committee

Chairs

Bruce Thomas, University of South Australia, Australia
Dave Parry, Auckland University of Technology, New Zealand

Members

Matt Adcock, CSIRO, Australia
Boris Bačić, Auckland University of Technology, New Zealand
Fred Brown, University of Adelaide, Australia
Kuda Dube, Massey University, New Zealand
Julien Epps, University of New South Wales, Australia
Ken Hawick, Massey University, New Zealand
Michael E. Houle, National Institute of Informatics, Japan
Zhiyi Huang, University of Otago, New Zealand
Weidong (Tony) Huang, CSIRO, Australia
Shuji Kijima, Kyushu University, Japan
Paddy Krishnan, Bond University, Australia
Jiuyong Li, University of South Australia, Australia
Jixue Liu, University of South Australia, Australia
William Liu
Michael Marner, University of South Australia, Australia
Wolfgang Mayer, University of South Australia, Australia
Chris McDonald, University of Western Australia, Australia
Muhammad Asif Naeem, Auckland University of Technology, New Zealand
Linda Pagli, University of Pisa, Italy
Maurice Pagnucco, University of New South Wales, Australia
Jun Park, Magic Vision Lab, University of South Australia, Australia
Yuping Shen, Institute of Logic and Cognition, Department of Philosophy, Sun Yat-sen University, China
James Skene, Auckland University of Technology, New Zealand
Tim Simon, University of South Australia, Australia
Markus Stumptner, University of South Australia, Australia
Ewan Tempero, University of Auckland, New Zealand
Burkhard C. Wüensche, University of Auckland, New Zealand
Hua Wang, University of Southern Queensland, Australia
Ruili Wang, Massey University, New Zealand
Xinfeng Ye, University of Auckland, New Zealand
Jianlong Zhou, University of South Australia, Australia

Additional Reviewers

Quan Yu	A H M Sarowar Sattar
Selasi Kwashie	Lili Sun
Duncan Stevenson	Hamidu Abdel-Fatao
Viveka Weiley	Guangrui Dang
Lili Sun	S. M. Masud Karim
Chris Gunn	

Organising Committee

Chairs

Tony Clear and Russel Pears

Venue

Tony Clear

Communications

Russel Pears, Hui Ling Tan, Melanie Curry-Irons and Ryan Butler

Finance

Alison Clear and Eva Ihaia

Sponsorship

Stephen Thorpe

Operations

Adam Winship and Eva Ihaia

Programme, proceedings and booklet

Alison Clear

Catering and registration

AUT Hospitality Services

Welcome from the Organising Committee

On behalf of the Organising Committee, it is our pleasure to welcome you to Auckland and to the 2014 Australasian Computer Science Week (ACSW 2014). Auckland is New Zealand's largest urban area with a population of nearly one and a half million people. As the centre of commerce and industry, Auckland is the most vibrant, bustling and multicultural city in New Zealand. With the largest Polynesian population in the world, this cultural influence is reflected in many different aspects of city life. ACSW 2014 will be hosted at the City Campus of Auckland University of Technology (AUT), which is situated just up from the Town Hall and the Auckland central business district. ACSW is the premier event for Computer Science researchers in Australasia. ACSW2014 consists of conferences covering a wide range of topics in Computer Science and related areas, including:

- Australasian Computer Science Conference (ACSC) (Chaired by Bruce Thomas and Dave Parry)
- Australasian Computing Education Conference (ACE) (Chaired by Jacqueline Whalley and Daryl D'Souza)
- Australasian Information Security Conference (AISC) (Chaired by Udaya Parampalli and Ian Welch)
- Australasian User Interface Conference (AUIC) (Chaired by Burkhard C. Wünsche and Stefan Marks)
- Australasian Symposium on Parallel and Distributed Computing (AusPDC) (Chaired by Bahman Javadi and Saurabh Kumar Garg)
- Australasian Workshop on Health Informatics and Knowledge Management (HIKM) (Chaired by James Warren)
- Asia-Pacific Conference on Conceptual Modelling (APCCM) (Chaired by Georg Grossmann and Motoshi Saeki)
- Australasian Web Conference (AWC) (Chaired by Andrew Trotman)

This year reflects an increased emphasis for ACSW on community building. Complementing these published technical volumes therefore, ACSW also hosts two doctoral consortia and a number of associated workshops, including those for the Heads and Professors of Computer Science, plus for the first time the 'Australasian Women in Computing Celebration'. Naturally in addition to the technical program, there are a range of events, which aim to provide the opportunity for interactions among our participants. A welcome reception will be held in the atrium of the award winning newly built Sir Paul Reeves Building, which has integrated the city campus as a hub for student activity and provides a wonderful showcase for this year's ACSW. The conference banquet will be held on campus in one of the reception rooms in this impressive complex.

Organising a multi-conference event such as ACSW is a challenging process even with many hands helping to distribute the workload, and actively cooperating to bring the events to fruition. This year has been no exception. We would like to share with you our gratitude towards all members of the organising committee for their combined efforts and dedication to the success of ACSW2014. We also thank all conference co-chairs and reviewers, for putting together the conference programs which are the heart of ACSW, and to the organisers of the symposia, workshops, poster sessions and accompanying conferences. Special thanks to Alex Potanin, as the steering committee chair who shared valuable experiences in organising ACSW and to John Grundy as chair of CoRE for his support for the innovations we have introduced this year. We'd also like to thank Hospitality Services from AUT, for their dedication and their efforts in conference registration, venue, catering and event organisation. This year we have secured generous support from several sponsors to help defray the costs of the event and we thank them for their welcome contributions. Last, but not least, we would like to thank all speakers, participants and attendees, and we look forward to several days of stimulating presentations, debates, friendly interactions and thoughtful discussions.

We hope your stay here will be both rewarding and memorable, and encourage you to take the time while in New Zealand to see some more of our beautiful country.

Tony Clear

Russel Pears

School of Computer & Mathematical Sciences

ACSW2014 General Co-Chairs

January, 2014

CORE - Computing Research & Education

CORE welcomes all delegates to ACSW2014 in Auckland. 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 component conferences of ACSW have changed over time with additions and subtractions. ACSC, ACE, AISC, AUIC, AusPDC, HIKM, ACDC, APCCM, CATS and AWC have now been joined by the Australasian women in computing celebration (AWIC), two doctoral consortia (ACDC and ACE-DC) and an Australasian Early Career Researchers Workshop (AECRW) which reflect the evolving dimensions of ACSW and build on the diversity of the Australasian computing community.

In 2014, we have again chosen to feature a small number of keynote speakers from across the discipline: Anthony Robins (ACE), John Mylopolous (APCCM), and Peter Gutmann (AISC). I thank them for their contributions to ACSW2014. The efforts of the conference chairs and their program committees have led to strong programs in all the conferences, thanks very much for all your efforts. Thanks are particularly due to Tony Clear, Russel Pears and their colleagues for organising what promises to be a vibrant event. Below I outline some of CORE's activities in 2012/13.

I welcome feedback on these including other activities you think CORE should be active in.

The major sponsor of Australian Computer Science Week:

- The venue for the annual Heads and Professors meeting
- An opportunity for Australian & NZ computing staff and postgrads to network and help develop their research and teaching
- Substantial discounts for attendees from member departments
- A doctoral consortium at which postgrads can seek external expertise for their research
- An Early Career Research forum to provide ECRs input into their development

Sponsor of several research, teaching and service awards:

- Chris Wallace award for Distinguished Research Contribution
- CORE Teaching Award
- Australasian Distinguished Doctoral Dissertation
- John Hughes Distinguished Service Award
- Various Best Student Paper awards at ACSW

Development, maintenance, and publication of the CORE conference and journal rankings. In 2013 this includes a new portal with a range of holistic venue information and a community update of the CORE 2009 conference rankings.

Input into a number of community resources and issues of interest:

- Development of an agreed national curriculum defining Computer Science, Software Engineering, and Information Technology
- A central point for discussion of community issues such as research standards
- Various submissions on behalf of Computer Science Departments and Academics to relevant government and industry bodies, including recently on Australian Workplace ICT Skills development, the Schools Technology Curriculum and the Mathematics decadal plan

Coordination with other sector groups:

- Work with the ACS on curriculum and accreditation
- Work with groups such as ACDICT and government on issues such as CS staff performance metrics and appraisal, and recruitment of ?students into computing
- A member of CRA (Computing Research Association) and Informatics Europe. These organisations are the North American and European equivalents of CORE.
- A member of Science & Technology Australia, which provides eligibility for Science Meets Parliament and opportunity for input into government policy, and involvement with Science Meets Policymakers

A new Executive Committee from 2013 has been looking at a range of activities that CORE can lead or contribute to, including more developmental activities for CORE members. This has also included a revamp of the mailing lists, creation of discussion forums, identification of key issues for commentary and lobbying, and working with other groups to attract high aptitude students into ICT courses and careers. Again, I welcome your active input into the direction of CORE in order to give our community improved visibility and impact.

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 2013, and look forward to the continuing shaping and development of CORE in 2014.

John Grundy

President, CORE
January, 2014

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.

2015. Volume 37. Host and Venue - University of Western Sydney, NSW.

2014. Volume 36. Host and Venue - AUT University, Auckland, New Zealand.

2013. Volume 35. Host and Venue - University of South Australia, Adelaide, SA.

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 AUC.

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 Computing Education Conference
ACSC	Australasian Computer Science Conference
ACSW	Australasian Computer Science Week
ADC	Australasian Database Conference
AISC	Australasian Information Security Conference
APCCM	Asia-Pacific Conference on Conceptual Modelling
AUIC	Australasian User Interface Conference
AusPDC	Australasian Symposium on Parallel and Distributed Computing (replaces AusGrid)
AWC	Australasian Web Conference
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 2014 Sponsors

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

Host Sponsor



Auckland University of Technology,
www.aut.ac.nz

Platinum Sponsor



DATACOM,
www.datacom.com.au

Gold Sponsor



**THE UNIVERSITY
OF AUCKLAND**
NEW ZEALAND

Te Whare Wānanga o Tamaki Makaurau

The University of Auckland,
www.auckland.ac.nz

Silver Sponsors



COLAB - AUT Design+Creative Technologies,
colab.aut.ac.nz



Institute of IT Professionals, New Zealand,
www.iitp.org.nz



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

Bronze Sponsor



Software Engineering Research Laboratory

SERL - AUT Software Engineering Research
Laboratory,
www.serl.aut.ac.nz



Australian Computer Society,
www.acs.org.au

Publication Sponsor



University of Western Sydney,
www.uws.edu.au

CONTRIBUTED PAPERS

An Adaptive Aggregate Maintenance Approach for Mixed Workloads in Columnar In-Memory Databases

Stephan Müller

Lars Butzmann

Stefan Klauck

Hasso Plattner

Hasso Plattner Institute
University of Potsdam, Germany
August-Bebel-Str. 88, 14482, Potsdam
Email: {firstname.lastname}@hpi.uni-potsdam.de

Abstract

The mixed database workloads generated by enterprise applications can be categorized into short-running transactional as well as long-running analytical queries with resource-intensive data aggregations. The introduction of materialized views can accelerate the execution of aggregate queries significantly. However, the overhead of materialized view maintenance has to be taken into account and varies mainly depending on the ratio of queries accessing the materialized view to queries altering the base data, which we define as insert ratio. On the basis of our constructed cost models for the identified materialized view maintenance strategies, we can determine the best performing strategy for the currently monitored workload. While a naive switching approach already improves the performance over staying with a single maintenance strategy, we show that an adaptive aggregate maintenance approach with inclusion of the workload history and switching costs can further improve the overall performance of a mixed workload. This behavior is demonstrated with benchmarks in a columnar in-memory database.

1 Introduction

Despite the accustomed association of online transactional processing (OLTP) and online analytical processing (OLAP) with separate applications, a modern enterprise application executes a mixed workload with both – transactional *and* analytical – queries [19, 20]. For example, within the available-to-promise (ATP) application, the OLTP-style queries represent product stock movements whereas the potentially very resource-intensive OLAP-style queries aggregate over the product movements to determine the earliest possible delivery date for requested goods by a customer [24]. To speed up the execution of OLAP-style queries with aggregates, a technique called *materialized views* has been proposed [23]. Throughout this paper, we use the term *materialized aggregate* for a materialized view whose creation query contains aggregations [22]. Accessing tuples of a materialized aggregate is always faster than aggregating on the fly. But the main drawback of introducing materialized views is the process of view maintenance which is necessary to guarantee consistency when the base data is changed [10]. Especially in mixed workload

environments, where transactional throughput must be guaranteed, a downtime due to materialized view maintenance is not acceptable.

In-memory databases (IMDB) such as SAP HANA [19], Hyrise [9] or Hyper [13] are able to handle mixed workloads comprised of transactional and analytical queries on a single system. In contrast to traditional databases, their storage is separated into a read-optimized main storage and a write-optimized delta storage. Since the main storage is highly-compressed and not optimized for inserts, all data changes of a table are propagated to the delta storage to provide high throughput. Periodically, the delta storage is combined with the main storage in a process called *merge operation* [14].

This new storage architecture has implications on existing materialized view maintenance approaches which we have evaluated in our recent work [15]. We showed that IMDBs with a main-delta architecture are well-suited for a novel view maintenance strategy called *merge update* [15, 18]. Because of the main-delta separation, the materialized aggregates do not have to be invalidated when new records are inserted to the delta storage because the materialized aggregates are only based on data from the main storage. To retrieve the consistent, final query result, the newly inserted records of the delta storage are aggregated on the fly and are combined – using a SQL UNION statement – with the materialized aggregate table. While the merge update strategy outperforms other view maintenance strategies for workloads with high insert ratios, it is not the ideal choice for all workloads. Based on this premise, we showed in [17], that switching between materialized view maintenance strategies can increase the overall performance compared to staying with a single strategy. In this paper, we contribute by proposing more advanced switching approaches that include a smoothing of the monitored workload patterns and take switching costs into account. Further, we evaluate how these approaches can be applied to multiple materialized aggregate tables.

Although we assume that our findings can be transferred to a wide range of enterprise applications, we use the available-to-promise (ATP) application as it provides a mixed workload varying between high select ratios (when checking for possible delivery dates) and high insert ratios (stock movements) [24]. In our implementation, ATP relies on a single, denormalized table called *Facts* that contains all stock movements in a warehouse including past and future orders (Table 5a). Every movement consists of a unique transaction identifier, the date, the id of the product being moved, and the amount. The amount is positive if goods are put in the warehouse and negative if goods are removed from the warehouse. The materialized

aggregate based on this table is called *Aggregates* (Table 5b). The aggregate groups the good movements by date and product and sums up the total amount per date and product. The ATP application does not consider physical data updates and uses an insert-only approach. Logical deletes and updates are handled through differential inserts. We further manually define the materialized views and do not address the view selection problem [11] in the scope of this paper. We focused on the *sum* aggregation function as this is the dominant aggregate function for the TPC-H benchmark¹.

The remainder of the paper is structured as follows: Section 2 gives a brief overview of related work. Section 3 explains view maintenance strategies in detail and describes workload patterns that motivate our research about switching maintenance strategies. Section 4 outlines algorithms for maintenance strategy switching before Section 5 benchmarks these and discusses the results. Section 6 provides an outlook on future work and concludes the paper with our main findings.

2 Related Work

Gupta gives a good overview of materialized views and related issues in [10]. Especially, the problem of materialized view maintenance has received significant attention in academia [6, 4]. Database vendors have also investigated this problem thoroughly [3, 25] but besides our earlier work [15], there is no work that evaluates materialized view maintenance strategies for mixed workloads. Instead, most of the existing research is focused on data warehousing environments [26, 1, 12, 16] where maintenance downtimes may be acceptable. Consequently, available DBMS only provide static view maintenance and support basic view maintenance strategies.

Chaudhuri et al. highlight in [7] the importance of automated physical database design including index and materialized view selection based on changing workloads. Agrawal et al. extend the definition of a workload by not only considering the ratios of query types within a workload, but also their sequence [2]. However, neither of them do address the problem of materialized view maintenance and how the optimal maintenance strategy can be chosen based on a changing workload.

3 Aggregate Maintenance Strategies

In [15], various aggregate maintenance strategies were presented and evaluated. It was shown that the insert ratio of a workload has the biggest influence on the execution performance. Figure 1 shows the aggregate maintenance and access times of different maintenance strategies for workloads with insert ratios between 0 and 1. For each single workload, the insert ratio was constant to allow comparisons between them. For each insert ratio, either *smart lazy incremental update* (SLIU) or *merge update* (MU) has the lowest workload execution time. SLIU performs best for read intensive workloads, since only seldom writes, which change the aggregate, require maintenance activities. For workloads with increasing writes (more than 40 percent inserts), MU outperforms the other strategies.

However, enterprise workloads are not characterized by constant insert ratios. Workloads change and therefore the best performing maintenance strategy

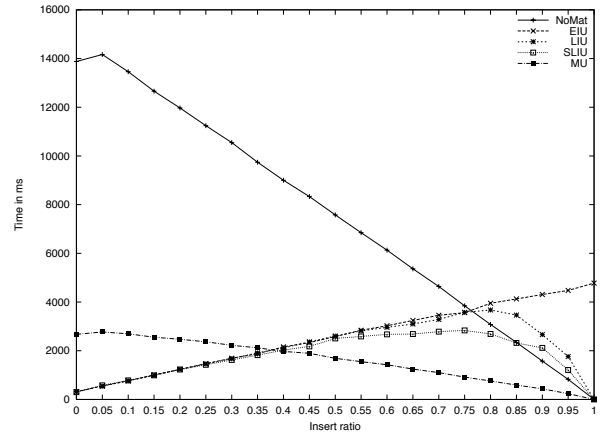


Figure 1: Aggregate maintenance and access time of different maintenance strategies for workloads with different insert ratios.

Table 1: Definition of symbols

Symbol	Definition
N_{total}	Total number of queries
N_{insert}	Number of insert queries
N_{select}	Number of select queries
N_{delta}	Number of records in delta storage
R_{select}	Select ratio
R_{insert}	Insert ratio
T_{select}	Time to select the aggregate
T_{delta}	Time to aggregate the delta storage
$T_{maintenance}$	Time to maintain the aggregate
T_{union}	Time to union two results
T_{dict}	Time to read from the dictionary structure

changes. The remainder of the section starts with a recap of SLIU and MU. The maintenance costs and switching costs are described to motivate the research of switching strategies. A definition of symbols used for the cost function is listed in Table 1. Additionally, patterns for changing workloads are listed.

3.1 Smart Lazy Incremental Update

Using the *smart lazy incremental update* (SLIU) strategy, the maintenance is done when processing selects querying the materialized aggregate. Thereby, the where-clause of the query is evaluated and only aggregates contained in the result set are maintained. Hence, after processing a select, the requested aggregates are up to date. In order to be able to maintain the aggregate during a select, one has to store the changes caused by inserts since the last maintenance point. This is done in a dictionary, called proxy structure, storing the difference between the materialized aggregate and the current correct aggregate for each combination of grouping values.

Table 2 shows the starting point of a SLIU maintenance scenario with a materialized view containing fresh aggregates based on the base table (c.f. Tables 2a, 2b). The corresponding proxy structure, shown in Table 2c, is empty.

Incoming inserts are not immediately included in the materialized aggregate (c.f. Table 3b). The resulting changes for the aggregate are temporarily stored in the proxy structure as shown in Table 3c.

When querying the aggregates table, the maintenance is triggered. Thereby, only requested aggregates are updated using the proxy structure. Af-

¹<http://www.tpc.org/tpch/>

(a) Snapshot of initial base table

Facts							
Main				Delta			
ID	Date	Prod	Amt	ID	Date	Prod	Amt
1	1/1/2013	1	100				
2	1/1/2013	1	-50				
3	1/1/2013	2	30				
4	1/1/2013	2	60				
5	1/2/2013	1	-10				

(b) Fresh materialized aggregate

Aggregates		
Date	Prod	SUM(Amt)
1/1/2013	1	50
1/2/2013	1	-10
1/1/2013	2	90

(c) Empty proxy structure

Proxy structure of Aggregates		
Key		Value
Date	Prod	SUM(Amt)

ter selecting the aggregates for product 1, the corresponding aggregates are up-to-date and the proxy structure contains no entries for those grouping values with product 1 (c.f. Tables 4a, 4b).

Equation 1 shows the costs for a single query using SLIU. The first summand describes the costs for read accesses on the materialized aggregate. T_{select} is the average time for a single read of an aggregate. This time is multiplied by the select ratio R_{select} to weight the costs, since they are not required for inserts. The costs to maintain the aggregate are calculated by the second summand. The costs of a single maintenance activity are $T_{dict} + T_{maintenance}$. The number of single maintenance activities increases with an increasing insert ratio R_{insert} , since each insert demands a maintenance activity when the corresponding aggregate is requested. However, with an increasing number of inserts, the maintenance process can be optimized. The calculation of the whole maintenance costs is therefore divided into two scenarios. With an insert ratio R_{insert} smaller than or equal to 0.5, the maintenance costs $R_{insert} * (T_{dict} + T_{maintenance})$ are linear. With an insert ratio greater than 0.5, the average maintenance costs decrease due to two facts. First, the possibility of combining multiple values in the proxy structure with the same grouping attributes. Second, a "bulk" maintenance where all relevant values from the proxy structure are processed together. This improvement is expressed by the optimization function in Equation 2.

$$costs_{SLIU} = R_{select} * T_{select} + optimization(R_{insert}) * R_{insert} * (T_{dict} + T_{maintenance}) \quad (1)$$

$$optimization(x) = \begin{cases} 1 & 0 \leq x \leq 0.5 \\ 2 - 2x & 0.5 < x \leq 1 \end{cases} \quad (2)$$

Table 3: SLIU maintenance: after three inserts

(a) Snapshot of base table after three inserts

Facts							
Main				Delta			
ID	Date	Prod	Amt	ID	Date	Prod	Amt
1	1/1/2013	1	100				
2	1/1/2013	1	-50				
3	1/1/2013	2	30				
4	1/1/2013	2	60				
5	1/2/2013	1	-10				
				6	1/2/2013	1	20
				7	1/1/2013	3	50
				8	1/1/2013	3	-10

(b) Materialized aggregate

Aggregates		
Date	Prod	SUM(Amt)
1/1/2013	1	50
1/2/2013	1	-10
1/1/2013	2	90

(c) Proxy structure

Proxy structure of Aggregates		
Key		Value
Date	Prod	SUM(Amt)
1/2/2013	1	-10
1/1/2013	3	40

Algorithm 1 Tear down for smart lazy incremental update strategy

```

1: procedure SLIU_TEAR_DOWN(mat_aggregate)
2:   for all rows row in the proxy_structure of mat_aggregate do
3:     (update the value of the mat_aggregate table at row.key by row.value)
4:   end for
5:   (delete proxy_structure)
6: end procedure

```

Setup A proxy structure has to be created to store the temporary changes caused by inserts.

Tear down All records from the proxy structure have to be included into the materialized aggregate. Algorithm 1 explains the required steps in detail.

3.2 Merge Update

The *merge update (MU)* strategy leverages the existence of a delta storage in a columnar IMDB. Using this strategy, the materialized aggregate always consists of the aggregated main storage. Values from the delta storage, which have been inserted after a merge operation, are not included in the materialized aggregate. Instead, when querying the aggregate, the data stored in the delta is aggregated on the fly and combined with the materialized aggregate to represent the fresh aggregate. With each merge operation [14], the values from the delta storage are aggregated and the materialized aggregate table is updated accordingly.

Table 5a shows a table consisting of a main and delta storage. The materialized aggregate (c.f. Table 5b) stores the values of the main storage. When querying the aggregate, the result (c.f. Table 5c) is calculated by combining the materialized aggregate with the on the fly aggregated delta.

Table 4: SLIU maintenance: after querying product 1

(a) Materialized aggregate

Aggregates		
Date	Prod	SUM(Amt)
1/1/2013	1	50
1/2/2013	1	-20
1/1/2013	2	90

(b) Proxy structure

Proxy structure of Aggregates		
Key		Value
Date	Prod	SUM(Amt)
1/1/2013	3	40

Algorithm 2 Tear down for merge update strategy

```

1: procedure MU_TEAR_DOWN(mat.aggregate)
2:   base_table  $\leftarrow$   $\langle$ get the base table of mat.aggregate $\rangle$ 
3:   delta  $\leftarrow$   $\langle$ get all rows in delta of base_table $\rangle$ 
4:   aggr_delta  $\leftarrow$   $\langle$ aggregate the rows in delta as per
      mat.aggregate create statement  $\rangle$ 
5:    $\langle$ combine mat.aggregate table with aggr_delta $\rangle$ 
6: end procedure

```

The merge update strategy only creates costs when requesting an aggregate. However, since it has to access the delta storage, these costs are higher compared to an aggregate access using SLIU and therefore have to be included. Equation 3 shows the costs T_{select} for accessing the aggregate, T_{delta} for aggregating on the delta and the costs to combine both results T_{union} .

$$costs_{MU} = R_{select} * (T_{select} + T_{delta} + T_{union}) \quad (3)$$

Setup After a strategy switch to MU, the materialized aggregate is up to date and therefore includes all records of the main and delta storage. Hence, the values from the delta storage have to be subtracted from the materialized aggregate, so that it only contains aggregated main storage records. Alternatively, a merge can be performed to combine the records of the delta storage and the main storage. In that case, the materialized aggregate stays the same.

Tear down The values from the delta storage have to be included into the materialized aggregate. This is done by aggregating the records of the delta storage and combine it with the materialized aggregate. Algorithm 2 explains the process in detail.

The introduced parameters, e.g. time for a select T_{select} and time to maintain the aggregate $T_{maintenance}$ depend on the underlying hardware. The calibrator introduced in [15] helps to determine these values.

3.3 Workloads

Workloads are characterized by queries differing in type and complexity. As our research focuses on aggregate maintenance, we study workloads containing queries that request or change aggregates. Our models distinguish two kinds of queries: single inserts changing the base table and selects querying single aggregate values. Resulting, the workload can be described by the terms *insert ratio* respectively *select ratio*. The insert ratio R_{insert} specifies the number

Table 5: MU: calculation of the fresh aggregate

(a) Snapshot of base table

Facts							
Main				Delta			
ID	Date	Prod	Amt	ID	Date	Prod	Amt
1	1/1/2013	1	100				
2	1/1/2013	1	-50				
3	1/1/2013	2	30				
4	1/1/2013	2	60				
5	1/2/2013	1	-10				
				6	1/2/2013	1	20
				7	1/1/2013	3	50
				8	1/1/2013	3	-10

(b) Materialized aggregate: based on main storage

Aggregates		
Date	Prod	SUM(Amt)
1/1/2013	1	50
1/2/2013	1	-10
1/1/2013	2	90

(c) On the fly calculated fresh aggregate

Result when querying Aggregates		
Date	Prod	SUM(Amt)
1/1/2013	1	50
1/2/2013	1	10
1/1/2013	2	90
1/1/2013	3	40

of insert queries in relation to the total number of queries (Equation 4). Consequently, the select ratio is $1 - R_{insert}$. These ratios change during a workload depending on the business application.

$$R_{insert} = \frac{N_{insert}}{N_{total}} \quad (4)$$

$$R_{select} = 1 - R_{insert} \quad (5)$$

There is no typical workload for enterprises as they have different business applications implying different database schema and queries. We use a randomized workload pattern, called random walk, as general pattern for enterprise workloads. Additionally, more regular patterns like periodic, linear and hard switching changes are employed. In the following, we characterize these patterns with its configuration parameters.

3.3.1 Random Walk

Enterprise workloads differ and cannot be described by a single workload pattern. To match many different scenarios, we use a configurable randomized workload. The insert ratio of the workload randomly increases or decreases after constant time frames. Configuration parameters influence the exact behavior, e.g. how fast the insert ratio changes or how high the probability of consecutive phase with insert ratio increases respectively decreases are. Additionally, upper and lower bounds of the ratio can be set. This way, we can setup highly unpredictable workloads to test our switching strategies.

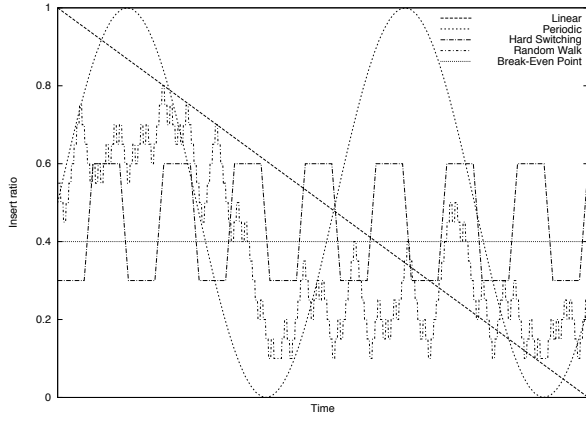


Figure 2: Different workload patterns we used for our evaluation.

3.3.2 Periodic Pattern

The periodic pattern is the first regular workload behavior. The insert ratio behaves like a sinus curve: it periodically increases to the configured maximum and decreases to its minimum. The periodic pattern can be further configured with the length of period and the amplitude. This pattern can match a typical customer-based workload with peaks during the day and lows during the night.

3.3.3 Linear Pattern

The insert ratio increases respectively decreases linearly. It is beside the hard switching pattern a simple changing behavior of the insert ratio. This pattern does not reflect any specific business application. It is rather used to show the benefit of switching the maintenance strategy in a simple scenario. However, linear insert ratio changes are often part of more complicated enterprise workloads.

3.3.4 Hard Switching Pattern

The insert ratio of the hard switching pattern jumps between certain values. The time of a constant ratio value can be configured. This pattern can reflect extreme changes of workloads. Enterprises with hourly batch jobs and businesses with several query peaks per day have workloads matching this pattern.

3.3.5 Examples

Figure 2 shows an example for each workload patterns. The linear pattern has a constantly decreasing insert ratio. The periodic pattern consists of two sinus periods with an amplitude between 0 and 1. The hard switching pattern jumps between 0.3 and 0.6. The random walk starts at 0.5, goes up to 0.8 and stays between 0.1 and 0.5 in the second half. Compared to the other three examples, the insert ratio of the random walk is not smooth. Additionally, the break-even point of the merge update and smart lazy incremental update strategy is included (cf. Section 3).

3.4 Multiple Materialized Aggregates

In [15], we have concentrated our work on a single materialized aggregate. However, enterprise applications typically work with multiple materialized aggregates depending on the current scenario.

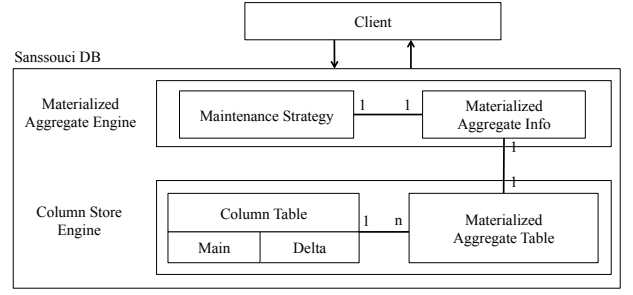


Figure 3: Internal architecture of Sanssouci DB including the novel materialized aggregate engine.

Figure 3 shows the architecture of SanssouciDB [21]. Each column table can have multiple materialized aggregates. Each aggregate has meta information and its own maintenance strategy. This independence is important for our materialized aggregate engine and the switching strategies. As a result, the engine is able to choose the optimal maintenance strategy for each aggregate individually. The required information about the number of accesses on the aggregate and the number of modifications is therefore stored in the meta information.

Figure 4a shows the SQL statements of three materialized aggregates: *Aggregates.Total*, *Aggregates.Q3* and *Aggregates.2013*. They all have the same base table *Facts*. The first aggregate has two grouping attributes and no where clause, meaning that all new inserts affect the aggregate. The second aggregate has a where clause including a date (year 2013) and an amount filter. In the ATP scenario, amounts greater than 0 refer to incoming products. Therefore, approximately half of the inserts affect the aggregate. The third aggregate has, compared to the previous ones, the most restricted where clause. It has filters on the date (third quarter of 2013) and the amount (all outgoing products). Since the third quarter is already over, only a few inserts queries affect the aggregate.

These three aggregates only represent a subset of the aggregates that real enterprises have.

Figure 4b shows one example for an insert query and one example for accessing the aggregate.

4 Maintenance Strategy Switching

As shown in Figure 1, changing insert ratios imply a change of the maintenance performance. Hence, it is preferable to change the maintenance strategy according to workload changes. [17] presented a simple switching algorithm henceforth called naive switching. It was shown that naive switching outperforms static maintenance strategies for workloads with varying insert ratios. However, naive switching with certain configurations can be unfavorable for workloads with insert ratios oscillating around the break-even point. That is why, new switching algorithms are introduced to avoid unnecessary maintenance strategy switches which can decrease the overall performance.

4.1 Naive Switching

Naive switching follows the idea to switch to the best performing strategy as early as possible. Therefore, it monitors the current workload for a configurable number of queries called window. The number of inserts and selects is counted. When the end of a window is reached, the number is divided by the window size to calculate the ratios. Given the insert ratio,

```

CREATE MATERIALIZED VIEW Aggregates_Total AS
SELECT date, product, SUM(amount)
FROM Facts
GROUP BY date, product;

CREATE MATERIALIZED VIEW Aggregates_2013 AS
SELECT date, SUM(amount)
FROM Facts
WHERE date >= 1/1/2013 AND date <= 12/31/2013
AND amount > 0
GROUP BY date;

CREATE MATERIALIZED VIEW Aggregates_Q3 AS
SELECT product, SUM(amount)
FROM Facts
WHERE date >= 6/1/2013 AND date <= 9/30/2013
AND amount < 0
GROUP BY product;

```

(a) Three materialized aggregate creation queries.

```

INSERT INTO Facts
(id, date, product, amount)
VALUES (1, 7/30/2013, 1, 100);

SELECT date, product, amount
FROM Aggregates
WHERE product = 1;

```

(b) An insert into the base table and a select on the aggregate.

Figure 4: Example SQL queries of the ATP scenario.

the optimal maintenance strategy can be obtained by evaluating the cost functions of the single strategies. The naive strategy switches to the best performing strategy after each end of a window. The window size thereby controls two things: On the one hand, how long a maintenance strategy stays active until the next switch is possible, namely at least during the next window. On the other hand, how precise the calculated insert ratio is, because the greater the window the more precise the insert ratio.

In the following, the naive switching algorithm is explained in detail using the example workload in Figure 5a. Assume that the workload starts with SLIU as configured maintenance strategy. In the first window, an insert ratio of 0.3 is measured, meaning SLIU was the optimal maintenance strategy for the first window. That is why, SLIU is used for the second window, too. However, the insert ratio of the second window is 0.5 so that MU would have been the better choice. Hence, naive switching changes the maintenance strategy for the third window. For the windows three to six, MU stays the optimal maintenance strategy. Resulting, naive switching keeps MU until the seventh window. At the end of window seven, an insert ratio of 0.3 is measured and the used maintenance strategy is changed back to SLIU. Summarizing, the maintenance strategy is changed twice: after the second and seventh window. The workload was not executed with the best performing strategy during window two and seven.

The naive switching algorithm evaluates only the last window and does not consider the costs for switching to the optimal strategy. Hence, it is not the best switching strategy for specific benchmark scenarios, especially when the optimal aggregate maintenance strategy changes for each window. Figure 5b shows such a workload. Naive switching reacts on each workload change. However, since it takes one window to adjust the maintenance strategy, a non-

optimal strategy is used for each window.

4.2 History-Aware Switching

History-aware switching is an extension to the naive switching strategy. It includes not only the insert ratio of the last window to calculate the optimal maintenance strategy, but also the insert ratios of lapsed windows. By including multiple windows, history-aware switching calculates a smoothed insert ratio. In this way, unnecessary switches in the case of strongly varying workloads can be prevented. History-aware switching uses Brown's simple exponential smoothing [5].

$$\begin{aligned}
 s_1 &= x_0 \\
 s_t &= \alpha x_t + (1 - \alpha)s_{t-1} \quad 0 < \alpha < 1
 \end{aligned} \tag{6}$$

Given a starting value x_0 and a smoothing factor α , a smoothed value s_t is calculated based on the previous input x_{t-1} and the previous smoothed value s_{t-1} (Equation 6). The smoothing factor α determines the level of smoothing. The bigger α , the lower the level of smoothing. α can be chosen based on the level of information that is known in advance, e.g. historic data from previous days or weeks.

4.3 Cost-Aware Switching

Switching between two maintenance strategies creates tear down and setup costs as explained in Section 3. Neither naive nor history-aware switching consider these costs. Cost-aware switching takes the costs to switch to another strategy into account to prevent switches, which waste more time for switching than they gain for optimized maintenance performance. Algorithm 3 describes the procedure that determines if a switch is favorable or not. Therefore, the switching costs for each maintenance strategy have to be known. Additionally, a data structure to keep track of savings is required. As long as the current strategy is the fastest strategy, nothing happens. As soon as another strategy is faster than the current strategy, the cost difference between the current strategy and the other strategy is added to the savings data structure. In case the savings are smaller than the switching costs, nothing changes. In the other case when the savings are greater than the switching costs, the system switches to the new strategy and the data structure is reset.

5 Evaluation

To evaluate our presented concepts, we implemented the materialized aggregate engine in SanssouciDB [21] but we believe that they can be applied to other columnar IMDBs with a main-delta architecture such as SAP HANA [8]. Figure 3 illustrates the architecture of our implementation. The column store engine represents the persistence layer and contains the column table with its main and delta storage. The novel materialized aggregate engine is on top and is responsible for creating and maintaining materialized aggregates. All logic for the maintenance strategies as well as the switching strategies is included there.

For our benchmarks, we used a data set of an ATP scenario that is based on customer data which we parametrized to generate different workload characteristics and patterns. The base table size for all benchmarks is 1M records. We have chosen this size for faster data imports and because the base table

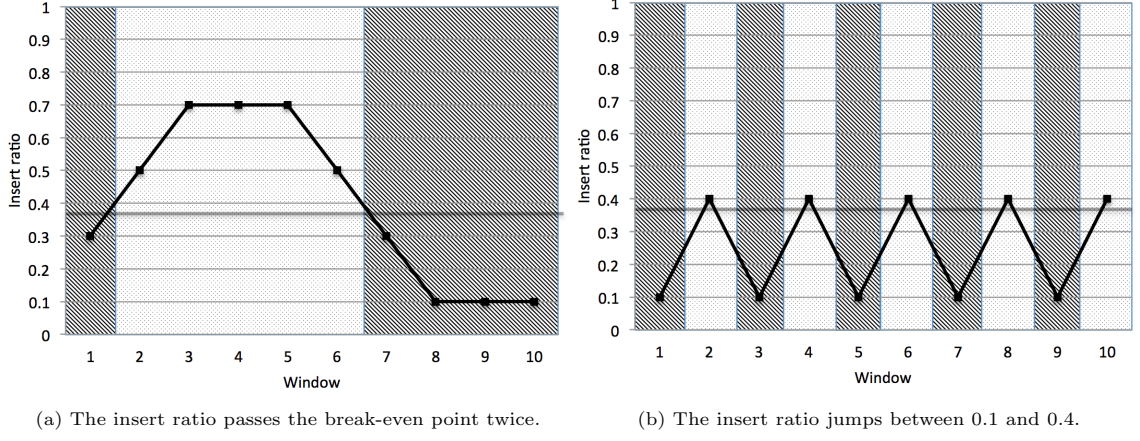


Figure 5: Workloads with changing insert ratios.

Algorithm 3 Saving cost calculation

```

1: switching_costs  $\leftarrow$   $\langle$ define switching costs for strategies $\rangle$ 
2: savings  $\leftarrow$   $\langle$ initialize savings for all strategies $\rangle$ 
3:
4: procedure CALCULATE_SAVINGS(costs_of_last_interval)
5:   current_strategy  $\leftarrow$   $\langle$ get current strategy $\rangle$ 
6:   fastest_strategy  $\leftarrow$   $\langle$ retrieve fastest strategy from
     costs_of_last_interval $\rangle$ 
7:   savings_per_strategy  $\leftarrow$   $\langle$ get current saving $\rangle$ 
8:   if current_strategy is fastest_strategy then
9:      $\langle$ reset savings to 0 $\rangle$ 
10:  else
11:     $\langle$ update savings_per_strategy with the
      cost delta of current_strategy
      and fastest_strategy $\rangle$ 
12:    savings  $\leftarrow$   $\langle$ savings for fastest_strategy
      from savings_per_strategy $\rangle$ 
13:    if savings for fastest_strategy are greater than
      switching_costs then
14:       $\langle$ switch to fastest_strategy $\rangle$ 
15:       $\langle$ reset savings to 0 $\rangle$ 
16:    end if
17:  end if
18: end procedure
    
```

size has no influence on the performance since we use incremental view maintenance strategies (as shown in [15]). The materialized aggregate contains about 4,000 records (i.e. date - product combinations). The workloads consist of two query types: selects querying aggregates filtered by product, and inserts with about 1,000 different date - product combinations. Three queries to create a materialized view, one to insert a value into the base table and one to select the aggregate are shown in Figure 4. Each workload contains 20k queries divided into 200 phases of constant insert ratios. Between consecutive phases, the insert ratio can stay constant or increase respectively decrease by 5 percent. For the history-aware switching strategy, α is chosen to be 0.5. This is equivalent to a history of three intervals.

All benchmarks have been conducted on a server featuring 8 CPUs (Intel Xeon E5450) with 3GHz and 12MB cache each. The entire machine was comprised of 64GB of main memory. Every benchmark in this section is run at least three times and the displayed results are the median of all runs.

The different switching strategies are compared with MU and SLIU, which use the same maintenance strategy all the time.

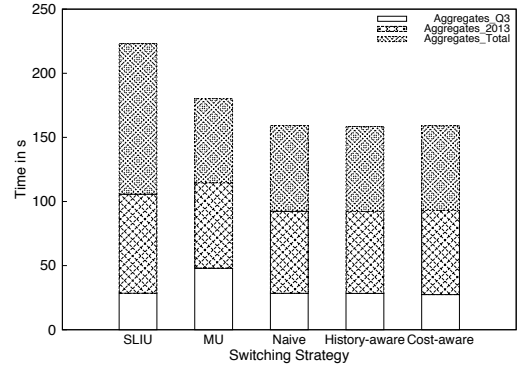


Figure 6: Performance of the switching strategies for multiple views.

5.1 Multiple Views

In Section 3.4, we have explained the need for multiple materialized aggregates in one system and on one table. The benchmark in Figure 6 evaluates the performance of the switching strategies for three different aggregates. The SQL for the aggregates *Aggregates.Total*, *Aggregates.2013* and *Aggregates.Q3* is shown in Figure 4a.

The results in Figure 6 show that individual switching strategies for aggregates perform better than static maintenance strategies. Since each aggregate has its own insert ratio caused by different materialized view definitions, the materialized view engine chooses the optimal strategy for each of them.

5.2 Basic Workload Patterns

The following benchmarks are based on our evaluation in [17], where we investigated the performance of a naive switching strategy for individual aggregates. Figure 7 shows the result of benchmarking the linear, periodic and hard switching workload patterns which were introduced in Section 3.3.

Two things can be observed. First, switching is always faster. All three patterns cover most of the insert ratio interval $[0, 1]$ and therefore cross the break-even point (see Figure 2). Consequently, switching is faster compared to the non switching approach. Second, all three switching strategies have nearly the same performance. This is a result of the characteristics of the workload patterns. All patterns are

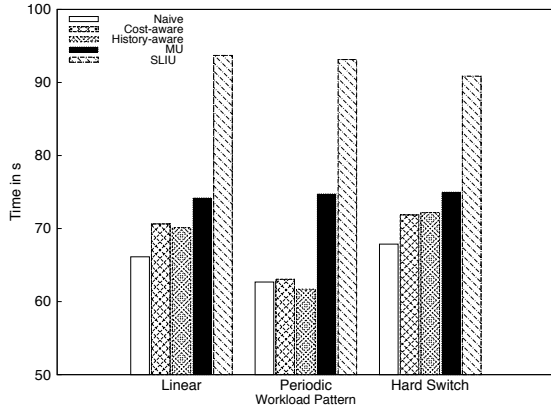


Figure 7: Performance of the switching strategies for a linear, periodic and hard switching pattern.

relatively simple and have a smooth behavior.

This benchmark only shows one example for each pattern. The characteristics of the three patterns can be varied, e.g. the amplitude of the periodic pattern can be smaller or the difference between the two values for the hard switching pattern can be larger. The more a workload stays on both sides of the break-even point, the greater is the advantage for the switching strategies.

5.3 Random Workloads

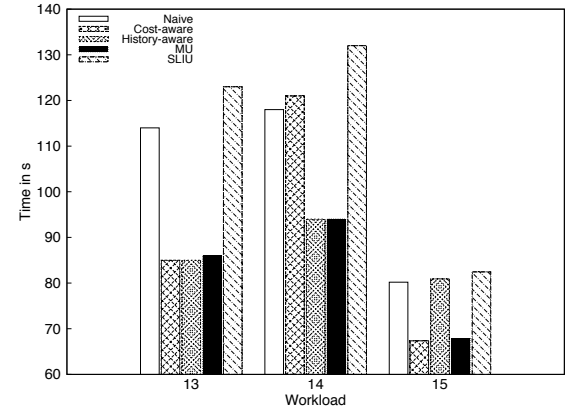
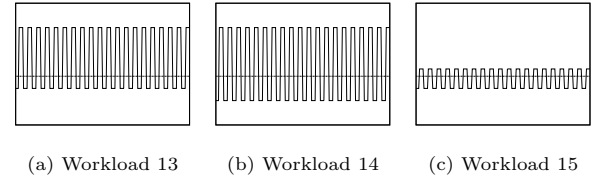
To measure the performance of the switching strategies for unpredictable workloads, we used random walks (see Section 3.3.1). Thereby, we varied the interval of possible insert ratios in the way it should influence the advantage of switching strategies:

1. $[0, 1]$ covers the largest possible interval. Switching in this setup should bring the most.
2. $[0.2, 0.6]$ covers the area close to the break-even point. The benefit of switching is expected to be lower.
3. $[0.3, 0.8]$ covers the interval beneficial for MU. The lower boundary crosses the break-even point slightly.
4. $[0, 0.5]$ covers the interval beneficial for SLIU. The upper boundary crosses the break-even point slightly.

Figure 9 includes benchmarks of the four intervals with three workloads each. Figure 9a shows the performance for workloads with the largest possible insert ratio interval ranging from 0 to 1. Switching is 27 percent faster than the fastest non switching strategy. Among the switching strategies, naive and cost-aware perform best. History-aware is slightly slower because of a deferred switching point.

The workloads, whose benchmark results are presented in Figure 9b, have an insert ratio interval of $[0.2, 0.6]$ (i.e. close to the break-even point). As a result, the performance advantage of switching strategies is smaller. The average improvement is approximately 18 percent. The cost-aware and naive switching strategy perform nearly the same.

In Figure 9c, the benchmark results for select-intensive workloads are presented. SLIU outperforms MU. However, its performance is beaten by the switching strategies. During the short period with insert ratios higher than 40 percent, switching strategies



(d) Insert ratio interval $[0.3, 0.8]$, $[0.2, 0.8]$ and $[0.3, 0.46]$

Figure 10: Three switching cost intensive workloads which cross the break-even point after each window.

change the maintenance strategy to the advantageous MU. The improvement of switching is 10 percent.

Workloads with insert ratios ranging from 0.3 to 0.8 are good for MU (Figure 9c). Again, switching has a slightly better execution time than MU (5 percent), since the workloads contain phases (with insert ratios smaller than 40 percent) where SLIU is the better maintenance strategy.

5.4 Worst Case Analysis

Even though naive switching is often the best approach, we have analyzed the behaviour of the switching strategies in extreme cases. As assumed, we measured that naive switching is not optimal for all workloads.

The workloads 13 to 15 in Figure 10 have an alternating pattern and jump between different insert ratios (0.3/0.8, 0.2/0.8 and 0.3/0.46). In Figure 10d, naive switching is significantly slower than the other two approaches. This is a result of unfavorable switching points due to the short interval for each insert ratio. The cost-aware switching decides not to switch based on the calculated savings (workload 13 and 15). However, in workload 14, the saved costs per interval are bigger than the switching costs and therefore the strategy is switched as in the naive approach. Only the history-aware strategy detects the fluctuating pattern and stays with one maintenance strategy, because the smoothed insert ratio only crosses the break-even point in the beginning and then evens out.

6 Conclusion

This paper introduces advanced algorithms to identify and switch to the optimal aggregate maintenance strategy. It has been shown that the performance of maintenance strategies depends on workload characteristics such as the insert ratio. According to workload changes, it is desirable to switch to the best performing aggregate maintenance strategy. Naive

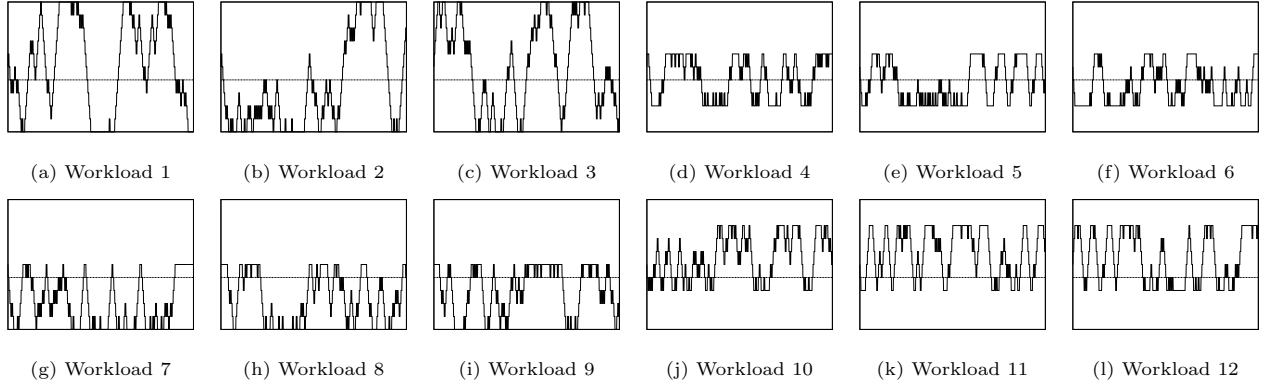


Figure 8: A visualization of the insert ratios for the workloads benchmarked in Figure 9.

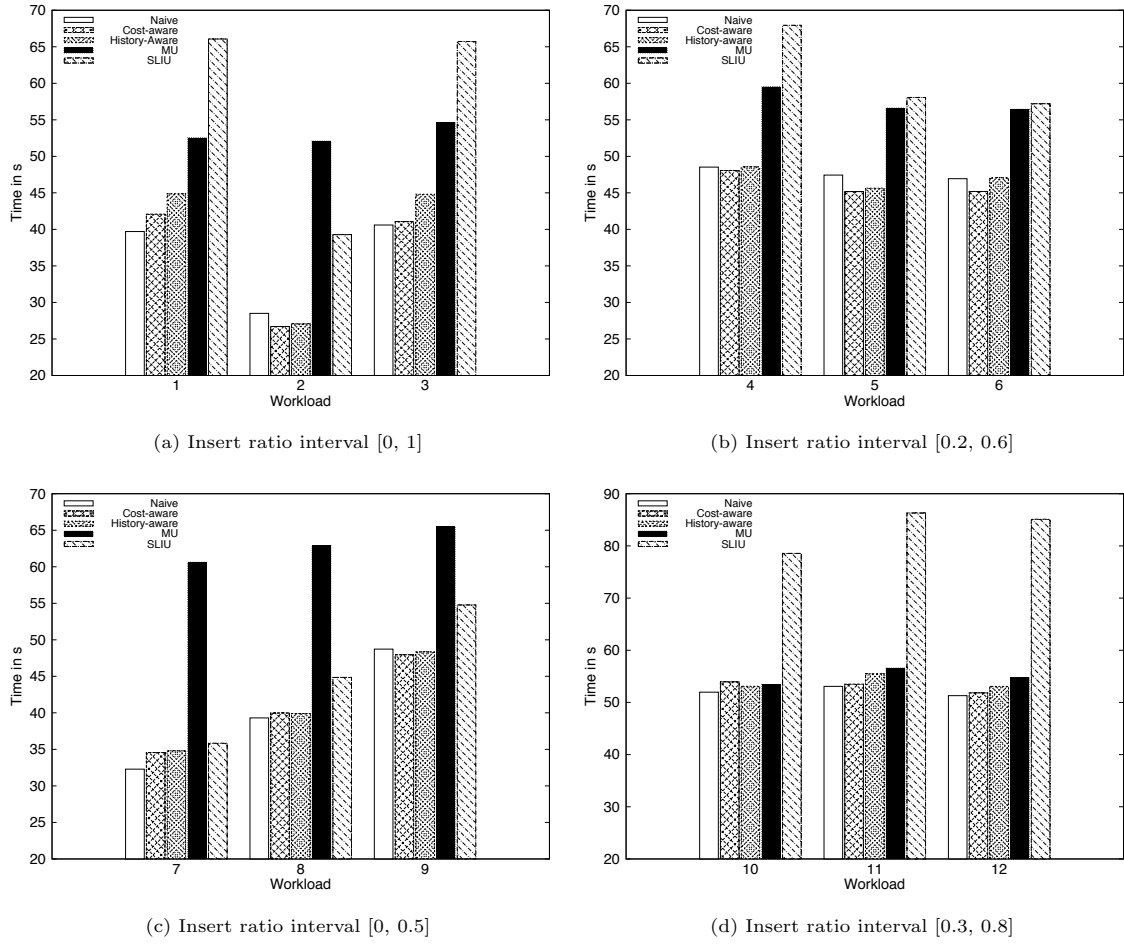


Figure 9: Benchmarks with different insert ratio intervals. Each workload consists of 20k queries.

switching uses a simple algorithm to select the maintenance strategy by following the goal to switch to the best performing strategy as early as possible. However, switching between strategies causes costs and the best performing strategy can often change in a short period of time. The two introduced switching algorithms tackle this issue and reduce the number of unnecessary maintenance strategy switches. To achieve that, they include the history of insert ratios and the costs of strategy switches.

We implemented the introduced switching strategies as part of a materialized aggregate engine in SanssouciDB. The materialized aggregate engine monitors the current workload, evaluates the cost functions and is able to switch to the optimal maintenance strategy. We benchmarked the various aggregate maintenance switching algorithms for workloads with different insert ratio courses. The results reveal that switching between maintenance strategies is beneficial for all identified workloads as it decreases the overall execution time. We evaluated three different switching strategies that reduce the execution time up to 27 percent. Among the different switching strategies, the naive switching strategy performs well. However, for certain workload patterns, the more sophisticated cost-aware and history-aware switching strategies are more beneficial.

As a direction of future work, we plan to employ a machine learning approach that predicts future workload changes and adjusts the materialized view maintenance strategy proactively.

References

- [1] D. Agrawal, A. El Abbadi, A. Singh, and T. Yurek. Efficient view maintenance at data warehouses. In *SIGMOD*, 1997.
- [2] S. Agrawal, E. Chu, and V. Narasayya. Automatic physical design tuning: Workload as a Sequence. In *SIGMOD*, pages 683–694, New York, New York, USA, 2006. ACM Press.
- [3] R. G. Bello, K. Dias, A. Downing, J. J. F. Jr., J. L. Finnerty, W. D. Norcott, H. Sun, A. Witkowski, and M. Ziauddin. Materialized views in oracle. In *VLDB*, pages 659–664, 1998.
- [4] J. A. Blakeley, P.-A. Larson, and F. W. Tompa. Efficiently updating materialized views. In *SIGMOD*, pages 61–71, 1986.
- [5] R. Brown. *Smoothing, Forecasting and Prediction of Discrete Time Series*. Dover Phoenix editions. Dover Publications, 2004.
- [6] O. P. Buneman and E. K. Clemons. Efficiently monitoring relational databases. *ACM Transactions on Database Systems*, 1979.
- [7] S. Chaudhuri and V. Narasayya. Self-tuning database systems: a decade of progress. In *VLDB*, 2007.
- [8] F. Färber, S. K. Cha, J. Primsch, C. Bornhövd, S. Sigg, and W. Lehner. SAP HANA database: data management for modern business applications. *SIGMOD*, 2011.
- [9] M. Grund, J. Krüger, H. Plattner, A. Zeier, P. Cudre-Mauroux, and S. Madden. Hyrise: a main memory hybrid storage engine. *VLDB*, pages 105–116, 2010.
- [10] A. Gupta and I. S. Mumick. Maintenance of materialized views: Problems, techniques, and applications. *IEEE Data Eng. Bull.* 1995.
- [11] H. Gupta. Selection of views to materialize in a data warehouse. *ICDT*, 1997.
- [12] H. Jain and A. Gosain. A comprehensive study of view maintenance approaches in data warehousing evolution. *SIGSOFT Softw. Eng. Notes* 2012.
- [13] A. Kemper, T. Neumann, F. F. Informatik, T. U. München, and D-Garching. Hyper: A hybrid oltp&olap main memory database system based on virtual memory snapshots. In *ICDE*, 2011.
- [14] J. Krueger, C. Kim, M. Grund, N. Satish, D. Schwalb, J. Chhugani, H. Plattner, P. Dubey, and A. Zeier. Fast Updates on Read-Optimized Databases Using Multi-Core CPUs. In *VLDB*, 2012.
- [15] S. Müller, L. Butzmann, K. Höwelmeyer, S. Klauck, and H. Plattner. Efficient View Maintenance for Enterprise Applications in Columnar In-Memory Databases. *EDOC*, 2013.
- [16] I. S. Mumick, D. Quass, and B. S. Mumick. Maintenance of data cubes and summary tables in a warehouse. In *SIGMOD*, 1997.
- [17] S. Müller, L. Butzmann, S. Klauck, and H. Plattner. Workload-aware aggregate maintenance in columnar in-memory databases. In *BPOE, in conjunction with IEEE International Conference on Big Data*, 2013.
- [18] S. Müller and H. Plattner. Aggregates caching in columnar in-memory databases. In *IMDM, in conjunction with VLDB*, 2013.
- [19] H. Plattner. A common database approach for oltp and olap using an in-memory column database. In *SIGMOD*, pages 1–2, 2009.
- [20] H. Plattner. Sanssoucidb: An in-memory database for processing enterprise workloads. In *BTW*, 2011.
- [21] H. Plattner and A. Zeier. *In-memory data management: an inflection point for enterprise applications*. Springer-Verlag Berlin Heidelberg, 2011.
- [22] J. M. Smith and D. C. P. Smith. Database abstractions: Aggregation. *Commun. ACM* 1977.
- [23] D. Srivastava, S. Dar, H. Jagadish, and A. Levy. Answering queries with aggregation using views. In *VLDB*, 1996.
- [24] C. Tinnefeld, S. Müller, H. Kaltegärtner, S. Hillig, L. Butzmann, D. Eickhoff, S. Klauck, D. Taschik, B. Wagner, O. Xylander, A. Zeier, H. Plattner, and C. Tosun. Available-to-promise on an in-memory column store. In *BTW*, pages 667–686, 2011.
- [25] J. Zhou, P.-A. Larson, and H. G. Elmongui. Lazy maintenance of materialized views. In *VLDB*, pages 231–242, 2007.
- [26] Y. Zhuge, H. García-Molina, J. Hammer, and J. Widom. View maintenance in a warehousing environment. In *SIGMOD*, pages 316–327, 1995.

Combining the Shortest Paths and the Bottleneck Paths Problems

Tong-Wook Shinn¹

Tadao Takaoka²

Department of Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand

¹ Email: tad@cosc.canterbury.ac.nz

² Email: tong-wook.shinn@pg.canterbury.ac.nz

Abstract

We combine the well known Shortest Paths (SP) problem and the Bottleneck Paths (BP) problem to introduce a new problem called the Shortest Paths for All Flows (SP-AF) problem that has relevance in real life applications. We first solve the Single Source Shortest Paths for All Flows (SSSP-AF) problem on directed graphs with unit edge costs in $O(mn)$ worst case time bound. We then present two algorithms to solve SSSP-AF on directed graphs with integer edge costs bounded by c in $O(m^2 + nc)$ and $O(m^2 + mn \log(\frac{c}{m}))$ time bounds. Finally we extend our algorithms for the SSSP-AF problem to solve the All Pairs Shortest Paths for All Flows (APSP-AF) problem in $O(m^2n + nc)$ and $O(m^2n + mn^2 \log(\frac{c}{mn}))$ time bounds. All algorithms presented in this paper are practical for implementation.

Keywords: Shortest Paths, SP, Bottleneck Paths, BP, Single Source Shortest Paths, SSSP, All Pairs Shortest Paths, APSP

1 Introduction

The problem of finding the shortest paths between pairs of vertices on a graph is one of the most extensively studied problems in algorithm research. This problem is formally known as the Shortest Paths (SP) problem and is often categorized into the Single Source Shortest Paths (SSSP) problem and the All Pairs Shortest Paths (APSP) problem. As the names suggest, the SSSP problem is to compute the shortest paths from one single source vertex to all other vertices on the graph, and the APSP problem is to compute the shortest paths between all possible pairs on the graph. The most well known algorithm for solving the SSSP problem is the algorithm by Dijkstra (3) that runs in $O(n^2)$ time, where n is the number of vertices in a graph. This algorithm can be enhanced with a priority queue, and if the Fibonacci heap (6) is used to implement the priority queue then the time complexity becomes $O(m + n \log n)$ where

m is the number of edges in the graph. If edge costs are integers bounded by c , then the SSSP problem can be solved in $O(m + n \log \log c)$ time (9) using a complex priority queue for integers. For solving the APSP problem, the $O(n^3)$ algorithm by Floyd (4) is the most well known.

If edges have capacities, the bottleneck of a path is the minimum capacity out of all edge capacities on the path. In other words, the bottleneck of a path from vertex u to vertex v is the maximum amount of flow that can be pushed from u to v down the path. Finding the paths that give the maximum bottlenecks between pairs of vertices is also a well studied problem and is formally known as the Bottleneck Paths (BP) problem. The Single Source Bottleneck Paths (SSBP) problem can be solved with a simple modification to the algorithm by Dijkstra (3). For an undirected graph the All Pairs Bottleneck Paths (APBP) problem can be solved in $O(n^2)$, which is optimal (7).

The SP and BP problems are concerned with finding the minimum or the maximum possible values. However, the shortest path may not give the biggest bottleneck, and the path that gives the maximum bottleneck may not be the shortest path. If the flow demand from a vertex to another vertex is known, then it is clearly beneficial to find the shortest path that can fully accommodate that flow demand. Thus we combine the SP and BP problems to compute the shortest paths for all possible flow amounts. We call this problem the Shortest Paths for All Flows (SP-AF) problem. As is common in graph paths problems, we divide the SP-AF problem into the Single Source Shortest Paths for All Flows (SSSP-AF) problem, and the All Pairs Shortest Paths for All Flows (APSP-AF) problem.

There are many obvious real life applications for this new problem, such as routing in computer networks, transportation, logistics, and even planning for emergency evacuations. The city of Christchurch has recently been hit by a series of strong earthquakes, most notably in September 2010 and in February 2011. If we consider hundreds of people evacuating from a building in such emergencies, if the amount of people (flow) can be predetermined, we can find the shortest route for all people in various locations around the building to their respective evacuation points such that the flow of people can be fully accommodated. If the flow amounts (of people) are not considered in calculating the evacuation routes, congestion may occur at various points in the building that could lead to serious accidents.

In this paper we present algorithms for solving the SSSP-AF and APSP-AF problems on directed graphs with non-negative integer edge costs and real edge capacities that are faster than the straightforward methods. We first give an algorithm to solve SSSP-AF

This research was supported by the EU/NZ Joint Project, Optimization and its Applications in Learning and Industry (OptALI).

Copyright ©2014, Australian Computer Society, Inc. This paper appeared at the Thirty-Seventh Australasian Computer Science Conference (ACSC2014), Auckland, New Zealand, January 2014. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 147, Bruce H. Thomas and David Parry, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.

on graphs with unit edge costs in $O(mn)$ time. We then present two algorithms for solving SSSP-AF on graphs with non-negative integer edge costs of at most c in $O(m^2 + nc)$ and $O(m^2 + mn \log(\frac{c}{n}))$ time bounds. Finally we show that the main concepts behind the SSSP-AF algorithms can be extended to solve APSP-AF in $O(m^2n + nc)$ and $O(m^2n + mn^2 \log(\frac{c}{mn}))$ time bounds.

2 Preliminaries

Let $G = \{V, E\}$ be a directed graph with non-negative integer edge costs bounded by c and non-negative real capacities. Let $n = |V|$ and $m = |E|$. Vertices are given by integers such that $\{1, 2, 3, \dots, n\} \in V$. Let (i, j) be the edge from vertex i to vertex j . Let $\text{cost}(i, j)$ denote the edge cost (or distance) and let $\text{cap}(i, j)$ denote the edge capacity. Let $\text{OUT}(v)$ be the set of vertices that are directly reachable from v , and $\text{IN}(v)$ be the set of vertices that can directly reach v . There can be up to m distinct capacities if all edge capacities are unique. We refer to the distinct capacities as maximal flows. Then the SP-AF problem is to solve the SP problem for all maximal flows.

For all our algorithms only comparison operations are performed on the maximal flow values. Therefore the real values of maximal flows can be mapped to integer values without any loss of generality by first sorting the maximal flows in increasing order then assigning incremental integer values starting from 1. This allows us to use maximal flow values as indexes of arrays in our algorithms.

We use the computational model that allows comparison-addition operations and random access with $O(\log n)$ bits to be performed in $O(1)$ time.

3 Single Source Shortest Paths for All Flows

We first consider solving the SP-AF problem from a source vertex s to all other vertices in G . Initially we compute just the distances rather than actual paths then later show that the paths information can be computed in the same time bound. That is, we first solve the Single Source Shortest Distances for All Flows (SSSD-AF) problem, then show that the algorithm can be extended to solve the SSSP-AF problem with no increase in the worst case time complexity.

The SSSD-AF problem can be defined as the problem of computing the set of all (d, f) pairs for each destination vertex, where d is the shortest distance from s and f is the maximal flow value. Let $S[v]$ be the set of (d, f) pairs for the destination vertex v . Suppose (d, f) and (d', f') both exist in $S[v]$ such that $d < d'$. Then we keep (d', f') iff $f < f'$, that is, a longer path is only *useful* if it can accommodate a greater flow. If $d = d'$, we keep the pair that gives us the greater flow.

Example Solving SSSP-AF on the example graph in Figure 1 with $s = 1$ would result in $S[6] = \{(4, 2), (5, 3), (6, 5), (8, 6)\}$.

The straightforward method for solving the SSSP-AF problem is to iterate through each maximal flow f_i and solving the SSSP problem for the sub-graph that only have edges with capacities f_i or greater. On graphs with unit edge costs SSSP can be solved in $O(m)$ time with a simple breadth-first-search (BFS), resulting in $O(m^2)$ time bound for solving the SSSP-AF problem. On graphs with integer edge costs

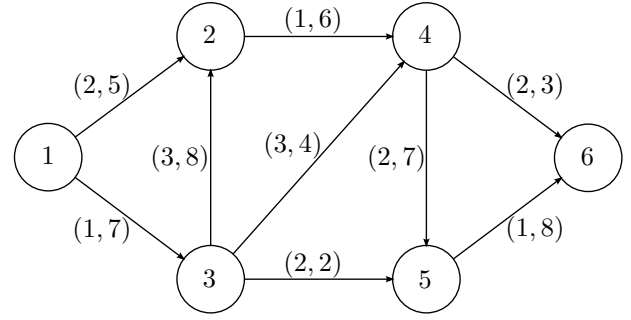


Figure 1: An example of a directed graph with $n = 6$, $m = 9$ and $c = 3$. The first number in the parenthesis is the edge cost and the second number is the edge capacity.

bounded by c , we can use the algorithm by Thorup (9) to solve SSSP-AF in $O(m^2 + mn \log \log c)$ time.

Note that SSSP-AF cannot be solved with a simple decremental algorithm, where edges are removed one by one in decreasing order of capacity then the connectivity of all affected vertices are checked. This method fails because edges with larger capacities may later be required to provide shorter paths for smaller maximal flows.

3.1 Unit edge costs

Algorithm 1 solves SSSP-AF in $O(mn)$ time by utilizing the fact that even though there are $O(m)$ maximal flows, there can only be $O(n)$ paths with unique path costs for each destination vertex. As noted earlier, if multiple paths from s to v exists with equal path costs, we only need to keep the path that can accommodate the biggest maximal flow out of those paths. Thus for graphs with unit edge costs, even with m maximal flows, the size of $S[v]$ is $O(n)$ for each v .

Let $B[v]$ be the bottleneck of a path from s to vertex v . Let $D[v]$ be a possible distance from s to v . Let $Q[i]$ be a set of vertices that may be added to Spanning Tree (SPT) at distance i , such that $1 \leq i \leq n - 1$, i.e. one set of vertices exists for each possible distance from s .

The algorithm starts off with just s in the SPT and makes incremental changes to the SPT as we iterate through the maximal flow values in increasing order. The SPT is a persistent data structure and we do not build it up from scratch in each iteration. In summary, as we iterate through each maximal flow we cut nodes from the SPT that cannot accommodate the maximal flow and add the nodes back to the SPT at the shortest possible distance from s (the root).

Lemma 3.1 *Algorithm 1 correctly solves SSSD-AF on directed graphs with unit edge costs.*

Proof By iterating from $Q[1]$ to $Q[n - 1]$ for each maximal flow, we ensure that all vertices are added to the SPT at the minimum possible distance from s for the maximal flow value in the current iteration. When the minimum possible distance is found for a vertex v to be added to the SPT (line 14), all potential parent nodes, $\text{IN}(v)$, are inspected to ensure that v is added to the SPT such that the bottleneck from s to v is maximized for the given distance (line 16). Since $D[v]$ is monotonically increasing (lines 8 and 22), v cannot be added to the SPT multiple times at the same distance. Thus any time a vertex v is added to the SPT, $(D[v], B[v])$ can be appended to $S[v]$ as

Algorithm 1 Solve SSSD-AF on graphs with unit edge costs in $O(mn)$ time

```

1: for all  $v \in V$  do
2:    $B[v] \leftarrow 0, D[v] \leftarrow 0$ 
3:  $B[s] \leftarrow \infty, SPT \leftarrow s$ 
4: for all maximal flows  $f$  in increasing order do
5:   for all  $v \in V$  such that  $B[v] < f$  do
6:     if  $v$  is in SPT then
7:       Cut  $v$  from SPT
8:        $D[v] \leftarrow D[v] + 1$ 
9:       Add  $v$  to  $Q[D[v]]$ 
10:  for  $i \leftarrow 1$  to  $n - 1$  do
11:    while  $Q[i]$  is not empty do
12:      Remove  $v$  from  $Q[i]$ 
13:      for all  $IN(v)$  as  $u$  do
14:        if  $D[u] = D[v] - 1$  then
15:           $b \leftarrow \min(\text{cap}(u, v), B[u])$ 
16:          if  $b > B[v]$  then
17:             $B[v] \leftarrow b$ 
18:            Add  $v$  to SPT,  $u$  as parent
19:        if  $v$  is in SPT then
20:          Append  $(D[v], B[v])$  to  $S[v]$ 
21:        else
22:           $D[v] \leftarrow D[v] + 1$ 
23:          Add  $v$  to  $Q[D[v]]$ 
    
```

a (d, f) pair. It follows that once we iterate through all maximal flows we have retrieved all relevant (d, f) pairs. ■

Lemma 3.2 *Algorithm 1 runs in $O(mn)$ worst case time.*

Proof We perform lifetime analysis to determine the upper bound of Algorithm 1. Each vertex v can be cut from the SPT and be re-added to the SPT $O(n)$ times, once per each possible distance from s . Cutting/adding v from/to the SPT takes $O(1)$ time, achieved by setting the parent of v to either $NULL$ or u , respectively. Therefore the total time complexity of all operations involving the SPT is $O(n^2)$. Also there are a total of $O(n^2)$ (d, f) pairs. Before each vertex v is added to the SPT all incoming edges (u, v) are inspected. This results in $O(m)$ edges being inspected in total for the entire duration of the algorithm for each possible distance from s . Since there are $O(n)$ possible distances from s , the total time taken for edge inspection is $O(mn)$. Thus the total worst case time complexity becomes $O(n^2 + mn) = O(mn)$. ■

Theorem 3.3 *There exists an algorithm to solve SSSP-AF on directed graphs with unit edge costs in $O(mn)$ time.*

Proof There are $O(n)$ destination vertices, $O(m)$ maximal flows, and the length of each path is $O(n)$. Therefore storing all explicit paths as a solution to the SSSP-AF problem takes $O(mn^2)$, which is too expensive. As is common in graph paths algorithms, we work around this problem by storing just the predecessor vertex for storing the path information. The predecessor vertex for a path from s to v is the vertex that comes immediately before v on the path.

We extend Algorithm 1 to store the parent vertex, u , alongside the (d, f) pair in line 20 i.e. we can extend the (d, f) pair to the (d, f, u) triplet. Then u is the predecessor vertex for the shortest path from s to v that can accommodate flow up to f . By using d and u , any explicit path can be retrieved by recursively following the predecessor vertices in time linear to

the path length. Clearly the additional storage of the predecessor vertex does not increase the worst case time complexity of the algorithm. ■

3.2 Cascading Bucket System

Before we move onto solving the SSSP-AF problem on graphs with integer edge costs, we review the k -level cascading bucket system (CBS) (1, 2). A detailed review of this data structure has also been provided by Takaoka (8).

In the k -level CBS the key value d is given by:

$$d = x_{k-1}p^{k-1} + x_{k-2}p^{k-2} + \dots + x_1p^1 + x_0$$

where p is the number of buckets (or length) of each level.

Let i be the largest index such that x_i is non-zero. Then an element with key of d is inserted into the x_i^{th} bucket at level i . The values of x_i for all $0 \leq i < k$ are calculated only once when an element is inserted, and each insertion takes $O(k)$ time.

The *decrease-key* operation can be performed by removing the element from the CBS in $O(1)$ time, updating the key value, then re-inserting in the same level in $O(1)$ time, or re-inserting at a lower level in $O(l)$ time where l is the difference between the initial level and the new level.

The *delete-min* operation is more involved than the *insert* or the *decrease-key* operations. We maintain an active pointer at each level, a_i for all $0 \leq i < k$, such that a_i is the minimum index of the non-empty bucket at level i . $a_i = p$ means level i is empty. To perform the *delete-min* operation, if level 0 is not empty, we simply pick up the minimum non-empty bucket pointed to by a_0 . If level 0 is empty, then we find the lowest non-empty level, j , and re-distribute the elements in the a_j^{th} bucket into level $j - 1$, then re-distribute the elements in the a_{j-1}^{th} bucket into level $j - 2$, and so on, until level 0 is non-empty. This process of repeated re-distribution from a higher level down to lower levels is referred to as *cascading*, hence the name for the data structure. Each *delete-min* takes $O(k + p)$ time if $j < k - 1$, and $O(k + M/p^{k-1})$ if $j = k - 1$, where M is the maximum key value that the CBS supports.

Example Figure 2 shows an example of a 3-level CBS that can support key values up to 1399. $p = 10$ was chosen to make the example easier to understand, since it becomes straightforward to determine the correct bucket for base 10 numbers. If the element with key equal to 19 is removed from the CBS, a_0 becomes 10, and the next *delete-min* operation will trigger the *cascading* operation, resulting in elements in a_1 to be re-distributed into level 0.

3.3 Integer edge costs

In Section 3.1 we gave an algorithm to solve SSSP-AF on directed graphs with unit edge costs in $O(mn)$ time, that is faster than the straightforward method of $O(m^2)$. In this section we present two algorithms to solve SSSP-AF on directed graphs with non-negative edge costs in $O(m^2 + nc)$ and $O(m^2 + mn \log(\frac{c}{m}))$ time bounds. Both time bounds are faster than the $O(m^2 + mn \log \log c)$ time bound of the straightforward method for a wide range of values for c , m and n , and have the added benefit of not relying on a complex data structure that is difficult to implement in real life situations. We note that Algorithm 1 can also be used to solve SSSP-AF in $O(mnc)$ time since the

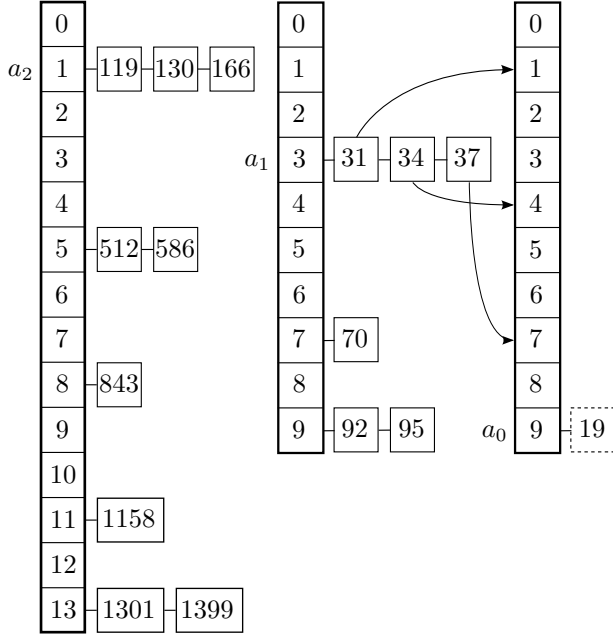


Figure 2: An example of a cascading bucket system with $k = 3$ and $p = 10$.

maximum distance from s for any vertex for any maximal flow value is $O(nc)$. $O(mnc)$ is a comparatively efficient time bound for dense graphs with small c .

The two time bounds of $O(m^2 + nc)$ and $O(m^2 + mn \log(\frac{c}{m}))$ actually both come from the same algorithm, Algorithm 2, but using different data structures to implement the priority queue. Algorithm 2 is a natural extension to the well known algorithm by Dijkstra (3). For this algorithm we define the triplet (v, d, f) , where v is the destination vertex and d and f are equivalent to the (d, f) pair as defined in Section 3. We let $D^f[v]$ be the current shortest distance from s to v for the maximal flow value of f . We let Q be the priority queue for the (v, d, f) triplets with d as the key where the operations performed on Q are *insert*, *decrease-key* and *delete-min*. In summary, all $O(mn)$ (v, d, f) triplets are added to the priority queue, Q , then as they are removed one by one, the (d, f) pair is appended to $S[v]$ if the (d, f) pair is *useful* i.e. f is greater than any flow in existing pairs in $S[v]$.

Definition (d, f) in $S[v]$ is correct if d is the shortest distance of a path that can push flows up to f from s to v .

Lemma 3.4 Algorithm 2 correctly solves SSSD-AF on directed graphs with integer edge costs.

Proof We provide a formal proof by induction. Let S be the set of (v, d, f) such that $S[v]$ contains the pair (d, f) , for all v , for all (d, f) pairs. Then in the beginning of each iteration:

1. The set of (d, f) pairs in each $S[v]$ are all correct i.e. all (v, d, f) triplets in S are correct.
2. For any (v, d, f) in Q , d is the distance of the shortest path from s to v that can push f only through the path that lies in S except for the end point v .

Basis. Before the while-loop begins (line 9) all are correct, and we suppose the theorem is correct at the beginning of some iteration. Then:

Algorithm 2 Solve SSSD-AF on graphs with integer edge costs

```

1: for all  $v \neq s \in V$  do
2:   for all maximal flows  $f$  do
3:      $D^f[v] \leftarrow \infty$ 
4:    $Q \leftarrow$  empty
5:   Add  $(s, 0, \infty)$  to  $Q$ 
6:   for all  $v \neq s \in V$  do
7:     for all maximal flow  $f$  do
8:       Add  $(v, \infty, f)$  to  $Q$ 
9:   while  $Q$  is not empty do
10:    Delete  $(v, d, f)$  with minimum  $d$  from  $Q$ 
11:    for all  $w \neq s \in OUT(v)$  do
12:       $f' \leftarrow \min(f, cap(v, w))$ 
13:       $d' \leftarrow d + cost(v, w)$ 
14:      if  $d' < D^{f'}[w]$  then
15:        Let  $x = (w, D^{f'}[w], f')$  in  $Q$ 
16:         $D^{f'}[w] \leftarrow d'$ 
17:        Put  $x$  in the correct position in  $Q$ 
18:    if  $S[v]$  is empty then
19:      Append  $(d, f)$  to  $S[v]$ 
20:    else
21:      Let  $(d_0, f_0) \leftarrow$  last pair in  $S[v]$ 
22:      if  $f_0 < f$  then
23:        if  $d_0 = d$  then
24:          Delete  $(d_0, f_0)$  from  $S[v]$ 
25:          Append  $(d, f)$  to  $S[v]$ 

```

1. Let (d_0, f_0) be the last pair in $S[v]$. Suppose (d, f) is appended to $S[v]$ at the end of the loop. Note that d values are sorted in increasing order in $S[v]$. Since f is appended only when $f > f_0$, there can be no shorter path in $S[v]$ that can push f . Thus (d, f) is appended as a correct pair.
2. We remove (v, d, f) from Q in line 10. The new distance d' and flow f' from v to w are computed for all possible w . If $d' < D^{f'}[w]$, (v, d, f) now lies in the path from s to w , and (w, d', f') is added to Q . Since (v, d, f) is added to $S[v]$ at the end of the loop, d' is the distance of the shortest path from s to w that can push f only through the path that lies in S except for the end point w . ■

Lemma 3.5 Algorithm 2 can run in $O(m^2 + nc)$ worst case time.

Proof We use a one dimensional bucket system to implement Q . Insert and decrease-key operations can be performed in $O(1)$, resulting in $O(mn)$ time bound for both operations for the whole algorithm. The delete-min operation is performed simply by scanning through Q from $i = 0$ to nc one by one, where i is the distance from s . We only have to scan through the distances once, and therefore the time complexity of the delete-min operation for the whole duration of the algorithm is $O(nc)$. Each vertex can be inspected exactly once at each maximal flow value. This means $O(m)$ edge inspections are performed per maximal flow value, resulting in $O(m^2)$ for the whole algorithm. Thus we have $O(m^2 + mn + nc) = O(m^2 + nc)$ as the total worst case time complexity of Algorithm 2 using the one dimensional bucket system to implement the priority queue. ■

Lemma 3.6 Algorithm 2 can run in $O(m^2 + mn \log(\frac{c}{m}))$ worst case time.

Proof We use k -level CBS to implement Q . Since there are a total of $O(mn)$ (v, d, f) triplets to be inserted into Q , the total time complexities for operations performed on Q are: $O(kmn)$ for *insert*, $O(kmn)$ for *decrease-key*, and $O(kmn + pmn + cn/p^{k-1})$ for *delete-min*, where p is the length of each level. We choose $p = (\frac{c}{m})^{1/k}$ and $k = \log(\frac{c}{m})$ to implement our k -level CBS. Then the term $O(kmn)$ dominates and the total time complexity for all three operations involving Q becomes $O(mn \log(\frac{c}{m}))$. As shown in the proof of Lemma 3.5, the total time taken for edge inspection is $O(m^2)$, resulting in the total worst case time complexity of $O(m^2 + mn \log(\frac{c}{m}))$ for Algorithm 2 using the k -level CBS to implement the priority queue. ■

Theorem 3.7 *There exists algorithms to solve the SSSP-AF problem on directed graphs with integer edge costs in $O(m^2 + nc)$ or $O(m^2 + mn \log(\frac{c}{m}))$ time bounds.*

Proof We take the same approach as discussed in the proof of Theorem 3.3. In line 17 of Algorithm 2, we store v as the predecessor vertex alongside the (w, d', f') triplet. ■

Note that in the proof of Lemma 3.6 we could have chosen $k = O(\log(\frac{c}{m})/\log \log(\frac{c}{m}))$ to speed up the algorithm by a polylog factor. We also note that if $c = 1$ then Algorithm 2 has the time complexity of $O(m^2)$, thus Algorithm 1 has not been made redundant by Algorithm 2.

4 All Pairs Shortest Paths for All Flows

The key achievement in this paper is not to come up with an original data structure but to devise algorithms to successfully utilize existing well known data structures, based on the observation that the maximum distance of any simple path on a graph with integer edge costs bounded by c is $O(nc)$. From this observation what we have effectively achieved is to find a method to *share resources*. That is, instead of having to repeatedly scan over $O(nc)$ distances for solving SSSP for each maximal flow value, we solve SSSP for all maximal flows at the same time while sharing the common resource, Q , thereby allowing us to scan $O(nc)$ only once. Takaoka (8) used a similar idea to achieve $O(mn + n^2 \log(\frac{c}{n}))$ for the APSP problem. In this section we further extend the idea of sharing common resources to solve the problem of APSP-AF.

We let $D^f[u][v]$ be the currently known shortest distance from vertex u to vertex v for maximal flow f . We extend the (v, d, f) triplet that was defined in Section 3.3 to the quadruple (u, v, d, f) , where u and v are the starting and the ending vertices of a possible path, respectively. Then we can extend Algorithm 2 to solve the APSD-AF problem, as shown in Algorithm 3

Lemma 4.1 *Algorithm 3 correctly solves APSD-AF on directed graphs with integer edge costs.*

Proof Essentially the same argument as the proof of Lemma 3.4 can be applied. The only differences are that we now have quadruples (u, v, d, f) instead of triplets in Q and S , and we need to go through more iterations as we are solving for all $O(n^2)$ pairs of vertices. Clearly these differences has no impact on the correctness of the algorithm. ■

Algorithm 3 Solve APSD-AF on graphs with integer edge costs

```

1: for all  $(u, v) \in V \times V$  do
2:   for all maximal flows  $f$  do
3:     if  $u = v$  then
4:        $D^f[u][v] \leftarrow 0$ 
5:     else
6:        $D^f[u][v] \leftarrow \infty$ 
7:    $Q \leftarrow \text{empty}$ 
8:   for all  $v \in V$  do
9:     Add  $(v, v, 0, \infty)$  to  $Q$ 
10:  for all  $(u, v) \in V \times V$  such that  $u \neq v$  do
11:    for all maximal flow  $f$  do
12:      Add  $(u, v, \infty, f)$  to  $Q$ 
13:  while  $Q$  is not empty do
14:    Delete  $(u, v, d, f)$  with minimum  $d$  from  $Q$ 
15:    for all For all  $w \neq u \in \text{OUT}(v)$  do
16:       $f' \leftarrow \min(f, \text{cap}(v, w))$ 
17:       $d' \leftarrow d + \text{cost}(v, w)$ 
18:      if  $d' < D^{f'}[u][w]$  then
19:        Let  $x = (u, w, D^{f'}[u][w], f')$  in  $Q$ 
20:         $D^{f'}[u][w] \leftarrow d'$ 
21:        Put  $x$  in the correct position in  $Q$ 
22:      if  $S[u][v]$  is empty then
23:        Append  $(d, f)$  to  $S[u][v]$ 
24:      else
25:        Let  $(d_0, f_0) \leftarrow$  last pair in  $S[u][v]$ 
26:        if  $f_0 < f$  then
27:          if  $d_0 = d$  then
28:            Delete  $(d_0, f_0)$  from  $S[u][v]$ 
29:            Append  $(d, f)$  to  $S[u][v]$ 

```

Lemma 4.2 *Algorithm 3 can run in $O(m^2n + nc)$ worst case time.*

Proof There are $O(n^2)$ pairs of vertices resulting in $O(mn^2)$ (u, v, d, f) quadruples. Each vertex pair can be observed exactly once at each maximal flow value hence the number of edge inspections that occur at one maximal flow value is $O(mn)$, resulting in the total time bound of $O(m^2n)$ for edge inspections. Using the one dimensional bucket system to implement Q , we have $O(m^2n + mn^2 + nc) = O(m^2n + nc)$ as the worst case time complexity. ■

Lemma 4.3 *Algorithm 3 can run in $O(m^2n + mn^2 \log(\frac{c}{mn}))$ worst case time.*

Proof We use the k -level cascading bucket system to implement Q , where the *delete-min* operation now takes $O(kmn^2 + pmn^2 + cn/p^{k-1})$ time. We choose $p = (\frac{c}{mn})^{1/k}$ and $k = \log(\frac{c}{mn})$ for the total time complexity of the *delete-min* operation to become $O(mn^2 \log(\frac{c}{mn}))$. Thus the total worst case time complexity using the cascading bucket system is $O(m^2n + mn \log(\frac{c}{mn}))$. ■

Theorem 4.4 *There exists algorithms to solve the APSP-AF problem on directed graphs with integer edge costs in $O(m^2n + nc)$ or $O(m^2n + mn^2 \log(\frac{c}{mn}))$ time bounds.*

Proof We take the same approach as before and modify Algorithm 3 to store v as the predecessor vertex alongside the (u, w, d', f') quadruple in line 21. ■

5 Concluding remarks

We have introduced a new graph path problem and provided non-trivial algorithms to solve the new problem that are both practical and faster than the straightforward methods.

The example of evacuation planning has been used in the introduction of the paper to show the relevance of the SP-AF problem in real life. Another possible application of the SSSP-AF problem is in computer networking, as a more sophisticated dynamic routing protocol than the currently commonly used protocols such as RIP and OSPF. And with the introduction of Software Defined Networking (SDN) (10), we can also propose APSP-AF as a possible algorithm to calculate the routes in the entire network.

Trivial lower bounds of $O(mn)$ and $O(mn^2)$ exist for the SSSP-AF and APSP-AF problems, respectively, on weighted digraphs. This paper has investigated only graphs with integer edge costs. Can we provide a better time bound than the straightforward $O(m^2 + mn \log n)$ for the SSSP-AF problem on directed graphs with real edge costs? Is there a faster algorithm on undirected graphs? How close can we get to the lower bounds of the SP-AF problems? We conclude the paper with these open questions and look forward to further research that may address these open problems.

References

- [1] Ahuja, K., Melhorn, K., Orlin, J. B., & Tarjan, R. E. (1990), Faster algorithms for the shortest path problem, in ‘Journal of ACM’, Vol. 37, pp. 213–223
- [2] Denardo, E. V. & Fox, B. L. (1979), Shortest-route methods: I. Reaching, pruning, and buckets, in ‘Operations Research’, Vol. 27, pp. 161–186
- [3] Dijkstra, E. (1959), A note on two problems in connexion with graphs, in ‘Numerische Mathematik’, Vol. 1, pp. 269–271
- [4] Floyd, R. (1962), Algorithm 97: Shortest Path, in ‘Communications of the ACM’, Vol. 5, pp. 345
- [5] Fredman, M. (1976), New bounds on the complexity of the shortest path problem, in ‘SIAM Journal on Computing’, Vol. 5, pp. 83–89
- [6] Fredman, M. & Tarjan, R. (1987), Fibonacci heaps and their uses in improved network optimization algorithms, in ‘Journal of the Association for Computing Machinery’, Vol. 34, pp. 596–615
- [7] Hu, T. C. (1961), The maximum capacity route problem, in ‘Operations Research’, Vol. 9, pp. 898–900
- [8] Takaoka, T. (2012), Efficient Algorithms for the All Pairs Shortest Path Problem with Limited Edge Costs, in ‘Proceeding of 18th CATS’, pp. 21–26
- [9] Thorup, M. (2003), Integer Priority Queues with Decrease Key in Constant Time and the Single Source Shortest Paths Problem, in ‘Proceeding of 35th STOC’, pp. 149–158
- [10] 2012ONF Open Networking Foundation (2012), Software-Defined Networking: The New Norm for Networks, in ONF White Paper

Lazy and Eager Approaches for the Set Cover Problem

Ching Lih Lim

Alistair Moffat

Anthony Wirth

Department of Computing and Information Systems
The University of Melbourne
Victoria 3010, Australia

Abstract

The SET COVER problem is tantalizingly simple to describe: given a collection F of sets, each containing a subset of a universe U of objects, find a smallest sub-collection A of F such that every object in U is included in at least one of the sets in A . However, like many such combinatorial problems, SET COVER is NP-hard, meaning that it is unlikely that an efficient algorithm will be found, and that approximation algorithms must be preferred for non-trivial problem instances. One well-known approximation approach for SET COVER is to repeatedly add the set with the most uncovered items to the solution, continuing until every element in the universe is covered; this GREEDY approach has a provable logarithmic approximation ratio, essentially the best feasible ratio. Here we study the implementation of the GREEDY approach to SET COVER, evaluating eager and lazy versions and other implementation options. Experiments with several large datasets demonstrate that lazy “as required” priority queue updates should be preferred, rather than eager “as soon as possible” ones; and that when implemented in this way, the GREEDY mechanism can solve some large instances of the SET COVER problem very quickly. This practical superiority contrasts with the lazy version’s having a demonstrably higher worst-case operation cost.

Keywords: set cover, priority queue, lazy update, approximation algorithm.

1 Introduction

The SET COVER problem arises in a very broad range of logistic and layout problems, including monthly beer production, the roll-out schedule of a network, and data mining of a collection of sequenced DNA data for simple repeats [11, 12]. An instance of SET COVER consists of a universe U of objects, and a collection F of subsets of U . The challenge is to identify a smallest sub-collection of F subject to the constraint that the union of the sets in A

must be equal to the union of the sets in F ; that is, A must cover the universe [3, 4, 12].

For example, suppose that the city government wishes to select a set of locations for fire stations so that every home is less than 5 km by road from at least one fire station, and so that the number of fire stations required is minimized. In this scenario, the set of homes forms the universe U of items that must be covered by any solution; and the i th possible location for a fire station gives rise to a set S_i of homes that are within 5 km of it. Assuming that the total pool of possible locations is sufficiently numerous that every home is within 5 km of one or more of the locations under consideration (equivalently: that $\bigcup_{S \in F} S = U$), the required SET COVER solution is a smallest set of locations A such that $\bigcup_{S \in A} S$ is also equal to U . That is, what is required is a smallest subset $A \subseteq F$ such that every home in U appears in at least one element in A . If there is more than one minimal-size solution, secondary criteria could be used to choose between them. Other SET COVER scenarios include information retrieval, where a query over words (elements) computes the smallest sub-collection of documents, drawn from a large collection of documents, that as a whole contain all of the query words.

Like many such combinatorial problems, SET COVER is NP-hard. Consequently, to find optimal solutions, some exhaustive search component is required; hence, only trivially small problem instances can be solved exactly within reasonable resource bounds. Approximation techniques must then be used for large-scale problem instances. Any approximation technique is a compromise between solution *effectiveness* and implementation *efficiency*, with high effectiveness usually only possible at the cost of low efficiency. At one extreme in this spectrum, exhaustive methods generate optimally effective solutions, but typically require time that is exponential in the size of the problem instance; at the other extreme, taking (for the SET COVER problem) $A = F$ certainly guarantees coverage and is very fast to compute, but might be ineffective by a very large multiplicative factor. Between these extremes, a good approximation algorithm balances a guarantee on effectiveness (for example, in the case of SET COVER, through a proof that the solution generated by the approximation algorithm is at most some bounded factor larger than the size of an optimal solution) with desirable asymptotic execution analysis. For most NP-hard problems, the latter requirement corresponds to running time that is polynomial in the size of the problem instance.

The GREEDY algorithm for SET COVER, introduced in detail in Section 2, can be measured against these criteria. The solutions it constructs are bounded relative to the size

of a corresponding optimal solution [6, 10], and it executes in polynomial time. It remains an important focus of study because it has provably the best approximation factor possible in polynomial time, is known to perform very well in practice across multiple kinds of instances and often generates solutions that are within 10% of optimum [7, 8]. Recent GREEDY variants include a disk-based version [4], and fast parallel algorithms [2, 3]; these options add to the versatility and usefulness of the approach.

The GREEDY algorithm proceeds by repeatedly adding to the solution the remaining set with the largest number of uncovered elements, seeking to obtain the greatest gain for each additional set that is used. To find the set with the greatest number of uncovered elements a priority queue data structure is employed to track the size of every candidate set. That structure is one of the focuses of this paper, as we consider in detail how to implement the GREEDY approach, and balance the costs of different types of queue update operations. One obvious option is to use a binary heap, but other – simpler – structures can also be used. A critical issue that affects the choice is the rate at which updates are performed – as each set is added to the growing solution, a large number of the other remaining sets might need to have their “uncovered element” counts decreased. In this EAGER-GREEDY implementation, the priority queue must always be able to quickly emit the next set to be added to the solution, meaning that many updates to set sizes might be required at each iteration.

To counter this possible inefficiency, we explore the use of lazy update options, in which changes to sets are deferred as long as possible and until they are actually necessary. The resultant implementation, LAZY-GREEDY, performs a different balance of elementary operations, and executes more quickly than the EAGER-GREEDY version on public datasets. Our experimental results indicate that LAZY-GREEDY outperforms the eager greedy method in terms of running time, and is also competitive against a recently described greedy bucket-based algorithm [4].

The notion of lazy evaluation can be further applied to the process of updating a set, that is, identifying in it (and removing from it) covered items. We explore this option too, terminating those operations as soon as it can be known that the set in question will not be the next one added to the solution.

The remainder of the paper is structured as follows. Section 2 explains the technical background of the SET COVER problem; shows how the GREEDY strategy computes solutions with a guaranteed approximation bound; and describes the EAGER-GREEDY implementation of that approach. Section 3 focuses on the bottleneck issue with EAGER-GREEDY, and describes an alternative lazy-update version of the same GREEDY paradigm. We also describe an input pattern that gives rise to a worst-case number of set updates. Section 4 describes the experiment environment and datasets used to evaluate the pool of SET COVER algorithms; and then demonstrates that the new lazy approach solves large instances of the problem significantly faster than does the eager implementation of the same procedure. The lazy implementation is also competitive against the more complex DF-GREEDY bucket-based implementation of the eager greedy procedure developed by Cormode et al. [4]. Extending the notion of “laziness” to the set difference operator is considered in Section 5. Section 6 then concludes our presentation.

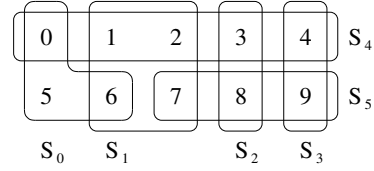


Figure 1: An instance of the SET COVER problem over $U = [0..9]$, with $F = \{S_0, \dots, S_5\}$, and $S_0 = \{0, 5, 6\}$, $S_1 = \{1, 2, 6, 7\}$, $S_2 = \{3, 8\}$, $S_3 = \{4, 9\}$, $S_4 = \{0, 1, 2, 3, 4\}$, and $S_5 = \{7, 8, 9\}$.

2 The SET COVER Problem

We now state the SET COVER problem, and describe the best-known approximation algorithm for solving it.

2.1 Definition

The universe of objects under consideration is supposed, without loss of generality, to consist of the n integers $[0..n-1]$, where, as a general notation, $[0..z-1]$ stands for the set $\{0, 1, \dots, z-1\}$. An instance of the SET COVER problem relative to the universe $U = [0..n-1]$ is given by a collection F of m sets $\{S_0, \dots, S_{m-1}\}$, with $S_i \subseteq U$, and $\bigcup_{i \in [0..m-1]} S_i = U$. That is, it is known that every item in U appears in at least one of the sets S_i in F .

The challenge then posed is to find a smallest sub-collection $A \subseteq F$ such that $\bigcup_{S \in A} S = \bigcup_{S \in F} S = U$ [12]. The *size* of the solution is given by $|A|$. Another way of categorizing the cover size that we do not pursue in this paper is the total number of elements in the sets making up A , that is, the objective is to minimize $\sum_{S \in A} |S|$ subject to the constraint $\bigcup_{S \in A} S = U$. It is convenient to describe the set A in terms of the *indices* of the sets that are being included in the solution. The interpretation of A to be used in a particular situation will always be clear from the context.

Figure 1 illustrates a simple instance of the SET COVER problem in which $U = [0..9]$ and $F = \{S_0, \dots, S_5\}$. It is also helpful to define a value M as the total of the m sets that are involved in a SET COVER instance, $M = \sum_{i \in [0..m-1]} |S_i|$. In the example, $M = 19$, and is a measure of the size of the input required when describing the instance. The smallest solution for the instance in Figure 1 is $A = \{0, 4, 5\}$, since these three sets cover U , and there is no smaller sub-collection of F that also covers U . The alternative sub-collection $\{S_0, S_1, S_2, S_3\}$ also covers U and is free of redundancy (none of the sets can be removed without coverage being lost), but is not optimal.

The SET COVER problem is NP-hard [12]. Indeed, Feige [6] showed that, unless NP has “slightly super-polynomial time algorithms”, SET COVER cannot be approximated within a factor of $(1 - o(1)) \ln n$. The next section presents the GREEDY approach, whose approximation factor is very close to this bound.

2.2 The GREEDY Strategy

Algorithm 1 provides an overview of the GREEDY approximation mechanism for SET COVER. The idea behind it is very simple: at each iteration of the loop, the set with the largest number of uncovered items is identified, and added to a growing solution A . That process is continued until every element in U is covered; the precondition

Algorithm 1 The GREEDY approach

Input: Family F of m sets $S_i \subseteq U = [0..n-1]$, and with $\bigcup_{i \in [0..m-1]} S_i = U$

Output: Set A of indices of sets with $\bigcup_{i \in A} S_i = U$.

```

1:  $A \leftarrow \{\}$ 
2:  $covered \leftarrow \{\}$ 
3: while  $covered \neq U$  do
4:    $i \leftarrow \operatorname{argmax}_{j \in [0..m-1]} \{|S_j - covered|\}$ 
5:    $A \leftarrow A \cup \{i\}$ 
6:    $covered \leftarrow covered \cup S_i$ 
7: end while
8: return  $A$ 
    
```

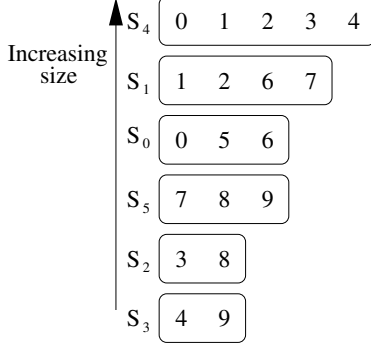


Figure 2: The six sets of the SET COVER instance shown in Figure 1, sorted into increasing size order. The first set to be considered is S_4 .

that $\bigcup_{i \in [0..m-1]} S_i = U$ guarantees that the termination condition for the loop will be met.

Figure 2 shows the initial state of Algorithm 1, as applied to the problem instance shown in Figure 1. The GREEDY approach first adds set S_4 to the solution, and in doing so, covers items $\{0, 1, 2, 3, 4\}$. The second set considered is then S_5 , since it (now) has three uncovered items, more than the larger set S_1 , which only has two uncovered items. Once S_5 is added to the solution A , there are only two uncovered items, 5 and 6, and S_0 covers both of them, whereas S_1 only covers one of them. With S_0 included in the solution, the main loop terminates; in this case, the optimal solution is identified.

Johnson [10] demonstrated that the GREEDY approach constructs a solution that is not more than $1 + \ln n$ times larger than the minimal one; improved bounds were provided by Slavík [13]. Young [14] and Dutta [5] provide surveys of this development. The GREEDY approach also performs well in practice, typically identifying solutions that are close to optimal [3, 4]. Extensions include parallel implementations designed for multi-processor architectures [2]; for disk-based operation [4]; enhancements that improve the worst-case approximation ratio [9]; and methods for on-line SET COVER problems [1].

2.3 Implementing GREEDY: Eager Evaluation

Algorithm 1 leaves many details unspecified. Critical decisions that are required include:

- How to represent the sets S_i ;
- How to manage the collection of sets so that the computation implied by the *argmax* evaluation at step 4 can be carried out; and
- How to represent the set *covered* so that the computations at steps 4 and 6 can be carried out.

Algorithm 2 The EAGER-GREEDY implementation

Input: As for Algorithm 1

Output: As for Algorithm 1

```

1:  $A \leftarrow \{\}$ 
2:  $covered \leftarrow \{\}$ 
3:  $pqueue \leftarrow \{\}$ 
4: for  $j \leftarrow 0$  to  $m-1$  do
5:    $pqueue.insert(\langle |S_j|, j \rangle)$ 
6:    $updates[j] \leftarrow \{\}$ 
7: end for
8: while  $covered \neq U$  do
9:    $\langle uc, i \rangle \leftarrow pqueue.maximum()$ 
10:   $A \leftarrow A \cup \{i\}$ 
11:  for  $c \in S_i - covered$  do
12:     $covered \leftarrow covered \cup \{c\}$ 
13:    for each set  $S_j$  that contains  $c$  do
14:       $updates[j] \leftarrow updates[j] \cup \{c\}$ 
15:    end for
16:  end for
17:  for all  $j$  for which  $updates[j] \neq \emptyset$  do
18:     $\langle uc, j \rangle \leftarrow pqueue.delete(j)$ 
19:     $S_j \leftarrow S_j - updates[j]$ 
20:     $pqueue.insert(\langle uc - |updates[j]|, j \rangle)$ 
21:     $updates[j] \leftarrow \{\}$ 
22:  end for
23: end while
24: return  $A$ 
    
```

Algorithm 2 shows one way in which these issues can be resolved. An explicit priority queue is introduced, with the usual operations assumed: *insert()*, to add a tuple $\langle uc, i \rangle$ consisting of a set label i and queue weight uc (steps 5 and 20); *maximum()*, to remove and return the tuple with the largest weight uc (step 9); and *delete(j)*, to locate and remove the tuple with the specified label j (step 18). As well, details are given of the process to be followed when a set S_i is added to the solution. The loop at step 11 iterates over the newly covered items in S_i , and removes each of them from the every other set S_j in which it appears (step 13). Each such removal decrements the “uncovered symbol count” uc of a set S_j , but these changes are all deferred until they can be processed in a batch at step 18, which locates that set in the priority queue, extracts its current uc value, and then, at step 20, puts it back in the priority queue with a reduced uc value. The batching process ensures that each S_j that shares items with S_i is updated just once per occurrence of step 9.

Two further details require elaboration. First, in order to identify at step 11 the items c that are being newly covered, it is necessary for each element in S_i to be checked for membership in *covered*. The most appropriate mechanism is for *covered* to be stored as an indexed bitvector, with $covered[c]$ set to 1 if and only if $c \in covered$. Each lookup to $covered[c]$ requires $O(1)$ time; over all sets S_i , the cost is at most $\sum_{i \in [0..m-1]} |S_i| = M$, that is, is linear in the size of the input description. Second, step 13 requires knowledge of the collection of sets S_j that contain a given element c . An efficient way of supporting this need is to pre-compute an inverted index over the sets, so that a mapping is available from items to sets containing them [4]. Building an inverted index requires $O(M)$ time using distribution-sorting techniques, plus a corresponding amount of memory space to store it.

2.4 Priority Queue

With structures for *covered* and for identifying affected sets in place, the overall time required by EAGER-GREEDY is determined by the choice of priority queue structure: there are at most $|F| = m$ priority queue *maximum()* operations required at step 9; and at most M decrease-weight operations required at steps 18–20. If the priority queue is implemented as a binary heap each operation requires $O(\log m)$ time (since there are at most m sets in the heap); and the large number of update operations dominates the running time, giving rise to an $O(M \log m)$ overall cost.

An obvious question is whether a binary heap is in fact the best choice. When a large number of decrease-weight operations must be balanced against a smaller number of extract-maximum operations, other options are also possible. The fact that all of the item weights manipulated in the priority queue are integers between 0 and $n - 1$ also adds flexibility.

In particular, a simple array-plus-lists structure can be used as a priority queue. An array *pqueue* of n elements is maintained, with *pqueue*[*uc*] a list of the indices of the sets that currently have *uc* uncovered elements. A variable *pqueue.top* indicates the largest index in *pqueue* that is non-null; and after every extract-maximum operation, *pqueue.top* is updated via a sequential scan through *pqueue* looking for the next non-null entry. To carry out the decrease-weight operation at steps 18–20, the current entry in *pqueue* for set *j* is identified via a table indexed by set identifier; and that node is deleted from its current list *pqueue*[*uc*], then re-linked in to *pqueue*[*uc* - $|updates[j]|$] where $|updates[j]|$ is the number of changes being made to S_j . This takes $O(1)$ time per set moved. In addition, an auxiliary stack records the values of *j* for which *updates*[*j*] is non-empty, to that the loops at step 17 does not introduce unnecessary overhead.

Allowing for the fact that the total cost of all of the scanning required as part of the extract-maximum operations is $O(n)$, the execution time for EAGER-GREEDY is now $O(n + M)$. Moreover, since the collection *F* covers *U*, it must be that $M \geq n$. Hence, with this queue structure, the EAGER-GREEDY approach described in Algorithm 2 executes in $O(M)$ time.

3 A Lazy Implementation

Section 2.4 describes one way in which the small number of extract-maximum operations can be balanced against a much larger number of decrease-weight operations. But a significant imbalance remains between the number of extract-maximum operations and the number of decrease-weight operations, and it is interesting to ask if the overall envelope of operations can be decreased by explicitly trading a large drop in the number of decrease-weight operations for a smaller increase in the number of extract-maximum operations.

3.1 Deferred Cleaning

In Algorithm 2, each execution of step 9 returns the next largest set, according to the number of uncovered elements. Then, as that set is added to the solution, every other set that contains common uncovered elements is adjusted. Those adjustments ensure that the counts of uncovered elements for all sets are correct at all times.

Algorithm 3 The LAZY-GREEDY implementation

Input: As for Algorithm 1

Output: As for Algorithm 1

```

1:  $A \leftarrow \{\}$ 
2:  $covered \leftarrow \{\}$ 
3:  $pqueue \leftarrow \{\}$ 
4: for  $j \leftarrow 0$  to  $m - 1$  do
5:    $pqueue.insert(\langle |S_j|, j \rangle)$ 
6: end for
7: while  $covered \neq U$  do
8:    $\langle uc, i \rangle \leftarrow pqueue.maximum()$ 
9:    $S'_i \leftarrow S_i - covered$ 
10:  if  $|S'_i| = |S_i|$  then // all items in  $S_i$  were clean
11:     $A \leftarrow A \cup \{i\}$ 
12:     $covered \leftarrow covered \cup S_i$ 
13:  else // some items in  $S_i$  were covered
14:     $S_i \leftarrow S'_i$ 
15:     $pqueue.insert(\langle |S_i|, i \rangle)$ 
16:  end if
17: end while
18: return  $A$ 
```

If those counts were not reduced so relentlessly, the putative sizes of the competing sets – the weights manipulated by the priority queue – would deviate from their true values. But they would always be *upper* bounds, and the worst that could happen is that a set might be emitted by the priority queue as being the next largest, only for a subsequent check to find that in fact it contained covered elements that had not yet been noted, and could not be included as part of the greedy solution just yet.

Algorithm 3 presents the revised process. Step 8 accesses the largest set in the priority queue, denoted as S_i . In recognition of the fact that S_i might include one or more items covered in previous iterations, the first processing undertaken is a set difference $S_i - covered$, in which each element of S_i is checked against the set *covered*. If the set difference operation removes no elements from S_i , then all elements were uncovered, and it can be added to the greedy solution (steps 11–12).

But if there were covered elements present in S_i , it cannot be added to the solution, since it might not be the set containing the largest number of uncovered items – there might be another smaller set that has that claim. Instead, the cleaned set S'_i is retained, and is reinserted into the priority queue (step 15). The main loop then iterates, and the newly largest set is chosen as the next maximum. This process continues until a clean set containing only uncovered items emerges from the priority queue.

With the revised arrangement, the while loop at step 7 executes more than $|A|$ times. But each reinsertion operation at step 15 should move the set S_i down by multiple slots in the priority queue ordering; that is, there might be fewer reinsertions in total, decreasing the total number of queue operations required.

3.2 Example

Consider the example instance shown in Figure 1. The initial state of the priority queue is as shown in Figure 2. The largest set, S_4 , is removed, and because no items have been covered yet, all of its elements are clean, and S_4 is added to the solution. The next largest set is S_1 , with a putative *uc* of 4. But when S_1 is checked for cleanliness, it is discovered that two of its elements (1 and 2) have become covered, and its uncovered size is actually only two. So a cleaned set $S'_1 = \{6, 7\}$ is added back into the

i	Set S_i												
0	<table><tr><td>9</td><td>8</td><td>7</td><td>6</td></tr><tr><td>5</td><td>4</td><td>3</td><td>2</td></tr><tr><td>1</td><td>0</td><td></td><td></td></tr></table>	9	8	7	6	5	4	3	2	1	0		
9	8	7	6										
5	4	3	2										
1	0												
1	<table><tr><td>9</td><td>8</td><td>7</td><td>6</td></tr><tr><td>5</td><td>4</td><td>3</td><td>2</td></tr><tr><td>1</td><td>X</td><td>X</td><td></td></tr></table>	9	8	7	6	5	4	3	2	1	X	X	
9	8	7	6										
5	4	3	2										
1	X	X											
2	<table><tr><td>9</td><td>8</td><td>7</td><td>6</td></tr><tr><td>5</td><td>4</td><td>3</td><td>X</td></tr><tr><td>X</td><td>X</td><td>X</td><td>X</td></tr></table>	9	8	7	6	5	4	3	X	X	X	X	X
9	8	7	6										
5	4	3	X										
X	X	X	X										
3	<table><tr><td>9</td><td>8</td><td>7</td><td>6</td></tr><tr><td>X</td><td>X</td><td>X</td><td>X</td></tr><tr><td>X</td><td>X</td><td>X</td><td>X</td></tr></table>	9	8	7	6	X	X	X	X	X	X	X	X
9	8	7	6										
X	X	X	X										
X	X	X	X										

Figure 3: Pathological example with $m = 4$ sets. Items shown as “X” are unique, and appear one time only.

queue, with $uc = 2$; nothing is added to the solution at this loop iteration. Now the next largest set is S_0 , but it is also discovered to contain covered items, and is returned to the queue as $S'_0 = \{5, 6\}$. Next, S_5 is emitted from the queue, and it is clean – there are no covered items. So S_5 is added to the solution, and elements 7, 8, and 9 are marked as having been covered.

By this stage, the queue contains S_2 , S_3 , S'_1 , and S'_0 , all of size two. Assuming that they are considered in that order, S_2 is found to contain no uncovered elements; then S_3 is likewise found to be of no use; and then S'_1 is reduced to $S''_1 = \{6\}$ and is returned to the queue.

Now the queue contains $S'_0 = \{5, 6\}$ and $S''_1 = \{6\}$. The processing of S'_0 reveals that both of its items are as yet uncovered, and so S_0 is added to the solution A . That addition results in all items being covered, and so the final solution is $A = \{4, 5, 0\}$, the same solution as the EAGER-GREEDY approach.

In this particular example, a total of 8 extract-maximum operations are required compared to just 3 by EAGER-GREEDY. The anticipated payoff for that growth is a reduced number of check-covered operations; that gain does not arise in the example because of its small scale, but should be measurable on larger instances.

3.3 Worst-Case Performance

The change to lazy evaluation means that the number of iterations of step 8 is no longer bounded by m , the number of sets input to the problem. This observation gives rise to an interesting question: What is the maximum number of iterations possible, that is, is there a limit on the number of times that a given set can be reinserted at step 15 before it emerges “clean” and able to be added to the solution?

Consider the set structure shown in Figure 3. Each set contains a hierarchy of items present in other sets in the collection, marked by the various boxes; plus additional unique filler elements denoted “X”, whose role is to force the ordering of the queue operations. In a problem instance over m sets S_0 to S_{m-1} there are thus $m(m+1)/2$ different items that appear as part of the hierarchical pattern. In addition, set S_i , contains $i(i+3)/2$ unique filler elements, meaning that the total filler count is

$$\sum_{i=0}^{m-1} \frac{i(i+3)}{2} = \frac{m^3 + 3m^2 - 4m}{6}.$$

Hence, in total, the universe size is given by $n = (m(m+1)/2) + (m^3 + 3m^2 - 4m)/6 = (m^3 + 6m^2 - m)/6$. A similar computation shows that the input size is given by

$$M = \frac{m^3 + 2m^2 - m}{2} \approx \frac{m^3}{2}.$$

When processed by LAZY-GREEDY, the input arrangement shown in Figure 3 gives rise to a repetitive cycle of computations. It first processes set S_{m-1} , finds that all of its elements are clean, and hence covers them. It then cycles through S_{m-2} , S_{m-3} and so on down to S_0 , cleaning the same group of m elements out of each. Once all $m-1$ remaining sets are clean, set S_{m-2} returns to the head of the queue in a fully uncovered state, and the cycle is repeated recursively, exactly as if a pathological instance of $m-1$ sets had been input.

Since set S_i cycles through the head of the priority queue a total of $m-i$ times, the total number of executions of step 8 in Algorithm 3 is

$$\sum_{i=0}^{m-1} (m-i) = \frac{m(m+1)}{2} \approx \frac{m^2}{2}.$$

Hence, the number of extract-maximum operations performed on the pathological input instances is $\Theta(M^{2/3})$, and remains sub-linear in the size of the input description. That asymptotic bound also applies to the number of times that step 15 is executed.

The other place in the LAZY-GREEDY approach where the number of iterations might be different from the EAGER-GREEDY approach is at step 9. Each time a set S_i reaches the head of the queue, every element c in it is checked against *covered*, taking $O(1)$ time per element. Elements that have already been covered are removed as they are discovered, and only the uncovered elements are retained in the set S'_i that is reinserted in to the queue. But the elements that were not covered will be checked again when S_i reaches the head of the queue next time, and it is no longer possible to bound the number of “check covered” operations by M . Indeed, for the pathological input structure that is illustrated in Figure 3, the total number of check-covered operations for an instance of size m , is given by the summed size of the (recursive) problem instances from size m down to size 1:

$$\sum_{i=1}^m \frac{i^3 + 2i^2 - i}{2} = \Theta(m^4).$$

The behavior of the pathological input arrangement can be summarized thus:

Observation 1. *There is a SET COVER problem instance of size M that requires the LAZY-GREEDY method to spend $\Theta(M^{4/3})$ time, including $\Theta(M^{2/3})$ extract-maximum operations in the priority queue.*

Note that this time bound is asymptotically greater than the $O(M)$ time that is achieved by the EAGER-GREEDY implementation. Fortunately, typical problem instances do not display the particular structure required to force this bad behavior, and when executed on typical instances, the LAZY-GREEDY approach shows substantial benefits compared to the EAGER-GREEDY implementation.

4 Experimental Results

We now demonstrate the difference that lazy evaluation can make on typical SET COVER problem instances.

4.1 Datasets and Experiment Environment

Three datasets are used in the experiments – RETAIL, ACCIDENTS and WEBDOCS – taken from the Frequent Item-

Dataset	Variable	RETAIL	ACCIDENTS	WEBDOCS
Universe size	n	16,470	468	5,267,656
Number of sets	m	88,162	340,183	1,692,082
Largest set size		76	51	71,472
Median set size		8	34	98
Average set size		10.31	33.81	177.23
Total input size	M	908,576	11,500,870	299,887,139

Table 1: The properties of the datasets involved in our experiments.

Set Mining dataset repository¹. The RETAIL dataset represents anonymized data on shopping items purchased by customers in Belgium, with a set-cover solution being a minimal set of customer transactions such that every item stocked in the supermarket is purchased at least once; ACCIDENTS is a set of traffic accident records in a certain time period also in Belgium, with a set-cover representing a subset of them in which every one of the recorded variables occurs at least once; and WEBDOCS represents a crawled collection of Web html documents, with a set-cover solution being a smallest set of documents such that every word that appears in any of the documents appears in at least one of the documents in the subset. Each line in the datasets is a space-separated sequence of integers. Table 1 describes the properties of the datasets.

Experiments were conducted on a server running x86-64 Red Hat Enterprise Linux ES Release 4 (Nahant Update 7) with four 3.2 GHz Intel Xeon processors and 8 GB of primary memory, with programs implemented using the 64-bit version 1.5.0.08 of the Java Runtime Environment. The execution times presented below are all the average of ten runs of the corresponding scenario, in all cases starting with a configuration in which the set data structure is available in memory.

4.2 Implementations

Java implementations of EAGER-GREEDY and LAZY-GREEDY were prepared. All of the implementation executions were restricted to the usage of a single thread by default and had the runtime JIT (Just-In-Time) optimization enabled by default. All data structures were presumed to fit in to main memory. In the case of the EAGER-GREEDY implementation, the first operation carried out was to construct the index that is required, with that time included in the reported execution times.

The internal representations for the input sets and for *covered* were carefully chosen. A Boolean bit-vector of $\lceil n/32 \rceil$ four-byte integers represents the set *covered*, allowing constant-time check-covered operations. Each of the sets was stored in an allocated segment out of an array of M integers, with covered elements rotated to the end of each set's zone, and uncovered elements retained at the front of each zone. There was no resizing or de-fragmenting the underlying array.

The priority queue *pqueue* for tracking the set sizes was implemented as an array of size n , in which each bucket was indexed by *uc*, and contained the identifier of the first set of that size, with other sets threaded from that first one. In the case of the EAGER-GREEDY mechanism, the priority queue lists were doubly-linked, and contained three fields: a *next* pointer, a *prev* pointer, and the *uc* value for that set, the latter being needed to allow the

Structure	EAGER-GREEDY	LAZY-GREEDY
<i>covered</i>	$n/32 + O(1)$	$n/32 + O(1)$
<i>pqueue</i>	$n + 3m + O(1)$	$n + m + O(1)$
<i>sets</i>	$m + M + O(1)$	$m + M + O(1)$
<i>index</i>	$n + M + O(1)$	—

Table 2: Space cost, in integers.

pqueue.delete(j) operation to be efficient (step 18 in Algorithm 2). Double threading of list nodes was required, because the nodes being deleted might be anywhere in their list. Access to the j th set, required at the same step, is achieved in $O(1)$ time by simply storing the set of priority queue nodes in a single large array (rather than as independently malloc'ed objects), with S_j stored in the j th array element. Each list was implemented as last-in first-out structure, with insertions always at the head.

In the LAZY-GREEDY approach a similar priority queue is maintained, but because deletions only occur as extract-maximum operations at the front of each list, no *uc* value is required, and single-threading suffices.

These various considerations lead to the space costs shown in Table 2. For the file WEBDOCS, the total nominal space requirement for the EAGER-GREEDY implementation was thus 2,355 MB; with 1,178 MB required for the LAZY-GREEDY implementation. That is, the index that is required for the EAGER-GREEDY approach essentially doubles the amount of memory space required. All of the values are stored as integer indices into either *pqueue* or into the array of sets. Naturally, if any of n , m , or M is greater than 2^{32} , then eight-byte integers would be required rather than the four-byte ones assumed in these calculations.

4.3 Worst-Case Behavior

Table 3 shows execution times and operation counts for the EAGER-GREEDY and LAZY-GREEDY implementations when applied to the extreme problem instances described in Section 3.3. As anticipated by Observation 1, when m doubles, the number of extract-maximum operations required by LAZY-GREEDY increases by a factor of four, and both the number of check-covered operations required and the measured running time increase by a factor of sixteen. As m doubles, the input size M increases by a factor of roughly eight. For SET COVER instances that have this structure, the extra space required by the EAGER-GREEDY approach is a sound investment, and prevents super-linear (in M) execution times.

¹<http://fimi.ua.ac.be/data/>

m	M	time (s)		extract-maximum		check-covered	
		E-G	L-G	E-G	L-G	E-G	L-G
200	4,039,900	0.37	1.17	200	20,100	4,039,900	204,681,650
300	13,589,850	1.28	5.65	300	45,150	13,589,850	1,028,283,725
400	32,159,800	2.98	17.47	400	80,200	32,159,800	3,237,393,300
500	62,749,750	5.72	39.92	500	125,250	62,749,750	7,885,510,375
600	108,359,700	9.88	85.36	600	180,300	108,359,700	16,326,134,950

Table 3: Execution time and operation counts LAZY-GREEDY and EAGER-GREEDY on pathological input sequences.

Measurement	Dataset	EAGER-GREEDY	LAZY-GREEDY	DF-GREEDY
Solution size	RETAIL	5,106	5,113	5,119
	ACCIDENTS	181	180	185
	WEBDOCS	406,372	406,400	406,428
Extract-maximum	RETAIL	5,106	162,961	—
	ACCIDENTS	181	1,080,524	—
	WEBDOCS	406,372	3,291,585	—
Check-covered	RETAIL	92,540	1,239,924	1,223,062
	ACCIDENTS	6,630	18,170,446	17,766,461
	WEBDOCS	133,008,957	335,085,502	326,336,293
Queue reinsertions	RETAIL	778,704	74,817	74,009
	ACCIDENTS	2,365,299	743,400	709,066
	WEBDOCS	46,092,276	1,599,610	1,562,286
Running time (s)	RETAIL	0.39	0.18	0.18
	ACCIDENTS	1.53	0.85	0.84
	WEBDOCS	73.10	5.34	5.11

 Table 4: Performance of EAGER-GREEDY, LAZY-GREEDY and DF-GREEDY on three datasets. The DF-GREEDY method is measured using parameter $p = 1.1$. The best value in each row is highlighted.

4.4 Practical Applications

Table 4 shows what happens on the three problem instances described in Table 1. The final column is discussed in Section 4.5. As currently implemented, the time required to read each file (text format) is 1.17 sec, 4.82 sec, and 111.24 sec, for RETAIL, ACCIDENTS and WEBDOCS, respectively. These quantities are *not* included in the running times in Table 4.

In the first block of values, the solution sizes $|A|$ are listed. The variation between EAGER-GREEDY and LAZY-GREEDY, and the absence of a single “right” answer, is a consequence of differences in the way that equal-sized sets are handled. If a secondary key (such as set number), was introduced to the *pqueue* ordering, the implementations could be made to give the same answer. The drawback of doing that would be that queue reinsertions could no longer be made at the head of the list *pqueue[uc]*, and would potentially add to the execution time. Nor could there be any expectation that the single answer that would be generated through the use of a tie-breaking rule would always be the smaller of the two alternatives. Hence the retention of the current arrangement, which by chance alone happens to favor the EAGER-GREEDY implementation on two of the three datasets.

The next three blocks in Table 4 give detailed operation counts for extract-maximum, check-covered, and queue-reinsert operations for the three datasets. As anticipated, the EAGER-GREEDY implementation performs a minimal number of extract-maximum operations, just one per set

added to the solution A , whose size is at most m . But there is a very large number of queue reinsertion operations performed, close to M , and even with the array-based priority queue described in Section 2.4, each reinsertion is relatively costly, involving multiple tests for boundary cases and multiple pointer assignments.

Compared to EAGER-GREEDY, the LAZY-GREEDY implementation reduces the number of reinsertion operations by a large or (on WEBDOCS) very large factor. The tradeoff is that the number of extract-maximum and check-covered operations increases. But individual check-covered operations are fast (just an array access and a mask operation), and even extract-maximum operations are less costly than reinsertions.

The last block in Table 4 shows the measured execution time, including the cost of index construction in the case of EAGER-GREEDY. On the three test datasets LAZY-GREEDY does indeed execute more quickly, by factors that vary from 2.2 (RETAIL) to 13.7 (WEBDOCS).

4.5 Disk-Friendly Greedy

Cormode et al. [4] have also considered the question of how to implement the GREEDY method for SET COVER. Their DF-GREEDY (disk-friendly greedy) approach is designed to minimize the number of non-sequential operations, and maximize the number of sequential operations over the data, and hence provides good performance when the input data is so voluminous that it cannot be stored in main memory. Cormode et al. introduce a fidelity param-

eter $p > 1$ that permits further imprecision of the output size, in order to control the number of disk seek operations required. That is, the DF-GREEDY approach can be thought of as being an “approximate-approximation” mechanism, with a slightly weakened performance guarantee of $1 + p \ln n$ [4].

The DF-GREEDY implementation has much in common with the LAZY-GREEDY approach described here. The critical difference is that instead of the priority queue being indexed directly by current set size, as given the by uc component of the tuple, it is instead indexed by $\lfloor \log_p uc \rfloor$; that is, the number of distinct buckets in $pqueue$ is given by $\lfloor \log_p n \rfloor$. Each bucket is represented as a disk file containing explicit sets, and the only requirement for memory space in the on-disk version of DF-GREEDY is the $n/32$ words required for the *covered* bit-vector.

The main processing loop of the DF-GREEDY method works systematically through the sets in a given bucket, for example, the k th bucket containing sets of at least p^k and less than p^{k+1} elements. As each set is read from the corresponding file and processed, it is checked for cleanliness. Provided that a set is clean, or nearly clean – where nearly clean is defined as containing at least p^k uncovered items – it is incorporated into the solution A . On the other hand, if, after cleaning, the set contains fewer than p^k uncovered items, it is written back to disk by appending it to the file storing the corresponding range of set sizes, and held for later processing.

The use of a geometric sequence of sizes means that no set can be processed more than $\lceil \log_p n \rceil$ times, and hence that there are at most $m \log_p n$ read and write operations while the algorithm executes, and, summed across the geometric sequence of set sizes, at most $Mp/(p-1)$ (that is, $O(M)$) words of data written. As many as $\log_p n$ files are required to be open for writing at any given instant, and one file for reading.

The final column of Table 4 shows the performance of an in-memory version of the DF-GREEDY approach. It is implemented using the same data structures as shown in Table 2, using a $pqueue$ with $\log_p n + 2m + O(1)$ words required, with set size uc and a forward pointer required as part of each element. For the file WEBDOCS, for example, and with $p = 1.1$, there is a net space saving of around 14 MB compared to LAZY-GREEDY.

Needless to say, the DF-GREEDY approach is also fast when implemented this way, since it has a very similar execution profile to the LAZY-GREEDY mechanism. Indeed, the only significant difference between them is that in the LAZY-GREEDY approach, each bucket in $pqueue$ represents a single set size, whereas in the in-memory DF-GREEDY implementation, each bucket represents a range of set sizes spanning (when $p = 1.1$) a 10% margin. That is, another way of categorizing the LAZY-GREEDY approach is that it is the limiting implementation, as $p \rightarrow 1$, of the in-memory DF-GREEDY mechanism.

The size of the solutions is very slightly higher when the in-memory DF-GREEDY is employed, but they are still close to the range established by the two “exact” approximate implementations. (Of course, the size of “true” solutions to these problem instances is not known.)

Interestingly, on the extreme problem instances described in Section 3.3, DF-GREEDY performs very well. For instance, with $m = 400$, DF-GREEDY with $p = 1.1$ requires just 0.35 seconds to return the solution, checking whether 70,391,613 items are covered. This is ten times faster than EAGER-GREEDY and over forty times

faster than LAZY-GREEDY (Table 3). The point is that DF-GREEDY is not concerned when the set in focus has had a small fraction of its elements already covered, and in many cases adds the set to the solution anyway, bypassing the cascading cycle of cleansings that hinder the other two implementations.

Cormode et al. [4] compare in-memory and on-disk versions of their DF-GREEDY method with a range of other implementations, including an EAGER-GREEDY implementation. Using a 2.8 Ghz MacOS machine with 8 GB of main memory, their in-memory DF-GREEDY requires 93 seconds to process WEBDOCS with $p = 1.001$, and around 70 seconds with $p = 1.1$, compared to an EAGER-GREEDY execution time of 199 seconds. One possible explanation for the variation in execution time between their implementation of DF-GREEDY and our implementation of DF-GREEDY (shown as requiring 5 seconds in Table 4) is the use of different data structures; another contributing factor may be different experimental assumptions (for example, we do not include data reading times in Table 4).

5 Partial Set Cleansing

One further avenue was explored, still with the objective of reducing the execution time. Table 4 shows that the LAZY-GREEDY approach requires a minimum of three times more check-covered operations than does EAGER-GREEDY. Much of that effort is spent removing covered items from dirty sets even after it is known that set will not be the next one added to the solution. Removal of covered elements is, of itself, not wasted effort. But checking of the uncovered elements embedded in the same set is in some sense unhelpful, since those uncovered elements will all get checked again the next time this set emerges from the priority queue.

The idea of partial set cleansing is to interrupt the processing of checking a set once some threshold of “dirtiness” is exceeded, knowing that it will be some time before this set emerges again from the queue, and knowing that in the intervening period, more of the elements might have become covered.

Two different strategies were explored. The first was to abandon cleansing set S_i as soon as enough covered items had been identified in it to confirm that it could not be the next one added into the solution. That is, if set S_i was from $pqueue[uc]$, and the next largest set was in (say) $pqueue[uc']$, then as soon as $uc - uc' + 1$ covered items have been found in S_i , it is returned to the queue in bucket $pqueue[uc' - 1]$, and attention switched to the set or sets in bucket $pqueue[uc']$.

To avoid repeatedly testing and retesting elements from the start of each set, a variable *restart_point* is added to each node in the queue, to record the position from which check-covered operations should resume, when and if this set returns to the front of the queue again. Resuming the next loop of check-covered operations from *restart_point* rather than the beginning of the set increases the likelihood of identifying covered items; but does not remove the need for every item in the set to be checked before it can be added to the solution.

The second strategy was to continue checking set elements until some fixed fraction of the set was found to be covered, and reinsert a partially cleansed set into the queue as soon as that condition was met. When set cleansing is resumed, it is again from a stored *restart_point*.

Operations	LAZY-GREEDY	LAZY-GREEDY with partial set cleansing		
		$uc - uc' + 1$	$1 + 0.10 \times uc$	$1 + 0.50 \times uc$
Extract-maximum operations	3,291,585	287,420,896	44,731,342	10,276,636
Check-covered operations	335,085,502	314,325,433	319,843,092	326,240,303
Check-covered per extract-max.	101.80	1.09	7.15	31.75
Data movements	—	19,706,057	25,223,716	31,620,927
Running time (s)	5.29	189.71	37.40	12.37

Table 5: Operation counts for LAZY-GREEDY with full set cleansing, and three levels of partial set cleansing. The threshold of the partial cleansing is the maximum number of covered elements to be discovered prior to a suspension of the set difference. The critical factor determining execution time is the number of check-covered operations required.

Table 5 shows the costs associated with the revised process. Each approach to partial cleansing is able to slightly decrease the number of check-covered operations, but not by enough to also decrease the running time. Indeed, the number of extract-maximum operations grows significantly, adding considerably to the execution cost. It may be that other variants of this idea can be identified that achieve the hoped-for blend of attributes, but to date we have not identified a method that is significantly faster than the LAZY-GREEDY and DF-GREEDY implementations.

6 Conclusion and Future Work

We have explored the GREEDY approach to the SET COVER problem, describing and measuring the performance of a new implementation, the LAZY-GREEDY approach. Compared to the standard EAGER-GREEDY mechanism, the LAZY-GREEDY implementation requires around half the memory space, and as little as 5% of the execution time when applied to typical large problem instances. The trade-off is that the lazy set update process increases the worst-case running time to $\Omega(M^{4/3})$, compared to $O(M)$ for EAGER-GREEDY, where M is the total size of the input.

The LAZY-GREEDY implementation can be viewed as a special-case in-memory version of the DF-GREEDY method of Cormode et al. [4]. An interesting question is whether that relationship can be exploited in some way, perhaps to combine the speed of the DF-GREEDY method with the solution performance bound enjoyed by the LAZY-GREEDY approach in a single sequentially-processing implementation that is oblivious as to whether its data is resident in memory or on disk, and also retain the worst-case usefulness of DF-GREEDY.

We are also intrigued by the observation, arising from our experiments, that tie-breaking in the GREEDY mechanism can lead to solutions of differing quality. As a further part of our ongoing study, we plan to explore randomized choice evaluation to determine if hill-descending approaches might be effective in practice. The idea here would be to specify a total execution time that may be spent, and then execute GREEDY as many times as possible within that time, in order to obtain in aggregate a better solution than is likely to arise from a single execution.

Acknowledgments

Graham Cormode provided valuable input. This work was supported by the Australian Research Council.

References

- [1] G. Ausiello, N. Bourgeois, T. Giannakos, and V. T. Paschos. Greedy algorithms for on-line set-covering. *Algorithmic Operations Research*, 4(1):36–48, 2009. URL <http://journals.hil.unb.ca/index.php/AOR/article/view/5928>.
- [2] G. E. Blelloch, R. Peng, and K. Tangwongsan. Linear-work greedy parallel approximate set cover and variants. In *Proc. 23rd ACM Symp. Parallelism in Algorithms and Architectures*, pages 23–32, 2011. doi: 10.1145/1989493.1989497.
- [3] G. E. Blelloch, H. V. Simhadri, and K. Tangwongsan. Parallel and I/O efficient set covering algorithms. In *Proc. 24th ACM Symp. Parallelism in Algorithms and Architectures*, pages 82–90, 2012. doi: 10.1145/2312005.2312024.
- [4] G. Cormode, H. Karloff, and A. Wirth. Set cover algorithms for very large datasets. In *Proc. 19th ACM Int. Conf. Information and Knowledge Management*, pages 479–488, 2010. doi: 10.1145/1871437.1871501.
- [5] H. S. Dutta. Survey of approximation algorithms for set cover problem. Master’s thesis, University of North Texas, 2009. URL http://digital.library.unt.edu/ark:/67531/metadc12118/m1/1/high_res_d/thesis.pdf.
- [6] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998. doi: 10.1145/285055.285059.
- [7] F. Gomes, C. Meneses, P. Pardalos, and G. Viana. Experimental analysis of approximation algorithms for the vertex cover and set covering problems. *Computers & Operations Research*, 33(12): 3520–3534, 2006. doi: 10.1016/j.cor.2005.03.030.
- [8] T. Grossman and A. Wool. Computational experience with approximation algorithms for the set covering problem. *Eur. J. of Operational Research*, 101(1):81–92, 1997. doi: 10.1016/S0377-2217(96)00161-0.
- [9] R. Hassin and A. Levin. A better-than-greedy approximation algorithm for the minimum set cover problem. *SIAM J. Computing*, 35(1):189–200, 2005. doi: 10.1137/S0097539704444750.
- [10] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Computer and System Sciences*, 9:256–278, 1974. doi: 10.1016/S0022-0000(74)80044-9.
- [11] R. V. Kantety, M. La Rota, D. E. Matthews, and M. E. Sorrells. Data mining for simple sequence repeats in expressed sequence tags from barley, maize, rice, sorghum and wheat. *Plant Molecular Biology*, 48(5-6):501–510, 2002. doi: 10.1023/A:1014875206165.
- [12] R. M. Karp. Reducibility among combinatorial problems. In *Fifty Years of Integer Programming, 1958-2008*, pages 219–241. Springer, 2010. doi: 10.1007/978-3-540-68279-0.8.
- [13] P. Slavík. A tight analysis of the greedy algorithm for set cover. *J. Algorithms*, 25(2):237–254, 1997. doi: 10.1006/jagm.1997.0887.
- [14] N. E. Young. Greedy set-cover algorithms. In *Encyclopedia of Algorithms (Part 7)*. Springer, 2008. doi: 10.1007/978-0-387-30162-4.175.

Towards a Vertex and Edge Label Aware Force Directed Layout Algorithm

Roman Klapaukh¹

David J Pearce¹

Stuart Marshall¹

¹ School of Engineering and Computer Science
Victoria University of Wellington,
New Zealand
Email: {roma,djp,stuart}@ecs.vuw.ac.nz

Abstract

Many automatic graph layout algorithms can cause shaped vertices and edge labels (which have a size when drawn on the screen) to overlap in the resulting visualisation. Overlaps can hide information that users expect to see in cases where the graph is small. We perform two experiments on a large real-world set of small (10-110 vertex) graphs to compare how different combinations of forces in Eades' force directed layout algorithm affect the final graph layout. We identify an optimal combination of forces from those we tested. In particular, we found that adding charged walls, variable node charge and edge label charges, minimises overlaps. We also found that using Hooke's Law over Eades' logarithmic attractive force tends to reduce edge crossings.

1 Introduction

Many different kinds of data can be represented visually by graphs. Automatic graph layout allows for the visualisation of systems ranging from social networks to mind maps and flow charts. In these cases and many other real world applications the vertices and edge labels of the graph will be *shaped*; they will have some shape and size when drawn rather than being a single point. Shapedness comes from the context of the graph. In cases where the user wishes to render large numbers of vertices on the screen, all details of individual vertices and edges is lost due to the lack of screen resolution. In those cases it is the overall pattern and structure of the graph that is interesting. In contrast, when only a small number of vertices are rendered, each can take up a reasonable amount of space on the screen. Each vertex can contain information which a user may wish to see. Overlaps, where vertices or edge labels occlude each other, limit the amount of information that can be conveyed to the user. While a lot of research focuses on the visualisation of large graphs, we believe that there are still problems in the visualisation of small labelled graphs as shown by the existence of small graph benchmarks in venues such as Graph Drawing [12], and the use of small graph layout algorithms as part of large graph layout [24, 1].

Automatic graph layout is meant to position all the objects creating a "good" visualisation. What it means to be "good" is discussed in Section 2, but in

this work we are looking particularly at preventing overlaps between any combination of shaped vertices and edge labels. We consider both the count of how many overlaps there are and the proportion of pixels that are hidden. As a sanity check, we also record other properties of the layout and use the widely known and used heuristic of number of edge crossings [26].

Consider the example small shaped graph laid out in two different ways shown in Figure 1. This shows an extract from a social network where vertices contain the name of the person they represent, and edges have the type of relationship between people. In the top layout two of the vertices are overlapping, while in the bottom they are all drawn separately. In the top layout, it is hard to determine the names of the two overlapping vertices, and to determine which edge is connected to which. In the bottom layout there is no such confusion.

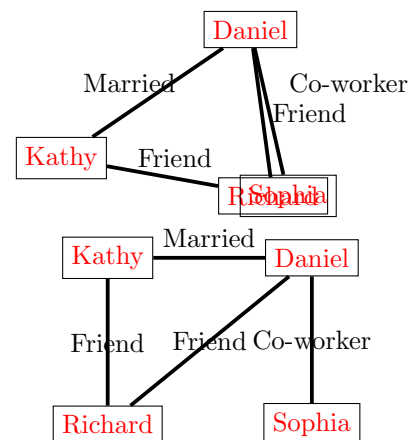


Figure 1: A good and a bad layout of a social network. In the above layout two of the vertices are overlapping and it is hard to read what they say or what edges connect to them.

While the situation of overlapping vertices can be avoided, most of the core layout algorithms assume that vertices and edge labels are not shaped – an assumption which is not upheld in many practical applications with small graphs. Furthermore, previous suggestions about how to resolve this issue have not been evaluated on a large scale (e.g. [14, 5, 4, 17, 9, 18]).

This paper contributes an evaluation of a range of variants to the standard force directed layout algorithm based on Eades' Spring Embedder [8] to prevent overlaps from occurring given shaped vertices and shaped edge labels. We experimentally evaluate a set of variants across a large real-world data set

using a suite of metrics.

We carried out a large scale experiment across 13,720 realistic graphs to evaluate how the numbers of overlaps and edge crossings changes as you use different combinations of forces. We followed this experiment with a second smaller experiment validating the results by ensuring that every combination was covered. The graphs ranged in size from 10 to 110 vertices and come from a graph drawing benchmark containing real world data from a major US corporation[12].

We found that:

1. Using charged walls, degree-based charge and charged edge labels reduce overlaps by an order of magnitude over the base algorithm.
2. Using Hooke's Law instead of the standard logarithmic attractive force tends to give layouts with fewer edge crossings.

Overall we found that the force directed layout algorithm without modifications performs significantly worse than the best modified version with respect to both overlaps and edge crossings. The ideas presented here can be used in the layout of large graphs when using multi-level layout algorithms [1], and can be used for techniques such as multi-dimensional scaling [24].

2 Background

The task of graph layout is to assign a position to each vertex in a graph. This set of positions is called a layout, and should be *good*. While an exact definition of *good* is largely subjective, metrics exist for approximating how good a graph layout is [25, 26]. These metrics quantify different properties of the graph, such as number of edge crossings, angle of separation between edges connected to the same vertex, or enforcing certain rules such as keeping edges as straight lines, or promoting symmetry. Some metrics, such as maximising the similarity of edge lengths, involve solving NP-Complete problems [8, 19]. As there are many metrics, we choose two on which to focus. The first is the number of overlaps, as that is the primary purpose of this paper. The second is the number of edge crossings. This metric is widely known, is used by people performing manual graph layout [26], and serves as a sanity check on how our changes affect other properties of the graph.

2.1 Force Directed Layout

Our work looks at extending the force directed layout algorithm, based on Eades' Spring Embedder [8], with the edges of the screen (or layout area) as an immovable barrier as per Fruchterman and Reingold [10]. There are other graph layout algorithms, such as: the Kamada and Kawai method which also deals with spring systems but solves them using Newton-Raphson on derivatives [20]; likewise spectral layout which works by finding eigenvectors [13]; finally even a method where users manually layout subgraphs [30]. More algorithms can be found in [6]. We chose to use force directed layout for this work because there are many real world graph layout systems that implement it (e.g. [16, 15, 2]); it is iterative, allowing users to interact with it while it is running; it can be used with large graphs [24, 1, 29]; and because it is claimed to promote symmetry where possible [8]. However, the ideas from this work could be extended to other layout algorithms.

The force directed layout algorithm simulates the graph as a physical system (Figure 2). The basic code can be found in Algorithm 1. Edges, like springs, pull vertices together. Vertices, like charged particles, repel each other. The system uses friction to prevent dynamic equilibrium so it tends towards a fixed state. The result of the computation is then used as a visualisation of the graph. The user can interact with the graph while the algorithm runs.

The algorithm terminates either when a certain number of iterations have finished or when the kinetic energy is low enough. The kinetic energy is a measure of the activity of the system. Given the mass (m) and the velocity (v) you have that $\text{kineticEnergy} = \frac{1}{2}mv^2$. Low values for the kinetic energy suggest that very little movement is happening so the algorithm can now be stopped.

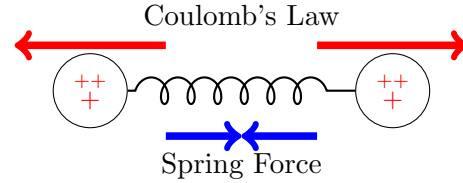


Figure 2: Basic Forces

Various forces can be used to model attraction and repulsion. Almost all the papers we surveyed used electrostatic repulsion (Coulomb's Law) as the repulsive force between vertices [8, 10, 6, 21]. In contrast, two different forces are used for the attractive force. Some papers, including Eades', use a logarithmic attractive force [8, 21, 23]. However, Eades originally describes edges as springs, so Hooke's Law (the physical law for ideal springs) is a natural alternative. It is not clear how they compare, though Battista et al. claim that from their experiences the logarithmic variant does not provide sufficiently better results given the extra computation [6]. We experimentally compare both attractive forces to see if there is a difference.

Algorithm 1 General force directed layout algorithm

```

1: generateInitialLayout()
2: for 0 .. maxIterations do
3:   totalEnergy = 0
4:   for all Vertex v do
5:     tempForce = (0,0)
6:     for all Edge {v,w} do
7:       tempForce += springForce(v,w)
8:     end for
9:     for all Vertex w do
10:      if v ≠ w then
11:        tempForce += coulombsLaw(v,w)
12:      end if
13:    end for
14:    v.move(tempForce)
15:    totalEnergy += v.kineticEnergy()
16:  end for
17:  if totalEnergy ≤ energyCutOff then
18:    break
19:  end if
20: end for

```

2.2 Existing Modifications

The original definitions used by Eades do not consider vertices as having any size, or edges as being

labelled [8]. In most graph applications, vertices have a dimension to consider, as they are represented on the screen, and edges may be labelled. For the sake of simplicity, we will refer to everything that is drawn except edges as images (e.g., text, raster images, etc). Having images leads to what we call *occluded* pixels, and *overlaps*. An occluded pixel is a pixel of an image which is covered by a pixel from a different image. An overlap occurs when two images are drawn so that one occludes some pixels of the other. This results in parts of the graph not being visible, reducing clarity and information retrieval from the visualisation, and so should be minimised. Our work focuses on minimising occlusions and overlaps.

Some previous works on reducing overlaps are modifications to the layout algorithm. Wang and Miyamoto implemented modifications that cancel out attraction of occluded vertices, vary constants to account for vertex size, and integrate a constraint solver [28]. They did not do an experimental analysis, except to time the layout of a single graph, and to generate six figures for the paper. Harel and Koren claim that naïve extensions to layout algorithms to deal with shaped vertices often have negative repercussions. They proposed changes to the Kamada and Kawai method and modifications to the spring method [14] that they claim do not have these limitations, but tested their ideas on only 7 graphs. Kumar et al. reduce clutter by giving certain vertices a stronger repulsive force in directed acyclic graphs [21], but test their algorithm on just two graphs. We extend this approach to create our degree-based charge force (discussed in Section 3.1). Lin et al. add torque to allow vertices to pack better [23], but this allows vertex images to end up at arbitrary angles, potentially decreasing readability. However, their analysis only contained 6 graphs.

Other works apply a post-processing step to ‘fix’ a layout. Force Transfer [18] and Force Scan [22] are two common algorithms to iteratively move vertices apart until they no longer overlap. Each of their experimental evaluations involved looking at only seven graphs. Frishman and Tal propose an algorithm to unclutter an existing layout [9], by mapping from the existing layout to one with a better information density. This algorithm is designed for huge graphs, where details on individual vertices are not visible, but was tested on just 5 graphs. Gansner and North use Voronoi diagrams to move vertex centres away from other each other [11], and notably introduce curved edges. Their experiment consisted of only nine graphs. Dwyer et al. use constraint solving to spread the vertices [7]. They tested their performance on some randomly generated graphs, but only tested the layout quality on a single graph.

None of the works above performed any large scale testing of their algorithms, with the largest test set containing only 12 graphs. This makes it hard to generalise their results. For this reason, we perform a large scale evaluation on over 13,720 realistic graphs, many of which are anonymised graphs from AT&T. Additionally, of all the algorithms we surveyed, only the ePRISM [17] algorithm explicitly considers edge labels, despite their common use in practice.

3 Algorithm Design and Variants

We use the general force directed algorithm as described in Section 2 as the basic algorithm. We describe some of the implementation details below. The system was implemented using Java 1.6.

Initial placement places each vertex at random. The natural length for the attraction force is set as the minimum distance such that the two connected vertices do not overlap [22, 1].

Vertex - vertex repulsion is done with Coulomb’s Law and has the form $\mathbf{F} = -k_e \frac{q_1 q_2}{\|\mathbf{r}_{21}\|^2} \hat{\mathbf{r}}_{21}$, where k_e is a constant, q_i is the charge on the given vertex, r_{21} is the distance between the two vertices, and $\hat{\mathbf{r}}_{21}$ is the unit vector between the two vertices. All vertices have the default charge of q_{vertex} .

To move a vertex we convert the total force on it to acceleration using Newton’s Law ($\mathbf{a} = \frac{\mathbf{F}}{m}$). All vertices have the same mass (m). Each time a vertex is moved dampening reduces the velocity by a fixed proportion, to ensure that the system eventually settles to a static layout. The plane boundaries in our system are an impenetrable barrier representing the edges of the screen. Any vertex that hits a wall has its component of velocity in the direction of the collision reversed.

3.1 Variable Forces

We will now describe the different forces we experimentally investigated. Recall the basic configuration of forces from Section 2.

Hooke’s Law (H) Hooke’s Law is the physical law for ideal springs, and so is a logical candidate for the spring force to model the edges. Its advantage is that it is less computationally expensive than the logarithmic spring force [6]. It has the form $\mathbf{F} = -k_h (\mathbf{x} - \mathbf{N})$, where \mathbf{x} is the vector between the two vertices, \mathbf{N} is the natural length, and k_h is a constant describing how stiff the spring is.

Logarithmic Spring Force (L) This is the spring force Eades’ original paper used, and is also used in other work [8, 6, 22]. It has the form $\mathbf{F} = k_l \log\left(\frac{x}{N}\right)$ where k_l is a constant, x is the distance between the two vertices, and N is the natural length. This gives it a behaviour that is similar in shape to Coulomb’s Law, making it a logical, if more computationally expensive, alternative to Hooke’s Law.

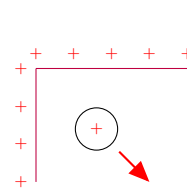


Figure 3: Charged Walls (W)

Charged Walls (W)

Davidson and Harel proposed (but did not implement) this as a mechanism for keeping vertices inside fixed boundaries [5]. Wall charge is just Coulomb’s Law applied to a line the length of the boundary as in Figure 3. This serves several purposes:

- it prevents unconnected components from moving infinitely far from each other;
- it prevents layouts settling close to the boundary where their ability to move is limited, impairing their ability to move into a minimal energy configuration;
- and it centers the resulting image.

Wall charge in Table 2 shows the charge of each wall as used in our experiments.

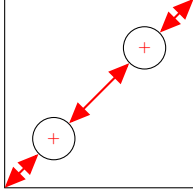


Figure 4: Wrap-Around Forces (A)

directions. Given any two vertices, this force pushes them to be equal distances from each other both ways around the plane, providing a similar effect to charged walls, where the vertices take the place of the charge on the wall. We hypothesize that combining wrap-around forces with charged walls will center all the graphs neatly. This is a novel force which we are testing.

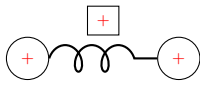


Figure 5: Charged Edge Labels (E)

Figure 5. This force is based on the obvious extension to force directed layout where edge labels are treated as special nodes. Each label has charge q_{label} , and so repels vertices to which it is not connected. Labels are unable to move independently so forces applied to a label instead affect the two vertices which it is connected to. This avoids moving the label independently, while allowing labels to affect the layout.

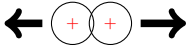


Figure 6: Collisions (C)

restitution. Collisions are only implemented for vertices.

Our implementation computes the velocity changes due to the collision and sets the new velocities appropriately as in Figure 6. While it is possible to move the vertices apart until they are no longer overlapping, we do not do this because there may not always be a sensible layout possible where there are no overlaps. Some graphs may have very dense regions, or just so many vertices that allowing some amount of overlaps is actually beneficial.

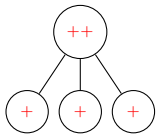


Figure 7: Degree-Based Charge (D)

charge. The larger the degree of a vertex the more

Wrap-Around Forces (A) Wrap-around forces cause the repulsive force to act as if the layout is on a torus as seen in Figure 4. This does not change how walls affect the vertices, but Coulomb's Law calculations are performed in both

Charged Edge Labels (E) Edge labels can be occluded by other images. To prevent that, charged edge labels makes labels charged in the same way that vertices already are as in

Collisions (C) Collisions between vertices is a trivial extension to force directed layout as it is based on physical simulation. The collisions are ideal billiard ball collisions using the coefficient of

Degree-Based Charge (D) Kumar et al. proposed to increase the charge on certain vertices to give them more space [21]. Their technique relies on the graph being a directed acyclic graph. We generalise their technique to general graphs, giving high degree vertices higher

space it needs, therefore the more it should repel other images as in Figure 7. Thus we multiply the standard Coulomb repulsion between the vertices by the max of 1 and $\frac{\text{degree}(v_1) * \text{degree}(v_2)}{4}$. We use the product of degrees to increase the strength of the repulsion quickly, as the repulsive force falls off quickly with respect to distance. The original formula cannot be used directly as it requires directed acyclic graphs. The constant 4 is chosen for the denominator so that low degree vertices are not affected (supposing an average low degree vertex has degree 2).

4 Experiment

We ran two experiments comparing different combinations of forces to see how they affect the number overlaps, and edge crossings. To measure overlaps, we recorded both the number of overlapping vertices and the number of occluded pixels. We also recorded the time taken to lay out each graph, to see how the modifications affect performance. We use Coulomb's Law as the repulsive force in all the experiments as it is used in Eades' original work [8], and makes intuitive sense as the relevant physical law. In the first experiment we ran each of the fourteen combinations shown in Table 1 once on each of the 13,720 graphs in the dataset. In the second we ran all combinations of forces on a random sample of 100 graphs, to show that we did not miss any potentially good force combinations and to get a more complete picture of how forces affect computational performance.

Each layout was run until either the kinetic energy (a measure of the activity of the system) dropped below a given force cut off, or the maximum number of iterations was reached. We use short names comprised of a letter for each active force (excluding Coulomb's Law) in this paper for compactness.

Hypothesis We set out to investigate the following hypotheses.

\mathcal{H}_1 Changing between H and L has no effect on layout.

\mathcal{H}_2 Adding in all the additional forces except for C (i.e. WEDA) results in the lowest number of overlaps.

In order to test the various algorithms we used three test data sets from GraphDrawing.org [12]. The sets are called *Rome*, *North*, and *random-dag*. We use these graphs as they mostly contain graphs from real world applications and are small - containing 10 to 110 vertices, and 9 to 241 edges. There are a total of 13,720 graphs, of which all are used.

Choice of Constants We selected the image size based on a personal social network context such that they are small, but still easily recognisable. The width and height of the plane for embedding was limited to what most modern screens support. The constants were all set by trial and error, such that they looked reasonable on several data sets. Choice of constants is a difficult exercise, as small changes can have unforeseen effects. Many papers have simply ignored the issue entirely e.g. [6, 8, 22, 3, 11, 22, 21, 4]. A standard practice is to use a heuristic method similar to ours, such as optimising for a simple case e.g. [10, 28, 20]. While such methods are not ideal, addressing the core issue of how to choose optimal constant values is beyond the scope of this paper.

In the second experiment, the width and height of the screen were fixed to 1920 x 1080 to mimic a normal screen, and the image size was changed to be square rather than rectangular like a screen. We made these changes to see if small changes to the setup would affect the relative performance of the best combination of modifications. The full set of constants for both experiments is shown in Table 2.

5 Results

We used R version 2.11.1 (2010-05-31) [27] to analyse the results of both experiments. We recorded the number of pixels drawn, the total number of pixels, the number of overlapping images, and the number of edge crossings. The number of pixels drawn compared to the total tells us how many pixels were occluded and therefore how much is hidden. The number of overlaps gives an indication of how crowded the layout is. While a layout can have no overlaps and still be crowded, we only look at crowding that results in occluded pixels. Edge crossings were recorded to see if other features of the layout were affected by our changes, and as minimising them is considered to increase goodness [25, 26].

As the resulting data does not seem to follow a normal distribution, we used the Wilcoxon rank-sum test for significance testing as opposed to the t-test. We hold that the difference is significant at 95% significance ($p < 0.05$).

5.1 Experiment One

The list of medians for each force in the first experiment is shown in Table 3. The entries in bold blue are minima. All reported values have been rounded to 4 significant figures, with trailing 0s omitted.

The medians for the proportion of possible overlaps in the graph (of any size) and the raw count of overlaps can be found in Table 3. In both cases H and L ($p = 0.1176, 0.3186$ resp.) are not significantly different and the best performer is LWED. In terms of area lost due to occlusion the best performer is LWED. With respect to the proportion of edge crossings compared to an estimated upper bound (calculation from [25]), and also as a raw count HWED is the best performer.

5.2 Experiment Two

The list of medians for each force in the second experiment is shown in Table 3. The entries in bold blue are minima. All reported values have been rounded to two decimal places.

LD was the best performing combinations with respect to edge crossings. LWED was best in percentage of overlaps and occluded pixels, and best equal (with HWED) in count of overlaps. In 35 (72%) of cases the same combinations of forces, with H rather than L had less edge crossings.

6 Discussion

The LWED algorithm was the most effective at reducing overlaps in the graph, as well as having the lowest number of hidden pixels. While both results tables shows that in terms of raw counts HWED and LWED are the same, this is only because we are showing medians in the table. If we considered arithmetic means then LWEDs would be lower. This would make it the best algorithm for our original purpose. It also fared well with respect to edge crossings in the first

experiment, coming second equal. HWED had the least edge crossings and came second with regards to minimising occluded pixels. An example graph generated using HWED can be seen in Figure 9.

This disproves our second hypothesis that HWED or LWED would be the best force, but does show that adding in extra forces improves performance with respect to the metrics used.

All tests run with wrap-around forces (labelled A) fared poorly. This was a surprising result, as with two vertices this causes vertices to spread out. Upon running some additional simulations with larger graphs, we found that while it did push all the vertices together, it did so too much leading to poor performance. It seemed to perform particularly poorly on sparse non-planar graphs.

The results from the second experiment confirm that the LWED and HWED still perform the best. The changes to the parameters only affected the ordering of force combinations that did not perform well.

They also show that keeping everything else constant, changing from using Hooke's Law (H) to the logarithmic spring force (L) generally increases the number of edge crossings. While we are not sure what causes this, we think it has to do with how the different spring forces change over a small distance while a vertex is trying to move over an edge to make a crossing.

We were interested in how different forces affected performance. We recorded the runtime of each layout in the second experiment and used that to find an average run time for a single iteration of the algorithm for each.

There are three forces which visibly affect program runtime - E, A and C. Figure 8 shows the average time taken for a single iteration of the loop coloured by which combination of these forces is active. The figure shows that the charged edge labels (E) force incurs a clearly noticeable time increase. This is due to this modification requiring an extra inner loop running over all the edges. The presence of both C and A increases the run time more than having both independently would suggest. We believe this happens because the wrap-around forces (A) force promotes collisions, creating more work for the collisions (C) force.

Validity The most notable omission of the experiments is that we do not explore how different combinations of constants affect the output of the program. This is something that has not been explored in other papers. It also has a very large search space, and would take infeasibly long to run.

While we used a large test set of graphs, they do not encompass all possible graphs. They span a variety of graph densities, from 0.8608% to 85.45%, though the majority are less than 8.602%, but all except 3 are connected. Nevertheless, we believe that the set of graphs tested here is representative of many small real life graphs as it is sampled from a real data set. We hypothesise that changes and variances in image size can be accounted for by linked changes to physical constants for each vertex. This is supported by the second experiment where images were smaller, but the best performing forces were the same.

Future Work One difficulty in evaluating algorithms is the selection of constants, and what effect this has on the resulting layout. Further work evaluating the effects of constants would be valuable for

Table 1: Sets of forces we tested in experiment one

Experiment	1	2	3	4	5	6	7	8	9	10	11	12	13	14
H (Hooke's Law)	✓	✓	✓	✓	✓	✓	✓							
L (Logarithmic Attraction)								✓	✓	✓	✓	✓	✓	✓
W (Charged Walls)		✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	
E (Charged Edge Labels)		✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
C (Collisions)			✓							✓				
D (Degree Based Charge)				✓		✓	✓				✓		✓	✓
A (Wrap-Around Forces)					✓	✓	✓					✓	✓	✓

Table 2: Constants used in both experiments

Constant	Value	Constant	Value
All Experiments			
k_e	50,000	Coefficient of Restitution	0.9
k_l	-60	Kinetic Energy Cut Off	3
k_h	0.2	Edge Label Length	2-4 characters
q_{label}	1	Natural Spring Length	Min dist to not overlap
q_{vertex}	3	Dampening	0.9
Wall Charge	1000	Max Iterations	10,000
Vertex Mass	2		
Experiment 1			
w, h	$400\text{px} \leq \# \text{Vertices} \times 100 \leq 8000\text{px}$		
Vertex Image Size	107x87 pixels		
Experiment 2			
$w \times h$	1920 x 1080	Vertex Image Size	80x80 pixels

Table 3: Medians by force from experiment one

Median	# Crossings	Prop. Crossings	# Overlaps	% Overlaps	% Occluded Pixels
H	40	0.02256	13	0.2043	1.213
HWE	38	0.02265	3	0.0407	0.1072
HWEC	39	0.0226	3	0.04024	0.104
HWED	34	0.01984	1	0.01463	0.02526
HWEA	80	0.05337	85	1.948	16.46
HWEDA	73	0.04748	53	1.276	12.21
HEDA	71	0.04523	49	1.144	11.34
L	41	0.02369	13	0.2016	1.195
LWE	42	0.02445	2	0.03487	0.08999
LWEC	41	0.02459	2	0.03518	0.09045
LWED	38	0.02231	1	0.01058	0.0159
LWEA	92	0.0615	103	2.266	18.4
LWEDA	91	0.06051	69	1.744	15.34
LEDA	89	0.05751	63	1.583	14.3

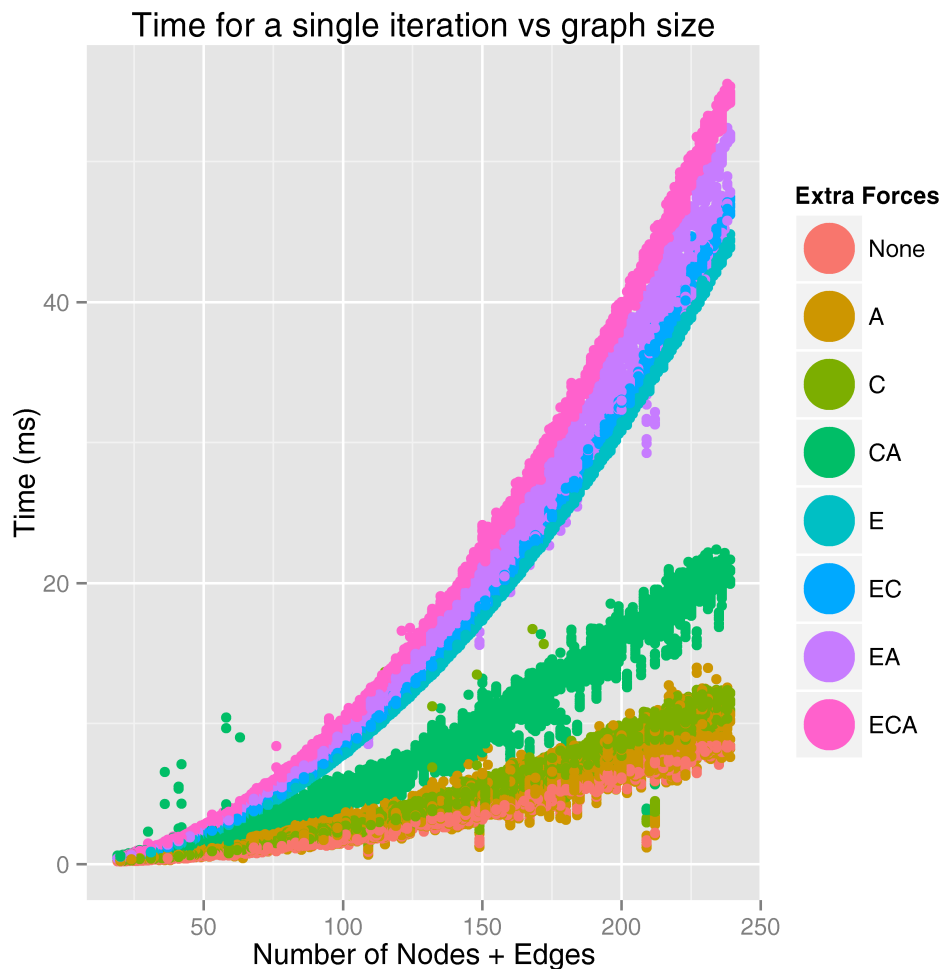


Figure 8: Time for a iteration vs number of vertices and edges in the graph coloured by forces which reduce performance. Times are an average over the whole runtime of the program with all combinations of forces. There are three well defined strata. The top one consists of combinations which contain E. Within this there are further divisions where C or A are also present. The second strata consists solely of combinations containing CA in the force. The final strata shows that A and C slow down the algorithm, and having none of E,C or A is the fastest.

everyone using any graph layout algorithm. Moreover, exploratory studies suggest that size of the layout plane affects the resulting layout significantly and we hope to explore this in future.

7 Conclusion

We looked at modifications to the force directed algorithm to avoid overlaps in small graph layout. We described a range of different forces which can be used with layout algorithms. We then performed two experiments to find which set of forces would produce a layout that was spread out applied to the force directed layout algorithm. We found adding in edge label charges, charged boundaries, and increasing vertex charge proportionally to its degree results in layouts that minimise the number of overlaps and occluded pixels; that using Hooke's law reduces the number of edge crossing; and that while there is a time cost for using charged edge labels, it is not so high as to make the modification too expensive. As such, we conclude that adding in additional forces is a viable way of preventing occluded pixels for graphs with large vertices and edge labels.

References

- [1] Archambault, D., Munzner, T. and Auber, D. [2007], 'TopoLayout: Multilevel graph layout by topological features', *IEEE TVCG* **13**(2), 305–317.
URL: <http://dx.doi.org/10.1109/TVCG.2007.46>
- [2] AT&T [n.d.], 'Graphviz'. www.graphviz.org/ accessed: 20 Nov 2012.
- [3] Brandes, U., Käab, V., Löh, A., Wagner, D. and Willhalm, T. [2000], 'Dynamic WWW structures in 3D', *JGAA* **4**(3), 183–191.
- [4] Chuang, J.-H., Lin, C.-C. and Yen, H.-C. [2004], Drawing graphs with nonuniform nodes using potential fields, in 'Proc. GD, LNCS', pp. 460–465.
- [5] Davidson, R. and Harel, D. [1996], 'Drawing graphs nicely using simulated annealing', *ACM TOG* **15**(4), 301–331.
- [6] Di Battista, G., Eades, P., Tamassia, R. and Tollis, I. G. [1999], *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall.

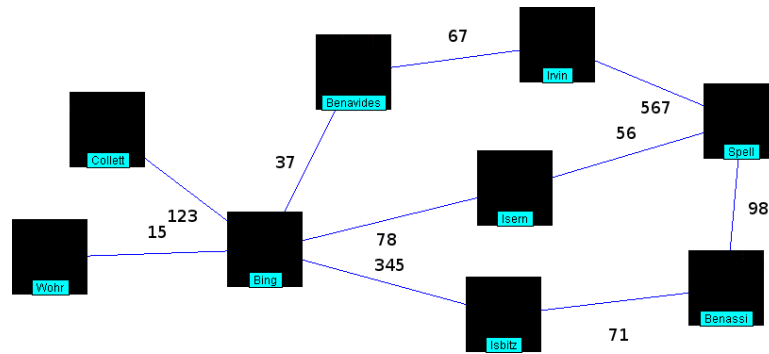


Figure 9: A graph produced by HWED. There are no overlaps or edge crossings allowing a human to see details on the vertices as well as the graph structure

- [7] Dwyer, T., Marriott, K. and Stuckey, P. J. [2005], Fast node overlap removal, in P. Healy and N. S. Nikolov, eds, 'GD', Vol. 3843 of *LNCS*, pp. 153–164.
- [8] Eades, P. [1984], 'A heuristic for graph drawing', *Congressus Numeratum* **42**, 149–160.
- [9] Frishman, Y. and Tal, A. [2009], 'Uncluttering graph layouts using anisotropic diffusion and mass transport', *IEEE TVCG* **15**(5), 777–788.
- [10] Fruchterman, T. M. J. and Reingold, E. M. [1991], 'Graph drawing by force-directed placement', *Software - Practice and Experience* **21**(11), 1129–1164.
- [11] Gansner, E. R. and North, S. C. [1998], Improved force-directed layouts, in S. Whitesides, ed., 'GD', Vol. 1547 of *LNCS*, Springer, pp. 364–373.
- [12] *graphdrawing.org* [n.d.]. <http://graphdrawing.org/data.html> accessed: 18 May 2011.
- [13] Hall, K. M. [1970], 'An r-dimensional quadratic placement algorithm', *Management Science* **17**(3), pp. 219–229.
URL: <http://www.jstor.org/stable/2629091>
- [14] Harel, D. and Koren, Y. [2002], Drawing graphs with non-uniform vertices, in 'Proc. of the Working Conf. on AVI', AVI 2002, ACM, New York, pp. 157–166.
URL: <http://doi.acm.org/10.1145/1556262.1556288>
- [15] Heer, J., Card, S. K. and Landay, J. A. [2005], Prefuse: a toolkit for interactive information visualization, in 'Proceedings of the SIGCHI Conference on Human Factors in Computing Systems', CHI '05, ACM, New York, NY, USA, pp. 421–430.
URL: <http://doi.acm.org/10.1145/1054972.1055031>
- [16] Hotson, D. [n.d.], 'Springy.js'. <http://getspringy.com/> accessed: 20 Nov 2012.
- [17] Hu, Y. [2009], 'Visualizing graphs with node and edge labels', *CoRR abs/0911.0626*.
- [18] Huang, X., Sajeev, A. S. M. and Lai, W. [2006], A scalable algorithm for adjusting node-node overlaps, in 'CGIV', IEEE Computer Society, pp. 43–48.
- [19] Johnson, D. S. [1982], 'The NP-completeness column: An ongoing guide', *J. of Alg.* **3**(2), 182–195.
URL: <http://www.sciencedirect.com/science/article/pii/0196677482900189>
- [20] Kamada, T. and Kawai, S. [1989], 'An algorithm for drawing general undirected graphs', *Information Processing Letters* **31**(1), 7–15.
- [21] Kumar, P., Zhang, K. and Wang, Y. [2008], Visualization of clustered directed acyclic graphs without node overlapping, in 'TV', IEEE Comp. Soc., pp. 38–43.
- [22] Li, W., Eades, P. and Nikolov, N. S. [2005], Using spring algorithms to remove node overlapping, in 'APVIS', Vol. 45 of *CRPIT*, pp. 131–140.
- [23] Lin, C.-C., Yen, H.-C. and Chuang, J.-H. [2009], 'Drawing graphs with nonuniform nodes using potential fields', *JVLC* **20**(6), 385–402.
- [24] Morrison, A., Ross, G. and Chalmers, M. [2003], 'Fast multidimensional scaling through sampling, springs and interpolation', *Information Visualization* **2**(1), 68–77.
- [25] Purchase, H. C. [2002], 'Metrics for graph drawing aesthetics', *JVLC* **13**(5), 501–516.
- [26] Purchase, H. C., Pilcher, C. and Plimmer, B. [2012], 'Graph drawing aesthetics - created by users, not algorithms', *IEEE TVCG* **18**(1), 81–92.
- [27] R Development Core Team [2010], *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.
URL: <http://www.R-project.org>
- [28] Wang, X. and Miyamoto, I. [1995], Generating customized layouts, in F.-J. Brandenburg, ed., 'GD', Vol. 1027 of *LNCS*, Springer, pp. 504–515.
- [29] Wong, P. C., Foote, H., Mackey, P., Jr., G. C., Sofia, H. J. and Thomas, J. [2008], 'A dynamic multiscale magnifying tool for exploring large sparse graphs', *Information Visualization* **7**(2), 105–117.
- [30] Yuan, X., Che, L., Hu, Y. and Zhang, X. [2012], 'Intelligent graph layout using many users' input', *IEEE TVCG* **18**(12), 2699–2708.

Table 4: Medians by force from experiment two. Short names are used in the column names for compactness. C is Crossings, O is Overlaps and P is Pixels Occluded. The minimum value for each metric is highlighted in blue.

Force	# C	% C	# O	% O	% P	Force	# C	% C	# O	% O	% P
H	34.50	2.28	13.00	0.24	0.91	HA	276.00	18.20	375.50	7.61	80.38
HC	48.00	3.39	19.00	0.34	1.31	HCA	298.00	19.66	172.00	3.12	29.67
HD	31.00	2.09	10.00	0.22	0.82	HDA	294.50	20.57	411.50	8.61	83.47
HDC	32.00	2.25	10.00	0.22	0.83	HDCA	312.00	20.46	172.50	3.10	29.34
HE	38.00	2.72	6.00	0.13	1.16	HEA	95.00	6.94	16.00	0.32	5.72
HEC	41.00	2.87	8.00	0.18	2.02	HECA	55.00	4.18	23.00	0.46	5.76
HED	40.00	2.85	6.00	0.12	1.22	HEDA	176.50	12.02	128.00	2.55	45.02
HEDC	41.50	2.89	6.00	0.14	1.57	HEDCA	93.00	6.51	41.00	0.72	8.96
HG	39.00	2.74	8.00	0.19	1.37	HGA	98.00	7.09	21.00	0.41	6.15
HGC	42.00	3.09	11.00	0.23	2.04	HGCA	60.00	4.57	28.00	0.54	6.15
HGD	40.00	2.88	9.00	0.18	1.38	HGDA	171.00	11.94	100.00	2.00	38.88
HGDC	43.00	3.14	10.00	0.21	1.68	HGDCA	99.00	7.11	44.50	0.77	9.34
HW	55.00	3.36	35.00	0.63	2.42	HWA	284.00	19.04	388.50	7.99	82.60
HWC	123.00	8.57	65.50	1.07	8.99	HWCA	306.50	20.66	177.00	3.34	30.74
HWD	43.00	2.62	22.00	0.40	1.54	HWDA	312.00	21.02	434.50	8.83	84.34
HWDC	48.00	3.46	24.50	0.51	1.74	HWDCA	323.00	21.09	176.00	3.33	30.69
HWE	34.00	2.17	4.00	0.08	0.33	HWEA	146.00	9.80	34.00	0.59	10.54
HEC	39.00	2.74	9.00	0.18	1.48	HECA	83.50	6.08	44.00	0.81	10.31
HWED	34.00	2.22	2.00	0.05	0.13	HWEDA	192.00	13.20	165.00	3.37	52.25
HWEDC	38.00	2.59	5.00	0.11	0.69	HWEDCA	125.50	8.52	61.00	1.09	13.19
HWG	35.00	2.19	7.00	0.15	0.49	HWGA	143.00	9.94	40.00	0.70	10.62
HWGC	44.00	3.03	13.00	0.27	1.75	HWGCA	98.00	6.98	51.00	0.89	10.75
HWGD	35.00	2.25	6.00	0.13	0.39	HWGDA	197.00	12.89	137.00	2.71	46.65
HWGDC	40.00	2.87	9.00	0.20	0.95	HWGDCA	133.50	9.65	64.00	1.15	12.89
L	33.00	2.09	13.00	0.24	0.91	LA	266.50	18.08	369.00	7.42	79.89
LC	42.00	2.81	17.00	0.30	1.16	LCA	285.50	19.60	172.00	3.03	29.73
LD	27.00	1.87	9.00	0.20	0.78	LDA	292.50	19.94	418.50	8.40	82.26
LDC	29.00	1.98	9.00	0.20	0.75	LDCA	315.50	20.60	176.00	3.20	29.54
LE	37.00	2.65	4.00	0.10	0.86	LEA	89.00	6.60	12.00	0.26	4.89
LEC	40.00	2.80	6.00	0.12	1.20	LECA	52.00	3.91	20.00	0.41	5.21
LED	39.00	2.77	4.00	0.08	0.73	LEDA	178.00	11.47	106.00	2.06	40.06
LED	40.00	2.87	5.00	0.10	1.18	LEDCA	95.00	6.36	40.00	0.71	8.80
LG	36.00	2.72	7.00	0.15	0.99	LGA	95.00	6.74	17.00	0.36	5.89
LGC	42.00	3.03	9.00	0.18	1.54	LGCA	60.00	4.45	24.00	0.50	5.34
LGD	39.00	2.85	7.00	0.16	1.01	LGDA	170.50	11.31	81.00	1.64	34.29
LGDC	40.50	3.02	7.00	0.17	1.22	LGDCA	100.50	7.27	43.00	0.76	9.11
LW	50.00	3.07	36.00	0.63	2.44	LWA	288.00	19.23	401.50	8.04	82.48
LWC	110.00	7.68	65.00	1.04	8.33	LWCA	311.50	20.93	179.00	3.32	30.71
LWD	35.00	2.28	19.50	0.38	1.41	LWDA	315.50	20.97	442.50	8.73	84.21
LWDC	40.00	2.85	22.00	0.45	1.57	LWDCA	322.50	21.50	179.00	3.37	30.78
LWE	32.00	2.10	3.00	0.06	0.25	LWEA	141.50	9.74	32.00	0.53	9.92
LWEC	36.00	2.53	6.00	0.13	0.90	LWECA	84.00	6.03	43.00	0.78	10.14
LWED	34.00	2.16	2.00	0.04	0.09	LWEDA	185.00	12.72	145.00	2.96	49.16
LWEDC	36.00	2.54	3.00	0.07	0.36	LWEDCA	118.00	8.56	61.00	1.08	12.97
LWG	33.00	2.11	6.00	0.14	0.41	LWGA	145.50	9.92	38.00	0.64	10.65
LWGC	41.00	2.95	10.00	0.22	1.15	LWGCA	96.00	6.72	49.00	0.85	10.63
LWGD	35.00	2.23	5.00	0.12	0.37	LWGDA	195.00	12.47	112.00	2.35	43.16
LWGDC	37.00	2.73	7.00	0.16	0.58	LWGDCA	133.00	9.51	66.00	1.14	13.11

DAC: Database Application Context Analysis applied to Enterprise Applications

Johannes Wust

Carsten Meyer

Hasso Plattner

Hasso Plattner Institute for Software Systems Engineering,
University of Potsdam,
PO Box 14440, Potsdam, Germany,
Email: johannes.wust@hpi.uni-potsdam.de

Abstract

In today's fast-paced business environment, we see an ongoing trend towards the need for analytics on the latest operational data. The data management layer of enterprise applications needs to adapt to this requirement and In-Memory Column Stores have been proposed as a new architecture that can handle such mixed workload scenarios. A thorough understanding of the resulting query workload is required to validate and optimize data management concepts for this new challenge. Consequently, this paper introduces Database Application Context (DAC) analysis—an holistic framework to analyze database workloads, data characteristics as well as access patterns on specific domain types. We present results for a productive enterprise resource planning system, as well as widely accepted database benchmarks for transactional and mixed workloads. In contrast to existing work, we have analyzed correlations between issued queries and the domain types of accessed attributes. Our main findings are (i) that enterprise workloads are read heavy, (ii) that specific database operators predominantly operate on attributes with a specific domain type, and (iii) that data characteristics differ depending on the data type. Furthermore, based on our analysis of trends in modern enterprise applications, we expect workloads with an increased runtime share of complex queries in the future. These findings help in designing and optimizing the data management layer of modern enterprise applications.

Keywords: Database, In-Memory Database, Workload Analysis, Enterprise Applications

1 Introduction

A traditional enterprise IT landscape largely separates systems for operational data management and reporting. The main reasons for this fact are fundamental differences in functional and performance requirements of both domains (Chaudhuri & Dayal 1997). However, companies often demand more flexible, ad-hoc reporting on the latest data, also referred to as *Operational Business Intelligence* (Gillin 2007, Golfarelli et al. 2004, Kuno et al. 2010, White 2005). To avoid an additional load of analytical capabilities on the operational database, these applications typically rely on synchronized copies of the operational data, so-called *operational data stores* (White 2005).

Maintaining such a redundant real-time copy is complex and expensive.

A radically different approach has been proposed by database architectures that are designed for a mixed workload of both, transactional and analytical queries (Plattner 2009, Kemper & Neumann 2010). The main reasons for the increased performance that allows processing both type of query workloads on a single database instance are massive intra-query parallelism on many-core CPUs and a primary data store in main memory instead of disks or SSDs. It is now possible to store and process data sets of enterprise applications, such as enterprise resource planning (ERP) systems, entirely in main memory (Plattner 2009). Holding the entire data set of an application in main memory, rather than on secondary storage such as hard disks and optimizing data access towards main memory and CPU-integrated memory, offers data access performance that is in orders of magnitudes faster than traditional disk-based systems. Applied to the domain of enterprise applications, the performance gain is so dramatic, that it becomes feasible to build analytical capabilities on top of transactional systems. Additionally, the performance increase allows a rethink of the design of database schema. As an example, complex materialized aggregates that require a predefinition of available aggregations can be replaced by dynamic views that aggregate on the fly, enabling a simpler and more flexible database schema.

Optimizing data structures of the data management layer for these new applications requires a thorough knowledge of the resulting workload in such a scenario with a mix of transactional and analytical applications on a single database instance. Therefore, the objective of this research is to improve the understanding of current and anticipate future database workloads of enterprise applications to optimize the data structures of in-memory databases for these scenarios. As we are right at the beginning of this revolutionary trend, we do not have access to a large productive database installation running in such a mixed scenario. Therefore, our approach presented in this paper is to start with an analysis of a large, productive ERP system as a representative for a transactional system. Based on this baseline, we look into two areas to understand how a mixed workload will look like: (i) we analyze an existing database benchmark, specifically designed for mixed workload, and (ii) we analyze recent developments in enterprise applications and formulate expected changes to the transactional baseline workload. It is important to note, that the obvious approach of combining the queries of existing transactional and analytical applications fails due to different data schema, as explained in more detail in Section 5. As a foundation of these

workload analyses, we have defined a framework to analyze database workloads by classifying queries and data access patterns. In contrast to existing work, we have also analyzed correlations between the domain types of attributes and data access patterns across workloads.

The contributions of this paper are as follows:

- Database Application Context (DAC) Analysis —A framework used to extract and parse queries, database operations, and data characteristics, as well as execution times from different SQL workloads
- A classification and quantification of queries, database operations and data schema definitions from a productive enterprise system, as well as a benchmark for mixed enterprise workloads
- An analysis of the expected changes of the database workload due to new enterprise applications, as well as changed database schema.

This paper is structured as follows: Section 2 gives a brief overview of the basic concepts of in-memory data management for enterprise applications and discuss trends we see in modern enterprise applications. Section 3 describes our framework DAC for classifying database workloads. In Section 4, we introduce the analyzed database workloads and present the results of our analysis in Section 5. We discuss the results of the workload analysis together with the trends we see in modern enterprise applications in Section 6. Section 7 discusses related work and the last Section closes with some concluding remarks.

2 Trends in Modern Enterprise Applications

This section gives a short overview of an in-memory database that allows processing of mixed database workloads of both, transactional, as well as analytical queries. Furthermore, we describe trends that we have identified by analyzing new applications that have been developed for these new database management systems. We will discuss these trends in the context of the results of the analyses of a productive enterprise application and of benchmarks in Section 6.

2.1 In-Memory Database Management

Here, we give a brief overview of the architecture of an in-memory database as introduced as SanssouciDB in (Plattner 2011). Other architectures for in-memory databases targeted towards mixed workloads have been proposed, for example HyPer (Kemper & Neumann 2010). In this paper we briefly introduce the architecture of SanssouciDB to demonstrate the change in database technology that triggers new usage patterns. However, the workload analyses presented in this paper are largely independent on a specific database architecture and can be generally applied as a starting point for optimization of data structures for enterprise application specific data management.

In SanssouciDB, all columns are stored dictionary-compressed to utilize main memory efficiently. Dictionary compression replaces all values by a small integer representative that references the original value that is uniquely stored in a dictionary. Databases that use a column-wise data store typically favor read-mostly analytical workloads, making updates and inserts into dictionary-compressed columns a challenge. To achieve high read and write performance, a common concept in column-oriented databases is to use an additional data store besides the read-optimized

main partition (Krueger et al. 2011): a write optimized differential store.

To achieve durability in case of a system failure, the in-memory database writes log information to persistent memory. This log information is used to recover the latest consistent state in case of a failure and thus guarantees durability. We have proposed an efficient logging mechanism for dictionary-compressed columns in (Wust et al. 2012). We apply multi version concurrency control (MVCC) based on transaction IDs (TID) to determine which records are visible to each transaction when multiple transactions run in parallel. TIDs issued by a transaction manager for each arriving query define the start order of transactions. See (Plattner 2011) for more details.

2.2 Mixed Database Workloads

New enterprise applications that leverage the performance of new database architectures potentially lead to a changed database workload. Our objective is to describe what changes we have to expect in order to optimize database architectures for these use cases. However, observations from analyzing individual applications are hard to generalize. During our research we have looked at existing applications that have been redesigned for in-memory databases, cases of new feature-sets on existing data and also entirely new applications that rely on analytical queries running on transactional data. In this paper, we briefly describe two major trends we have observed, namely replacing materialized aggregates with on-the-fly aggregations, and applications that mix transactional queries and analytical queries. In Section 6, we discuss the implications of these trends on future database workloads, considering our findings from the analysis of existing enterprise workloads using our analysis framework *DAC*, as presented in the following Sections.

On-the-fly Aggregates Replace Materialized Views

The concept of materializing certain views in order to speed up the processing of queries is common practice in database systems (Yang & Larson 1987, P. Larson and H. Z. Yang 1985). Today, all major analytical database systems support the definition of de-normalized, pre-calculated views (Bello et al. 1998, Halevy 2001) for frequently executed queries. In SAP ERP (SAP 2013), the concept of materialized views is realized using additional, transactional tables that are maintained by the application logic. They hold subsets or even entire table projections of huge transactional relations, filtered and aggregated on pre-defined granularity. This redundancy has been necessary to get reasonable response times for complex aggregates and lookups on huge tables. However, there are three major limitations:

- Data redundancy,
- Reduced query flexibility and
- Maintenance cost (inserts and updates of materialized tables)

Consequently, an in-memory database replaces materialized aggregates using on-the-fly queries that run on the original tables (Plattner 2009). In order to estimate the impact of such a redesign, we have analyzed the workload on the most relevant materialized tables in SAP ERP Financial, based on the execution count:

- Sum table holding customer balances on a monthly basis

- Secondary index table that allows fast access to accounting documents
- Sum table holding balances of general ledger account on a monthly basis

Each query that calculates these aggregates on the fly based on the transactional schema contains a join on at least two transactional tables, calculating aggregates, grouped by a set of attributes.

Mixed Workload Applications

Providing a platform that can run transactional and analytical queries on a common dataset opens the way for new business applications. As an example, (Wust et al. 2011) proposes an application that gives sales representatives a tool to generate cross-selling recommendations on the fly adjusted to the actual customer. Product recommendations are calculated on-the-fly, in order to handle a range of different parameters (products, regions, branches or customers). Additionally, the abundance of pre-calculated, materialized result sets leads to a less complex data schema. Prior to xSellerate, product recommendations needed long-running, inflexible batch-job operations that were persisted in dedicated, materialized tables. xSellerate shows the need and feasibility of analytic queries on transactional data. As another example, (Tinnefeld et al. 2011) presents a real-time availability-to-promise service that calculates a stock projection in real-time for each request; this application includes transactional queries for ordering products that depend on analytical style queries calculating the current stock level, and therefore need to run on a consistent data set of a single database instance. Furthermore, (Krueger, Tinnefeld, Grund, Zeier & Plattner 2010) gives a detailed overview of the characteristics of these applications. Analyzing at the queries issued by these applications, we see large joins, as well as groupings and aggregations.

3 Database Application Context (DAC) Analysis

This Section introduces our framework to analyze database workloads. We start with a presentation of the overall process and then describe the individual steps in more detail in the subsequent section.

3.1 Overview

The motivation of this research is to get a thorough understanding of the database workload issued by today's enterprise application and derive characteristics of future mixed workload scenarios. In the context of columnar databases we intend to provide rationales for optimizations of data structures, compression and operators based on existing domain type schema definitions.

Typical questions that arise are: What kind of statements and operations are issued most by certain workloads and on what attributes do they operate? Answers to these questions can help to optimize the data management layer for specific workloads. To find answers to these questions, we designed a framework called Database Application Context (*DAC*) analysis, shown in Figure 1.

An application's workload, data schema and data characteristics are the elementary aspects of an application that determine the performance of a database. *DAC* describes and quantifies those aspects with the objective of understanding the predominantly used

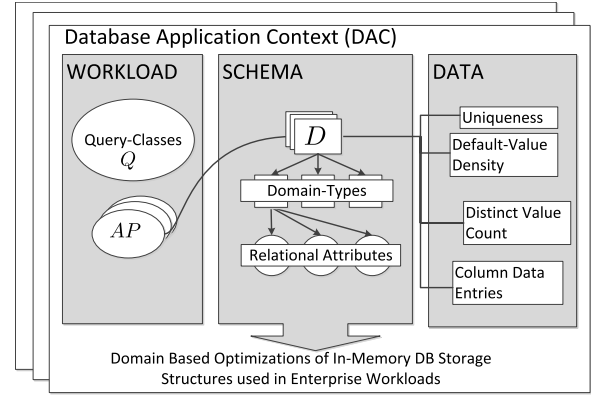


Figure 1: Database Application Context Analysis

queries, database operations, data types, and data characteristics for a given application.

First, we use the access pattern model *AP*, to describe database operations such as joins, groupings, selection, projections and aggregations parsed in each query statements. Based on that the query-class model *Q*, separates statements into distinct groups. Then all table attributes in *AP* are classified using a domain model *D*, that separates columns based on their domain information.

As a first step of the analysis, the application workload, its data schema definition and data statistics are captured from various sources. Then the workload is parsed for specific *AP* and the schema information is classified before it gets consolidated into a central data repository. Typically the workload is captured as an SQL trace, or as a snapshot of the SQL plan cache of a database, ideally annotated with performance data, such as execution count, runtime per query, and returned rows. Obviously, retrieving all performance data is only partially possible in a live system due to potential overhead. The data schema and information about the data can be extracted directly from the database.

The different models are applied to classify workloads into distinct query classes (*Q*), characteristic operations into access pattern (*AP*) and data segments, respectively relational attributes into domain groups (*D*), based on their domain-type information. Additionally, the data of each domain group in *D* is characterized, using the average data distribution, distinct value count and total number of the columns in each group. The extracted workload characteristics are then mapped with characteristics of the data schema and data statistics to find correlations for each workload.

3.2 Models of the DAC Analysis

This section describes the models used in *DAC* analysis in detail. We apply different models to analyze workload queries, access patterns as well as an application's data schema and data characteristics.

Classification of Queries

A first step towards characterizing a workload is to classify the queries of each workload.

The separation of statements into query classes is done by parsing the type of SQL statement as well as specific operations in the query. We currently consider the SQL statement types SELECT, INSERT, DELETE, and UPDATE. That allows a simple separation of all select- or read- (Q_R), delete- (Q_{DEL}),

	Description
AP_{Agg}	Aggregate operations (e.g. count(), max(), min(), sum())
AP_{Join}	Join operation on a field
AP_{Sel}	Data selection using where predicates
AP_{Grp}	Group by operation on a field
AP_{Sort}	Order by operation on a field
AP_{Proj}	Projection of a field

Table 1: Access Patterns

update- (Q_{UPD}), and insert- (Q_{INS}) queries. By analyzing the data characteristics stored in the database, Q_R -queries are further divided into Point or Single Select (Q_{Single}), Range Select (Q_{Range}), and Complex Select ($Q_{Complex}$) statements. We define Range Select (Q_{Range}) as statements that have a low selectivity and read a set of data entities. In contrast to Complex Selects ($Q_{Complex}$), they do not aggregate, join or group a set of data, but only read them. Point or Single Select (Q_{Single}) statements have a high selectivity, reading only single data sets. Finally, Complex Selects ($Q_{Complex}$) are defined as statements that operate and modify sets of data. Aggregate, grouping and join operations are considered complex, because they change and re-define the original data structure. Q_{Single} -queries must provide a equi-predicate on the (compound)-key attribute. Q_{Range} -queries are identified by a range-select operation or by a selection on non-key table attributes. $Q_{Complex}$ -queries are characterized by complex database operations, such as joins, groupings, or aggregations.

Classification of Access Pattern

Data Access Patterns (AP) considered in this paper are all standard SQL operations such as a projection, selection, join, grouping, aggregation or sort on a particular table column. Table 1 gives an overview of the considered access patterns. Provided that we have performance indicators for the issued SQL statements of the analyzed workload, such as execution count, rows processed and time elapsed, we can rank the relevance of database operations (AP) that are extracted from the statements.

We believe that a detailed access pattern analysis helps to characterize the application workload and focus on the relevant optimizations strategies. For example, conducting an access pattern analysis, it is possible to tell on which data segments join operations are executed, how often they are executed and if there are multi-column or single-column joins.

Listing 1: Example SQL Statement

```
SELECT * FROM DISTRICT WHERE
WID=:B2 AND DID>B1
```

As an example, consider the SQL query illustrated in Listing 1. Assuming that it was executed ten times, and that we retrieved the average runtime of 500 ms on the query on statement level. Then, we could capture access patterns and performance characteristics as shown in Table 2. Besides the AP we use an additional text field to capture AP relevant information, such as the aggregate function in AP_{Agg} or the predicate for AP_{Sel} .

It is important to note in that example, that the (run-)time performance for each AP is derived from the original query. Although, a more fine granular run-time information on access pattern level is supported by the model and would allow a better analysis

Type	Table	Attr	Exec	Time	Info
AP_{Proj}	DISTRICT	*	10	500	
AP_{Sel}	DISTRICT	WID	10	500	=
AP_{Sel}	DISTRICT	DID	10	500	>

Table 2: Extracted AP of Listing 1

of the importance of each access pattern, it is typically not a feasible option for analyzing productive workloads due to the overhead of generating the performance data. Therefore, we used the runtime information on statement level in our analysis presented in Section 5.

Analysis of the Data Schema

In a relational database, relations are the basic entities, where each element is defined by attributes. Each attribute has a defined data type, either a primitive type, predefined by the database system or a user defined type, also called domain-type. In enterprise applications, domain-types are used in a schema to enforce data integrity among attributes of different relations. A domain-type limits the range of possible values, of all columns of that type. It is an abstraction of a primitive data-type, having a set of additional, application-specific constraints. For example, a check constraint in a domain-type MONEY requires all price columns to be greater than zero or a foreign-key constraint in domain PRODUCT.ID limits all columns to the values defined in the primary table column.

Depending on the domain definition (constraints) the number and change frequency of legal values differs. Domains, such as NAME, STREET or DESCRIPTION, defined only by a data type do not have a well-defined, reasonably-sized list of values. They are called qualified domains. In contrast, the value range of enumerated domains (e.g. GENDER, CATEGORY.ID, PRODUCT.ID or CUSTOMER.ID) and all columns defined by them is predetermined and sorted by default. Either by a incremental primary table column or even fixed by the application itself.

Domain constraints provide valuable information for columns that are dictionary compressed. Especially enumerated domain constraints, as they provide a dictionary-like, well defined value range. Moreover, we believe that a column's domain type also has an impact on how it is accessed by a workload. For example, an enumerated domain column is likely to be used in a join operation in order to link the primary table relation with its own. In order to prove this assumption, the DAC analysis references a model to segment columns, based on their domain.

Our column classification first separates qualified (D_Q) and enumerated domain (D_E) columns. Within D_Q there are three subgroups: datetime type (D_{Q-DT}), numeric type (D_{Q-N}), and character type (D_{Q-C}) qualified domains, are separated based on the data type of the domain. We divided enumerated domains (D_E) by the type of their primary domain table and adopted the SAP specific definition of master data, transaction data and Organizational and customizing tables (SAP 2013).

While master data tables and their list of domain values ($D_{E-Master}$) are frequently read, new inserts are rare. Transaction data, respectively domain values ($D_{E-Trans}$) are frequently inserted. Organizational and customizing domain values (D_{E-Cust}) are unchanged during runtime, because they are defined before the system is run productively. Finally,

D_{E-Fix} domain columns have a value range that is defined at design time. We adopted this distinction, as it can provide valuable insights for designing specific compression techniques for different types.

Analysis of Data Characteristics

Data characteristics, such as number of distinct and total data items, value distribution and default-value density are essential for database compression and optimization. As an example, the effectiveness of dictionary compression depends on the number of distinct values of a column.

Therefore, being part of our *DAC* analysis, we extract three basic data indicators. In contrast to previously conducted data characterizations, we aggregate these indicators on domain level for enumerated domain types. We determine the following figures for each column:

- Default-value count,
- Distinct value count and
- Total data items (length).

The default-value count describes the number of data entries that are equal to the most frequent value in the column. This might be a NULL value as well as any other domain value. Distinct values count is the number of unique values in a column. Total data items of a column is equal to the number of rows in the table. Based on this information we derive the following characteristics for each domain group D .

- Distinct item count,
- Total data item count and
- Default value share

For D_Q domains the distinct data items are the sum of distinct values of all columns in that domain. In contrast, the number of distinct values in a D_E domain is only based on the distinct values of the primary domain column, because all other columns use a subset of the primary domain column. Total data items in a domain are the sum of all column data items. Default value share is calculated by the number default values across all domain columns and the total data item count. It gives us the share of default data items in a domain.

4 Analyzed Application Contexts

This Section introduces the different database application contexts we have analyzed.

Transactional applications are the backbone of any enterprise. Invoices, sales orders and accounting documents - all enterprise specific business entities are first captured and processed in a transactional systems. Analytical applications built upon a star schema, typically trade intense data compression, data redundancy as well as limited update performance for the efficiency to process and analyze increasing amounts of data in a fraction of time. However, transactional applications, built on a normalized data schema remain the necessary prerequisite and source for these systems. As proposed by Plattner in (Plattner 2009), our research builds on on the assumption that the central data source for mixed-workloads is based on a normalized transactional schema. Consequently, all application contexts analyzed here are based on a normalized data schema.

The first application context is a productive transactional workload, captured from a productive SAP

ERP (SAP 2013) system of a company with roughly 20,000 employees. It provides the status-quo and the basis of future mixed-workload applications. The well established *TPC-C* benchmark (TPC 2010) is used to compare a transactional benchmark with the productive, transactional workload. Furthermore, we have analyzed the mixed-workload benchmark *CH-benCHmark* (Cole et al. 2011).

4.1 Transactional *Enterprise* Application

Our transactional context analysis is based on data from a large productive SAP ERP system, used for financial-, sales-, distribution- and production- processing. The workload of that application was traced over the period of one work-week. During that time, there were no untypical periodic loads, such as year-end-processing, etc. The analysis of a single system may not be representative for transactional enterprise applications in general. However, as the analyzed system had only few modifications to the standard SAP ERP system, which is widely used and a de-facto standard in most industries, it can be considered as highly representative. Besides, we consider the possibility to analyze a productive ERP system of a company of that size as a great opportunity and expect that these results are valuable for the community. We constantly try to find more companies that are willing to share data to increase the data base.

In the presented case, the workload information was extracted from the shared pool buffer of the underlying database of the running ERP system. Along with the executed SQL statements (without variable binding), we tracked relevant usage statistics of each statement, such as the number of executions, the time elapsed (runtime) and the number of rows processed.

In total, we extracted the following quantities:

- Total number of SQL queries: 144.000
- Analyzed SQL queries: 23.000 (92% load)
- Users: \approx 1.000 active SAP Users (24h)
- Number of SQL executions: 3.300.000.000
- Rows processed: 20.200.000.000
- Total query runtime: 1.900 h

4.2 Transactional *TPC-C* Benchmark

In order to allow references to existing research and compare the transactional, enterprise application with a benchmark context, we analyzed the workload and schema of a *TPC-C* benchmark (TPC 2010). In the following, this workload is referred to W_{TPC-C} . W_{TPC-C} consists of 33 transactional queries, each having a defined share of the entire runtime and execution count.

4.3 Mixed-Workload *CH-benCHmark* Benchmark

With the motivation of comparing the performance of mixed workload databases, a group of researchers designed a benchmark, called *CH-benCHmark* (Cole et al. 2011). *CH-benCHmark* is a composite benchmark, combining the well known, transactional *TPC-C* (TPC 2010) and the analytical *TPC-H* (TPC 2013) on the normalized *TPC-C* data schema.

(Cole et al. 2011) examine three scenarios, weighting OLTP and decision support (DS) workload streams with different factors. Then they compare the execution results to see how those workloads effect each other in terms of *tpmC* and *QphH*, using

a reference number where each workload was run in isolation. The number of OLTP and DS streams does not emulate the number of business users, but can be interpreted as a ratio between those two workload types. For our workload analysis we chose two workload mixtures. One, where OLTP and DS queries have the same share - W_{CH11} and, as a reference for an analytical workload, one that consists of TPC-H queries only - W_{CH01} .

As we do not intend testing performance scalability of a system under load or how the different workloads effect each other, we are only interested in relative execution statistics (execution count, runtime) of each query in the same time frame. The benchmark conducted in (Cole et al. 2011), running on a PostgreSQL system provided the reference numbers. It is based on the assumption, that there is no contention between workloads, that they run on the same database size and that there is no think-time between requests. After a 5 minutes warm-up, one OLAP client running in isolation performs in average 5200 tpmC, whereas one OLAP stream runs 902 QphH. These reference numbers were used to calculate the execution statistics of each query presented in Section 5.

5 Analysis of Workload Characteristics

In this Section, we present the results of our *DAC* analysis of the workloads described in Section 4.

The results of our analysis are presented for each of the models introduced in Section 3 (i) analysis of query classes, (ii) analysis of access patterns, (iii) analysis of the accessed domain types, and (iv) data characteristics. A comprehensive discussion of the results will follow in Section 6.

5.1 Classification of Queries

Figure 2 shows the query classes of the workloads introduced in Section 4, by displaying for each of the six query classes (Q_{Range} , Q_{Single} , $Q_{Complex}$, Q_{INS} , Q_{UPD} , Q_{DEL}) of the Q model the percentage of total executions and total runtime for each query class. This way, we can compare the relative importance of each query class between workloads, even as they differ in absolute numbers.

Looking at the results of $Q_{Complex}$ in Figure 2, it is notable that the share of the total runtime is higher than the share of executions, which is in line with our expectations when defining the query classes. Comparing the workloads, it is striking that W_{TPC-C} and W_{CH11} have a much higher number of update and insert statements as the productive workload W_{Trans} . Comparing W_{TPC-C} , W_{CH01} and W_{CH11} we see an increasing percentage of complex query runtime, which is as expected given the increasing amount of analytical queries.

The total number of executed queries of W_{CH01} is almost negligibly low compared to W_{TPC-C} , as the analytical queries run orders of magnitude longer. The W_{CH11} workload is a mix of W_{TPC-C} and W_{CH01} . Consequently, the total number of executed queries in W_{CH11} is almost equal to W_{TPC-C} . However, comparing the runtime of the statements we see an increase of $Q_{Complex}$ queries for W_{CH11} and obviously for W_{CH01} as well, compared to the transactional workloads W_{TPC-C} and W_{Trans} .

The “total execution count” provides a runtime-independent figure of the share and the impact of each query class Q . Unfortunately, this is not meaningful for the W_{CH11} benchmark. Because the runtime for the TPC-C part is the same as for TPC-H part, it

causes very few TPC-H queries to be executed. A difference in execution count between W_{TPC-C} and W_{CH11} is almost not perceivable. However, comparing the runtime of the statements, we see an increase of $Q_{Complex}$ queries for W_{CH11} , and obviously for W_{CH01} as well, compared to the transactional workload W_{TPC-C} . Hence, we consider runtime as the performance indicator to analyze the relevance of access patterns in the next section.

5.2 Classification of Access Patterns

In order to measure the occurrence and importance of individual database operations, we use the extracted *AP* and the performance figures of their SQL statement. For each *AP* we know to which SQL statement it belongs. Based on the performance figures of those statements, we measure the share and impact of *AP*. This way, we determine the relevance of an *AP* in a workload. It is important to mention that we could only extract runtime information on statement level. Thus, the indicated percentage of total runtime for an *AP* indicates that the execution time of all queries that contained this specific *AP* accounts for this percentage, but not necessarily the *AP* itself. Extracting more fine granular performance data on database operation level would have had a too high performance impact on the live system and was not a viable option.

Figure 3 illustrates and compares the share of statements having certain *AP*, in W_{Trans} , W_{TPC-C} , W_{CH11} and W_{CH01} . It shows that joins do not account for a significant percentage of the total runtime in W_{TPC-C} , whereas W_{Trans} shows a relevant share. As expected the impact of complex database operations such as aggregations, sorting, joins or grouping plays a much greater role in W_{CH11} and W_{CH01} . Although this analysis does not reveal the individual runtime contribution of each of these more complex *AP*, we can confirm that more complex *AP* play a significant role in analytical style workloads executed on a normalized data schema.

In the next section the qualitative characteristics of access patterns in workloads will be analyzed, illustrating detailed figures on the data segments that are accessed by *AP*.

5.3 Classification of Data Schema

This section shows the qualitative analysis of the access patterns *AP* and their correlations with accessing certain domain types. We restrict the presentation here to read queries Q_R , as they account for most executions and runtime in our productive application context W_{Trans} .

Each *AP* references a data segment (table and attribute), has two performance indicators (execution count and runtime) and an additional text-field, used to qualify the *AP* with additional information, such as select-predicate or aggregate function (Section 3.2).

In this paper, we show in detail the access patterns of selections and joins, as selections are the most used access pattern and join is the access pattern that typically has the longest execution time. We have further analyzed the accessed data types of aggregations and groupings and briefly mention the results.

For aggregations, SUM, AVG as well as MAX and MIN are primarily used on D_{Q-N} columns. Especially SUM as the basic analytic operation is only conducted on numeric domain columns. The COUNT function is typically applied to the anonymous star-column and returns the number of rows in a query.

Group by operations are the requirement for most aggregate functions. As shown in Figure 3 they are

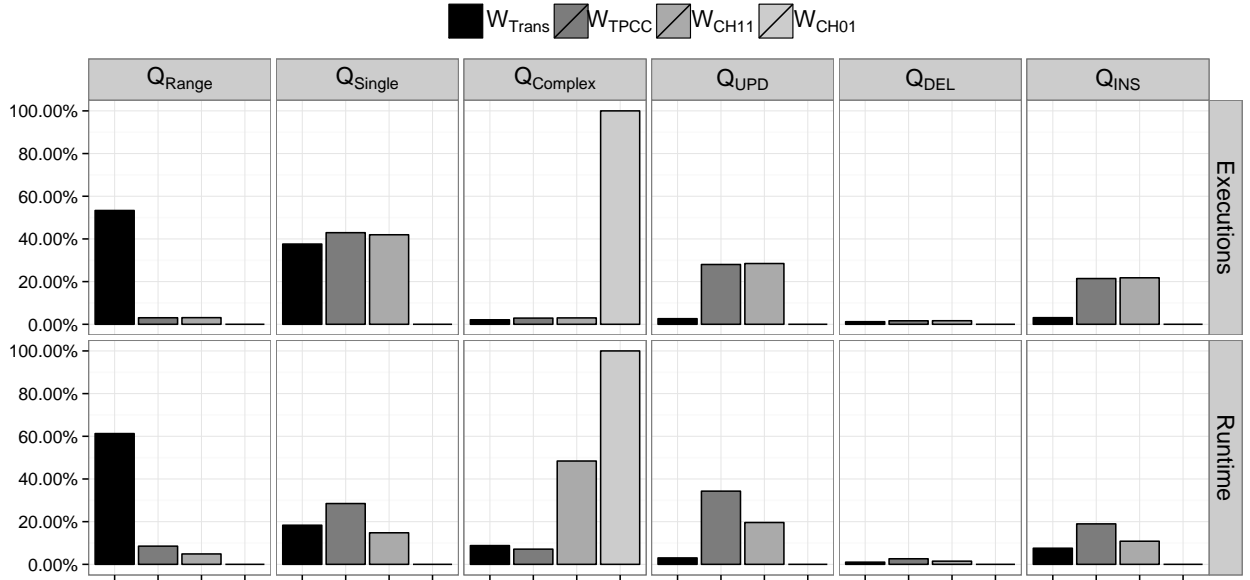


Figure 2: Query-Class Workload Characteristics

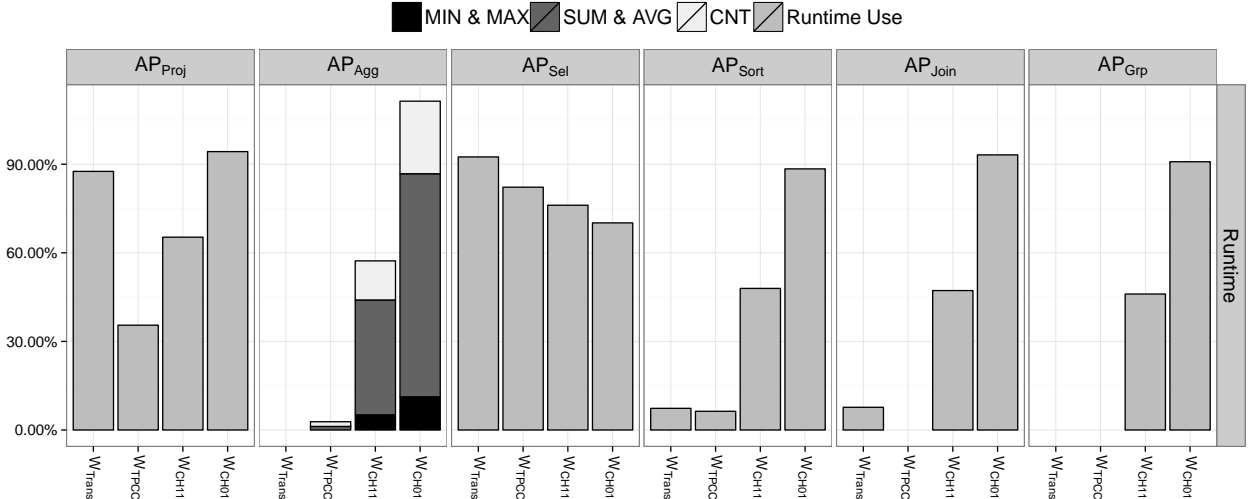


Figure 3: Access Pattern Workload Characteristics, based on Total Runtime

essential and intensively used in analytic queries. Our analysis showed that groupings happen largely on enumerated domain types. Although groupings happen on name or other qualified domains in W_{CH11} and W_{CH01} , we would expect them to be implemented as enumerated domains in productive applications, as names are typically not enforced to be unique. E.g. instead of using the customer.name attribute the customer.id attribute would be used for grouping. The analysis of accessed domain types revealed that in W_{Trans} , 82% of all AP_{Grp} are conducted on D_E columns. Besides, there are 17% D_{Q-DT} columns used to group selected data sets, based on their date-time attribute.

Data Selection on Domain Columns

Data selection is the most used AP in all analyzed workloads. There is almost no query without data selection operation(s) as shown in Figure 3. The AP_{Sel} has an additional variable, called “predicate-type”.

In Figure 4a, we use three predicate-groups: RANGE ($<$, $<=$, $>$, $>=$, between), LIKE and EQUI ($=$, $!=$, IN), to concentrate similar predicates in groups. The bars in the chart represent the overall fraction of AP_{Sel} predicate-groups within each W based on the runtime of their SQL statements.

In the purely transactional workloads W_{Trans} and W_{TPCC} the EQUI-predicate is predominant. RANGE and LIKE predicates play an important role in the workloads W_{CH11} and W_{CH01} that also consist of analytical queries. Looking at the domain-type distribution of EQUI predicated, most selections operate on D_E columns: From left, W_{Trans} to right, W_{CH11} , 77%, 98%, 89% and 57% operate on D_E and especially on D_{E-Cust} columns. Analogously, most RANGE predicates operate on qualified domains: 89% in W_{Trans} , 89% in W_{CH11} , and 99% in W_{CH01} . Only a small fraction of 11% of RANGE- AP_{Sel} operate on $D_{E-Trans}$ columns.

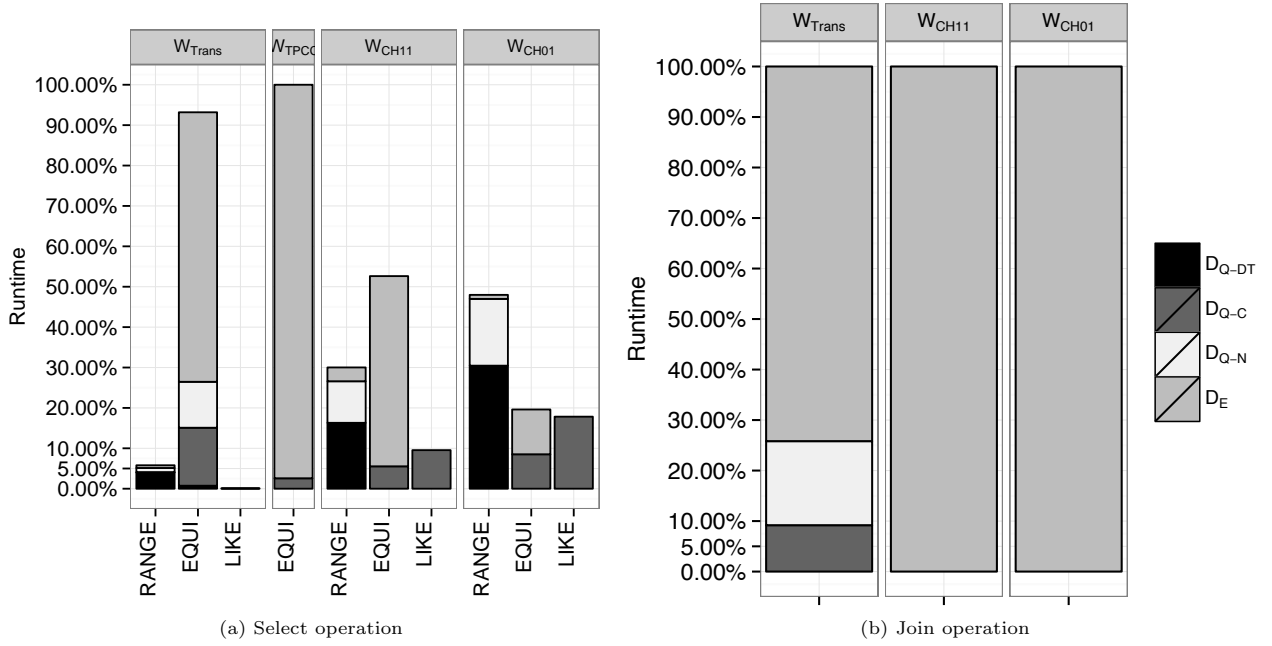


Figure 4: Operation / domain group dependencies based on runtime

Join Operations on Domain Columns

The access pattern AP_{Join} is mapped to the join conditions of SQL statements. One AP_{Join} always reflects one join attribute. Thus, a standard join-condition on one attribute, over two tables yields two AP_{Join} . This way, we can compare the domain-types of join-couples and see that they always relate to the same domain. It shows that while attribute names can be different, their legal set of possible values is always the same.

Figure 4b shows the domain-types used in AP_{Join} based on the SQL runtime. The presented analysis only considers equi-joins, as other join types have been negligible in runtime and execution time in the analyzed workloads. In W_{Trans} we see that 79% of all join attributes are D_E columns. In W_{CH11} and W_{CH01} all join attributes are D_E columns. Convinced that only D_E columns are appropriate to join relations we further investigated the 21% of D_Q columns used for join operations in W_{Trans} . It shows that these columns are also key attributes, having a primary table column. However, they are not defined as an D_E domain type in the schema definition. This might be due to historic reasons or simply suboptimal schema design.

Domain-Type Distribution and Characteristics in Data Schema

In this section we analyze the distribution of domain groups D in the schema of W_{Trans} , as well as the characteristics of the data that is stored in those groups. We only show W_{Trans} , as the data schema of the benchmarks W_{TPC-C} , and similarly W_{CH01} and W_{CH11} , are too simplified to be representative for productive enterprise systems. While D_E domains have a shared, well-defined set of distinct values for all columns of the same domain, D_Q columns do not have such a list of domain values. Every column defines its own distinct values.

The y-facet “Distinct Domain Items (%)” of Figure 5 shows the sum of all distinct values for each domain group. It points out that D_Q domains make

up 90% of all distinct values stored in the analyzed W_{Trans} . Especially D_{Q-C} and D_{Q-N} have a considerably high number of distinct values. D_E columns contribute only 10% of the distinct values. However, the share of “Data Items (%)” is almost equally distributed between D_E and D_Q . While not shown in the figure, the same spreading between D_E and D_Q can be seen for the number of columns, respectively domains in the data schema of W_{Trans} . In detail there are $\approx 17\%$ D_{E-Fix} , $\approx 28\%$ D_{E-Cust} , $\approx 3\%$ $D_{E-Trans}$ and $\approx 7\%$ $D_{E-Master}$ columns in the productive data schema. Additionally, there are $\approx 7\%$ D_{Q-DT} , $\approx 15\%$ D_{Q-N} and $\approx 23\%$ D_{Q-C} columns.

The low share of distinct values per domain, accompanied by the high share of data items in D_E columns results in a very low uniqueness (share of distinct values among all data items).

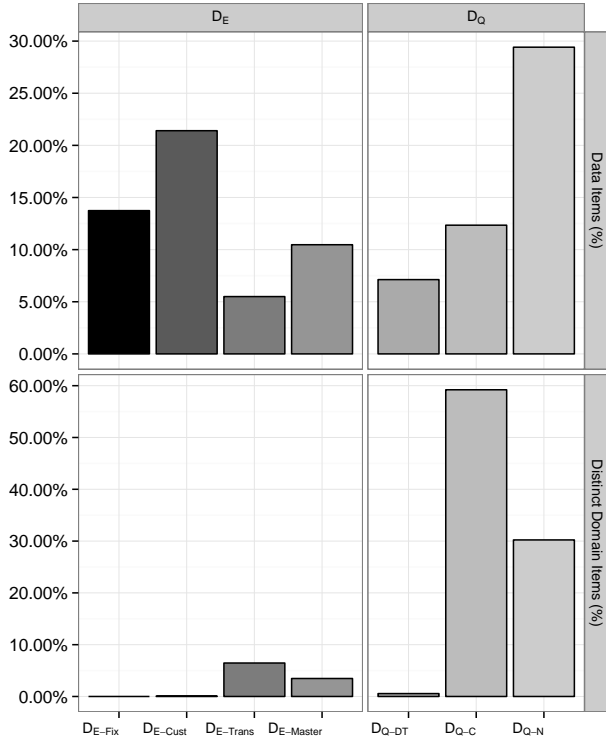
As generally shown in (Huebner et al. 2011), the distinct values of a domain are not equally distributed. Our analysis confirms this and reveals a significant share of default values among D_E columns. On average 84% of all D_{E-Fix} and 73% of all D_{E-Cust} data items contain the column’s default value. As these are just average numbers across all domain group columns a default-value aware compression algorithm can be very effective for certain relational attributes.

6 Discussion of Results

This section summarizes the main findings from the results of our DAC analysis presented in Section 5.

Our major insights of the DAC Analysis are:

- Enterprise workloads are read heavy; interestingly, we see a much higher percentage of read queries compared to the industry benchmark TPC-C.
- Specific database operators predominantly operate on attributes with a specific domain type.
- Data characteristics differ depending on the domain type.


 Figure 5: Data Characteristics of W_{Trans}

- We expect a trend towards more complex queries and operators with the introduction of modern enterprise applications.

We will briefly explain each of the insights and discuss implications.

Based on the analysis presented in Section 5.1, we see that more than 90% executed statements which account for around 90% execution time in the productive transactional workload are read-only statements. In contrast, the TPC-C workload shows an almost equal distribution of read and write statements. This is in line with the findings in a previous analysis of data access patterns (Krueger, Grund, Zeier & Plattner 2010), which derive the adequacy of read-optimized column stores based on this insight. This confirms that the system we have analyzed does not show an exceptional behavior and gives us confidence that we can generalize our findings.

When analyzing the domain types of attributes and the operators that access these attributes, as shown in Section 5.3, we see a clear correlation between domain types and operators across all workloads. Equi-selects, being the most important operation in all workloads, used in all query-classes (except Q_{INS}), strongly depend on enumerated domain columns, whereas D_{Q-N} and D_{Q-DT} columns make up more than 90% of all range-selects in all workloads. The importance of enumerated domains is also shown for join and grouping operations, that primarily depend on D_E domain-types. Character columns are only used in like-selection and data projections. These findings are valuable for optimizing operator implementations for domain types they mostly operate on. We think that a domain based optimization of data structures is a promising approach for highly normalized relational databases. Each group of domain types holds specific characteristics that can be leveraged by a different data structure. What we have

seen is that while D_Q columns contain any possible, user-defined data, D_E columns are determined and controlled by the application, only. As a consequence D_E columns are essential to enforce data consistency as well as join, select and group data. As they are deterministic they are preferred, if not required for many database operations. On the other hand range-selections, aggregate functions and like-selection are irrelevant or even illegal on these D_E columns, because their domain values are always of nominal scale.

A closer look at the value distributions of the data set of the productive ERP system shows that the number of distinct values in all columns with a qualified domain is roughly ten times higher than the distinct values in all columns with enumerated domains which share values among columns (See Section 5.3).

Domain type information are available during design time. Compared to approaches that analyze workloads and data characteristics during runtime, it does not pose any overhead. We see that the effectiveness of different compression techniques such as sorted, unsorted as well as shared and attribute-wise dictionary compressions depends on the specific domain context. A shared or global dictionary for example seems reasonable for D_E attributes, in order to leverage a common dictionary encoding during join processing. Besides, our result show that it is questionable if D_{Q-C} columns profit from column-wise data structures as there are almost no operations on single D_{Q-C} columns. Therefore, we plan to leverage these insights by designing specific encoding and compression techniques depending on the domain type of attributes.

Based on our identified trends in Section 2.2, we see more complex data operations in future enterprise workloads. Ad hoc aggregations as well as more analytical style queries will lead to a mix of shorter running transactions, as well as potentially longer running queries that read, join and process large amounts of data. In line with the analysis of W_{CH11} shown in Figure 2, we expect a higher absolute number of transactional queries, but a larger share of the total execution time accounted by analytical queries. This will be further intensified by a trend to push logic into the database instead of only reading data from the database in bulk and process it in the application. As a potential implication, short running transactional queries, as well as the more complex queries, might compete for resources. In the worst case, complex queries can occupy all database resources and block short transactional queries from executing. Hence, a direction for future work is to design workload management systems for mixed workloads.

7 Related Work

We have identified related work in two fields: characterization of database workloads, as well as the underlying data. Our work extends prior work in workload characterization by analyzing the correlation of database queries of a workload as well as accessed data. Furthermore, we present results that have been obtained from analyzing the workload of a large scale, productive enterprise application system.

7.1 Workload Characterization

Workload analysis and characterization is the requirement for many performance studies and a foundation for various benchmarks, built as the synthesized, controlled workload, which can be used to measure and compare the performance of different environments. (Elnaffar & Martin 2002) summarize and compare

workload analysis conducted in different application fields. They basically classify and characterize analysis techniques into two categories: *static* and *dynamic*. Static techniques, such as descriptive statistics, histogram, component analysis and clustering are used to characterize static workload components. Dynamic techniques are used to describe the probability of system transitions. Knowing a certain performance figure at a certain point in time is not sufficient. Instead the workload components must be captured periodically to show dependencies between different parameters. Based on our motivation, the analysis conducted in this paper is of static nature.

7.2 Data Characterization

In general, data characteristics of enterprise systems have been investigated in (Krueger et al. 2011), (Krueger, Grund, Zeier & Plattner 2010) and (Huebner et al. 2011). They show that many columns in standard enterprise systems have a low number of distinct values. A share of 35% of all analyzed table columns is even unused, due to the wide table schema needed in standard software to support various industries and customer requirements. Besides, Huebner et al. analyze the value distribution of FI table columns in an SAP ERP system. They found that half of all columns are best approximated by a uniform distribution, while the other part adheres to a zipf pdf distribution. Based on that information they built a merge strategy that is optimized for pdf zipf distributed columns.

8 Conclusion and Future Work

In this paper, we have presented Database Application Context (DAC) Analysis, a framework used to extract and classify queries, used database operations, and data access patterns, as well as execution times from SQL workloads. With *DAC*, we have analyzed a productive enterprise resource planning system, as well as established database benchmarks. Based on an analysis in trends in modern enterprise applications, we have derived expected changes to the workloads of enterprise applications in the future. We are confident that our results can be a valuable input for optimization data structures of the data management layer of enterprise applications. We plan to leverage these findings in two dimensions: (i) develop efficient database operations and compression techniques for enumerated domains that are shared among attributes, and (ii) implement effective query schedulers to handle a mixed workload of different query classes.

References

- Bello, R. G., Dias, K., Downing, A., Feenan, Jr., J. J., Finnerty, J. L., Norcott, W. D., Sun, H., Witkowski, A. & Ziauddin, M. (1998), Materialized views in oracle, in 'VLDB'.
- Chaudhuri, S. & Dayal, U. (1997), 'An overview of data warehousing and olap technology', *SIGMOD*.
- Cole, R., Funke, F., Giakoumakis, L., Guy, W., Kemper, A., Krompass, S., Kuno, H., Nambiar, R., Neumann, T., Poess, M. & Others (2011), The mixed workload CH-benCHmark, in 'DBTest'.
- Elnaffar, S. & Martin, P. (2002), Characterizing computer systems' workloads, Technical report, School of Computing, Queens University.
- Gillin, P. (2007), 'Bi @ the speed of business', *Computer World Technology*.
- Golfarelli, M., Rizzi, S. & Cella, I. (2004), 'Beyond data warehousing: what's next in business intelligence?', *DOLAP*.
- Halevy, A. Y. (2001), 'Answering queries using views: A survey', *VLDB*.
- Huebner, F., Boese, J.-H., Krger, J., Renkes, F., Tosun, C., Zeier, A. & Plattner, H. (2011), A cost-aware strategy for merging differential stores in column-oriented in-memory dbms, in 'BIRTE'.
- Kemper, A. & Neumann, T. (2010), Hyper hybrid oltp&olap high performance database system, Technical Report May, Technische Universitaet Muenchen.
- Krueger, J., Grund, M., Zeier, A. & Plattner, H. (2010), Enterprise application-specific data management, in 'EDOC 2010'.
- Krueger, J., Kim, C., Grund, M., Satish, N., Schwalb, D., Chhugani, J., Dubey, P., Plattner, H. & Zeier, A. (2011), 'Fast updates on read-optimized databases using multi-core cpus', *VLDB*.
- Krueger, J., Tinnefeld, C., Grund, M., Zeier, A. & Plattner, H. (2010), A case for online mixed workload processing, in 'DBTest'.
- Kuno, H. A., Dayal, U., Wiener, J. L., Wilkinson, K., Ganapathi, A. & Krompass, S. (2010), Managing dynamic mixed workloads for operational business intelligence, DNIS.
- P. Larson and H. Z. Yang (1985), 'Computing Queries from Derived Relations', *VLDB*.
- Plattner, H. (2009), A common database approach for oltp and olap using an in-memory column database, *SIGMOD*.
- Plattner, H. (2011), Sanssoucidb: An in-memory database for processing enterprise workloads, in 'BTW', pp. 2-21.
- SAP (2013), 'Sap erp', <http://www54.sap.com/solutions/bp/erp.html>. [Online; accessed 22-August-2013].
- Tinnefeld, C., Mueller, S., Zeier, A. & Plattner, H. (2011), Available-to-promise on an in-memory column store, in 'BTW'.
- TPC (2010), Tpc benchmark c (standard specification) - revision 5.11, Technical report, Transaction Processing Performance Council.
- TPC (2013), Tpc benchmark h (standard specification) - revision 2.16.0, Technical report, Transaction Processing Performance Council.
- White, C. (2005), 'The next generation of business intelligence: operational bi', *DM Review Magazine*.
- Wust, J., Boese, J.-H., Renkes, F., Blessing, S., Krueger, J. & Plattner, H. (2012), Efficient logging for enterprise workloads on column-oriented in-memory databases, in 'CIKM'.
- Wust, J., Krueger, J., Blessing, S., Tosun, C., Zeier, A. & Plattner, H. (2011), xsellerate: Supporting sales representatives with real-time information in customer dialogs, in 'In-Memory Data Management'.
- Yang, H. Z. & Larson, P.-A. (1987), Query transformation for psj-queries, in 'VLDB'.

Shape Predicates Allow Unbounded Verification of Linearizability Using Canonical Abstraction

David Friggens^{1,2}

Lindsay Groves²

¹ University of Waikato
Email: friggens@waikato.ac.nz

² Victoria University of Wellington,
Email: lindsay@ecs.vuw.ac.nz

Abstract

Canonical abstraction is a static analysis technique that represents states as 3-valued logical structures, and is able to construct finite representations of systems with infinite state spaces for verification. The granularity of the abstraction can be altered by the definition of instrumentation predicates, which derive their meaning from other predicates. We introduce shape predicates for preserving certain structures of the state during abstraction. We show that shape predicates allow linearizability to be verified for concurrent data structures using canonical abstraction alone, and use the approach to verify a stack and two queue algorithms. This contrasts with previous efforts to verify linearizability with canonical abstraction, which have had to employ other techniques as well.

Keywords: canonical abstraction, concurrent data structures, linearizability, verification

1 Introduction

Canonical abstraction (Sagiv et al. 2002) is a powerful static analysis technique that can be used to construct bounded finite systems representing unbounded or infinite systems for verification. Key to this is the ability to vary the coarseness of the abstraction by defining so called “instrumentation predicates” that can explicitly preserve specified properties of a state during abstraction. We observe that many of the instrumentation predicates defined previously in the literature record linear relationships between objects in a state, but it may be necessary to record geometric relationships, such as a group of objects forming a triangle shape or a square shape. To demonstrate the effectiveness of these “shape predicates”, we consider the problem of verifying that concurrent data structures are linearizable (Herlihy & Wing 1990) with respect to a sequential specification. Preserving the relationship between the implementation and specification data structures is tricky, and without shape predicates other authors (see Section 5) have had to invent other techniques to augment canonical abstraction.

The contributions of this paper are:

- Description of shape predicates, which have not

been used in the literature before, to our knowledge.

- Demonstration that linearizability can be verified using canonical abstraction alone.
- (Re)Verification of a stack and two queue algorithms using canonical abstraction with shape predicates.

The paper is structured as follows. Section 2 gives some background in two parts. In Section 2.1, an overview of canonical abstraction; in Section 2.2, a brief overview of concurrent data structures and the linearizability correctness condition, as well as a stack algorithm to be used as an example. Section 3 defines and explains the canonical abstraction model, including the shape predicates used to verify linearizability of the stack algorithm. Section 4 provides empirical results of verifying linearizability for the stack algorithm and two queue algorithms using canonical abstraction in the TVLA tool. Section 5 discusses related work. Finally, Section 6 concludes and discusses future work.

2 Background

2.1 Canonical Abstraction

Sagiv et al. (2002) represent states as logical structures, where predicates describe relationships between objects. Concrete states are represented using 2-valued structures. Abstract states are represented using 3-valued structures, which allow multiple concrete objects to be represented by a single abstract “summary object”. Since a summary object can represent two or more concrete objects, an abstract state with summary objects can represent an infinite number of concrete states.

First, a finite set of predicates $\mathcal{P} = \{\text{eq}, p_1, \dots, p_n\}$ is fixed for the analysis, and we define \mathcal{P}_k to be the set of k -ary predicates in \mathcal{P} (the equality predicate eq has arity 2). Then, a *concrete configuration* $S^\natural = \langle U^\natural, \iota^\natural \rangle$ has a *universe* U^\natural that is a (finite or infinite) set of objects and an *interpretation* ι^\natural over the logical values true (1) and false (0). For each k -ary predicate p ,

$$\iota^\natural(p) : (U^\natural)^k \rightarrow \{0, 1\}$$

Additionally, for each $u_1, u_2 \in U^\natural$ where $u_1 \neq u_2$, $\iota^\natural(\text{eq})(u_1, u_1) = 1$ and $\iota^\natural(\text{eq})(u_1, u_2) = 0$.

The definition of an *abstract configuration* $S = \langle U, \iota \rangle$ is similar to that of a concrete configuration, but the interpretation is over the truth values true (1),

Copyright ©2014, the authors. This paper appeared at the Thirty-Seventh Australasian Computer Science Conference (ACSC2014), Auckland, New Zealand, January 2014. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 147, Bruce H. Thomas and David Parry, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.

false (0) and unknown ($\frac{1}{2}$). For each k -ary predicate p ,

$$\iota(p) : U^k \rightarrow \{1, 0, \frac{1}{2}\}$$

Note that a concrete configuration is also trivially an abstract configuration. An object u , for which $\iota(\text{eq})(u, u)$ is unknown, is called a *summary object*.

Intuitively, an abstract configuration represents a concrete one if it contains the same information, except for some conservative information loss. In other words, it has the same universe of objects, though some may have been merged together into summary objects, and it has the same predicate interpretations, though some may have become unknown. This is formalised by the notion of embedding, which relates configurations (concrete or abstract¹) that are related by conservative information loss.

We say that a configuration $S_1 = \langle U_1, \iota_1 \rangle$ embeds into an abstract configuration $S_2 = \langle U_2, \iota_2 \rangle$ if there exists a surjective function $f : U_1 \rightarrow U_2$ such that for every k -ary predicate p , and $u_1, \dots, u_k \in U_1$,

$$\iota_1(p)(u_1, \dots, u_k) \sqsubseteq \iota_2(p)(f(u_1), \dots, f(u_k))$$

where, for $l_1, l_2 \in \{1, 0, \frac{1}{2}\}$, $l_1 \sqsubseteq l_2$ iff $l_1 = l_2$ or $l_2 = \frac{1}{2}$.

We further define a *tight embedding* to be one that minimises information loss, i.e. a predicate interpretation only becomes unknown if two objects are being merged together, one which has a true interpretation and the other a false interpretation. Formally, there exists a surjective function $f : U_1 \rightarrow U_2$ such that for every k -ary predicate p , and $u_1, \dots, u_k \in U_1$,

$$\iota_2(p)(u_1, \dots, u_k) = \begin{cases} 1 & \text{if } \forall u'_1 \in f^{-1}(u_1), \dots, u'_k \in f^{-1}(u_k) \bullet \\ & \quad \iota_1(p)(u'_1, \dots, u'_k) = 1 \\ 0 & \text{if } \forall u'_1 \in f^{-1}(u_1), \dots, u'_k \in f^{-1}(u_k) \bullet \\ & \quad \iota_1(p)(u'_1, \dots, u'_k) = 0 \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

Canonical abstraction is a method for constructing tight embeddings. Given a subset of the unary predicates $\mathcal{A} \subseteq \mathcal{P}_1$, called the *abstraction predicates*, we map objects in the original configuration to the same abstract object if they have the same interpretations over the abstraction predicates. The interpretation in the abstract configuration is constructed as per the definition of tight embeddings above. We say that a configuration is canonically abstract, with respect to \mathcal{A} , if it is the canonical abstraction of itself.

Canonical abstraction has a number of important properties:

- Every configuration has a single canonical abstraction, as each object has a single canonical mapping in the embedding function.
- Since there are a finite number of abstraction predicates, it follows that there is a finite bound on the number of objects in the universe of a canonically abstract configuration, and thus a finite bound on the number of potential states in an abstract system.

The soundness of the canonical abstraction approach rests upon the Embedding Theorem of Sagiv et al. (2002, Theorem 4.9). Informally, this says that if a structure S embeds into a structure S' , then

¹Since 2-valued configurations are trivially 3-valued configurations also, we will assume that configurations are 3-valued unless otherwise noted.

any information extracted from S' via a formula φ is a conservative approximation of the information extracted from S via φ . Alternatively, if we prove a property φ true or false in S' , then we know it has the same value in S .

The initial work of Sagiv et al. (2002) focused on sequential heap-manipulating programs, with a configuration universe representing the objects of the heap. This can be extended to represent concurrent programs, by including an object in the universe for each thread, and defining predicates to represent the threads' locations and fields (Yahav & Sagiv 2010).

2.1.1 Refining abstractions

Canonical abstraction using the fixed predicates \mathcal{P} is often too coarse, resulting in too much information being lost (i.e. evaluating to unknown) for a property to be verified. A key method for refining abstractions is to introduce additional predicates that record properties derived from the other predicates. These *instrumentation predicates* add no new information to a concrete state, since they evaluate to the same truth values as their defining formulas. However, in an abstract state they may add information: an instrumentation predicate may evaluate to a definite value (true or false) whilst its defining formula may evaluate to unknown. Additionally, unary instrumentation predicates may be added to the set of abstraction predicates, which can prevent some objects from being merged together into summary objects.

Defining instrumentation predicates to sufficiently refine the abstraction is the principal focus of Section 3.

2.2 Concurrent Data Structures

In this paper, we consider concurrent data structure algorithms (see e.g. Moir & Shavit 2004), which have multiple threads interacting with shared data, and synchronising access using locks or atomic primitives such as compare-and-swap (CAS).

A common correctness condition is linearizability (Herlihy & Wing 1990), which informally requires each operation to appear to take effect atomically at some point between its invocation and response. A system is linearizable, with respect to a given sequential specification, if the operations in any execution can be rearranged — respecting the ordering of non-concurrent operations — into an execution of the specification. One way of showing this is by determining “linearization points” for each operation, where the operation can be seen to take effect. If the specification is composed with the implementation and can perform a matching operation atomically at each linearization point then the implementation is linearizable.²

2.2.1 Example: Stack

Figure 1 gives the pseudocode for a linked list based stack algorithm. Each node of the list contains a value in the *val* field and a *next* field pointing to another node (or is *null*). A shared *Head* variable points to the first element when the stack is non-empty, and is *null* when the stack is empty. The algorithm assumes a garbage collector is present — popped nodes are not explicitly freed.

²In general this is more complicated, as an operation's linearization point may be a step of another operation, one step may be the linearization point for several operations, or a step may or may not be a linearization point depending on the future behaviour of other threads.

Type: Node = {*val* : T; *next* : Node}
Shared: *Head* : Node := null

```

1: operation PUSH(lv:T)
2:   n := new(Node)
3:   n.val := lv
4:   repeat
5:     ss := Head
6:     n.next := ss
7:   until CAS(Head, ss, n)
8: end operation
    
```

```

9: operation POP()
10:  repeat
11:    ss := Head
12:    if ss = null then
13:      return empty
14:    end if
15:    ss.next := ss.next
16:    lv := ss.val
17:  until CAS(Head, ss, ss.next)
18:  return lv
19: end operation
    
```

Figure 1: A lock-free stack algorithm

A push operation obtains a new node *n* and sets its value. It then takes a “snapshot” of *Head* and points *n*’s *next* field at the snapshot. A CAS operation is used to ensure that *Head* is updated to point to *n* only if it has not been modified. If *Head* has been modified then there has been a conflict with another (successful) operation so the loop is restarted.

A pop operation first takes a snapshot of *Head* and tests to see if the snapshot is *null*; if so it returns “empty”. Otherwise it takes a snapshot of this node’s *next* field and records the value in the *val* field. As for push, a CAS is used to detect a conflict with another successful operation — if *Head* has been modified it retries, otherwise it uses the snapshots to advance *Head* along the list.

This algorithm was first introduced by Treiber (1986) in IBM System/370 assembler. The version here assuming garbage collection follows that given by Colvin et al. (2005). Versions of the algorithm have been formally verified by several authors (including Colvin et al. 2005).

We can see that the algorithm is linearizable by determining the linearisation points of the operations:

- A push operation takes effect at line 7, when the CAS is successful.
- A non-empty pop operation takes effect at line 17, when the CAS is successful.
- An empty pop operation “takes effect” at line 11 when it reads a null *Head* value. The linearisation point is not at line 12 when the snapshot is tested, because *Head* may have been changed by other threads, so the stack cannot be guaranteed to be empty at that point in time.

For the first two, the successful CAS step is where the change of an added or removed node becomes observable to the other threads, and it is the trigger for leaving the loop, so cannot repeat. For the third, a *null* snapshot causes the thread to execute lines 12–13 and exit the loop (and operation), so the linearisation point cannot be repeated.

3 Verification of Stack

In order to attempt to verify linearizability for the concurrent stack algorithm in Section 2.2.1, we include an additional linked-list stack, which performs a Push or Pop operation atomically at the linearisation points of the implementation operation. If the implementation and specification operations always match, i.e. they always push and pop the same values, then the concurrent stack is linearizable. If they do not match, e.g. the specification Pop returns empty

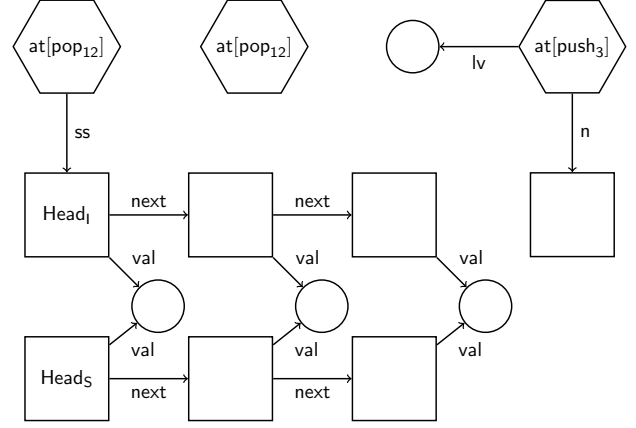


Figure 2: A potential concrete configuration

but the implementation Pop returns a value, then linearizability has not been shown.³

To represent this system for canonical abstraction, we define the set of predicates initially to contain unary predicates representing object types (*is_thread*, *is_node*, *is_data*), shared variables for the two stack lists (*Head₁*, *Head₅*) and thread locations (*at[loc]*, for *loc* ∈ {*idle*, *push₂*, *push₃*, ..., *pop₁₁*, *pop₁₂*, ...}). Additionally, we have binary predicates representing the fields of the nodes (*next*, *val*) and the threads (*n*, *lv*, *ss*, *ssnext*).

For clearer explanations, we will describe states diagrammatically, rather than logically. We use different object shapes to represent the type predicates — hexagons for threads, squares for nodes, and circles for data values. Unary predicates are shown as labels on objects when true, binary predicates are shown as arrows (solid for true, dotted for unknown, not shown for false), and summary objects have a double line. Figure 2 shows a potential concrete configuration, and Figure 3 its canonical abstraction.

The two stack lists have length three, with three distinct data values. One of the three threads has just begun a push operation; the other two have just begun pop operations, though one has a current snapshot of the *Head₁* and the other has a stale null snapshot taken when the stack was empty.

As is common, the canonical abstraction on core predicates alone is too coarse. For example, we cannot distinguish between the nodes of the two different lists, nor those from the nodes not yet pushed, and cannot tell whether a thread has a null or non-null field. This means that abstraction of some reachable

³The algorithm may not be linearizable, or it may be linearizable but we have chosen incorrect linearization points. Determining which is the case is outside the scope of this paper.

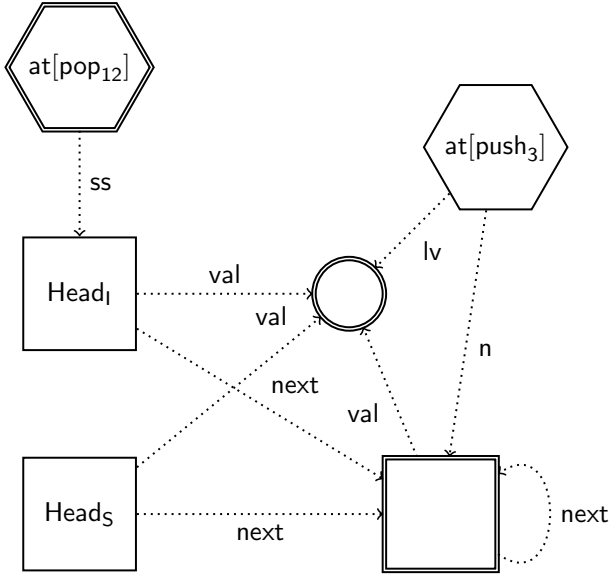


Figure 3: Canonical abstraction of potential configuration

states will also represent some nonreachable states, which will lead to executions with spurious errors detected. We can refine the abstraction by defining instrumentation predicates, of forms previously used by other authors (e.g. Sagiv et al. 2002, Yahav & Sagiv 2010):⁴

$$\begin{aligned}
 \text{has}[\text{field}](v) &\Leftrightarrow \exists u \bullet \text{field}(v, u) \\
 \text{r_by}[n](v) &\Leftrightarrow \exists u \bullet n(u, v) \\
 \text{shared}[n](v) &\Leftrightarrow \exists u_1, u_2 \bullet n(u_1, v) \wedge n(u_2, v) \\
 &\quad \wedge \neg \text{eq}(u_1, u_2) \\
 \text{circ}(v) &\Leftrightarrow \text{next}^+(v, v) \\
 \text{reach}_I(v) &\Leftrightarrow \exists u \bullet \text{Head}_I(u) \wedge \text{next}^*(u, v) \\
 \text{reach}_S(v) &\Leftrightarrow \exists u \bullet \text{Head}_S(u) \wedge \text{next}^*(u, v)
 \end{aligned}$$

where p^+ is the transitive closure of p , and p^* is the reflexive transitive closure. These instrumentation predicates allow more information to be preserved in the abstract states by, e.g. preventing the two list bodies from being merged together, and recording that each is connected and acyclic.

The two lists should have the same data values, in the same order. However, when the tails of the lists are abstracted to summary objects, this property is lost. In order to specify properties of the pair of i th nodes in the two lists, we introduce an auxiliary core binary predicate R to relate them. R is set between the head nodes of the lists by the specification Push operation, and is unset for the head nodes by the specification Pop operation.

Even with the auxiliary predicate, the instrumentation predicates defined above are not sufficient to preserve all the properties we need about the lists and the threads' fields. We observe that these predicates all define linear properties — $\text{has}[\text{field}]$ and $\text{r_by}[n]$ describe two objects related by one predicate; $\text{shared}[n]$ describes three objects related by two pred-

icates; reach_I describes an arbitrary number of objects related by a chain of predicates; circ describes the same, but the chain begins and ends with the same object. We defined three additional instrumentation predicates that describe geometric shapes relating three or four objects.

3.1 Matching triangle predicate

Consider Figure 5, which shows the abstraction of (the lists of) two states where the implementation and specification stacks have length 3 — in S_1^I the lists have the same values in the same order, and in S_2^I the lists' head values differ. Both states have the same canonical abstraction and the information about the values is lost.

In order to preserve the property that each corresponding pair of nodes in the lists have the same data value, we define an instrumentation predicate, called **matching**:

$$\begin{aligned}
 \text{matching}(n_1) &\Leftrightarrow \exists n_2, d_1 \bullet \\
 &\quad R(n_1, n_2) \wedge \text{val}(n_1, d_1) \wedge \text{val}(n_2, d_1)
 \end{aligned}$$

The predicate records a “triangular” relationship between nodes and data values, as shown in the first diagram in Figure 4.

Adding this instrumentation predicate to the concrete states in Figure 5 results in different canonically abstract states. Both would differ from S_1 as the implementation list summary node would be labelled with **matching**; and both would differ from each other as one would have the head implementation node labelled with **matching** and the other would not.

3.2 Commutes square predicate

Consider the two concrete states in Figure 6 — they both have three elements, and **matching** is true for all the implementation list nodes. In S_4^I , the R relations have “crossed”, so after a Pop operation the head nodes will have different values — another Pop from both lists will trigger a linearizability error. We see that these two states have the same canonical abstraction, so analysis of a linearizable stack can still provide spurious errors.

In order to preserve the property that related pairs of nodes have the same order in both lists, we define an instrumentation predicate that records whether the **next** and R predicates “commute”:

$$\begin{aligned}
 \text{commutes}(n_1) &\Leftrightarrow \exists n_2, n_3, n_4 \bullet \\
 &\quad \text{next}(n_1, n_2) \wedge R(n_1, n_3) \wedge \\
 &\quad \text{next}(n_3, n_4) \wedge R(n_2, n_4)
 \end{aligned}$$

The predicate records a “square” relationship between nodes, as shown in the second diagram in Figure 4.

Adding this instrumentation predicate to the concrete states in Figure 6 results in different canonically abstract states. The first two implementation nodes in S_3^I are labelled with **commutes**; then since all three implementation nodes have different abstraction predicate labels none of them will be merged in to summary objects in the canonical abstraction. The diagrams for the other concrete state and its canonical abstraction are identical to S_4^I and S_3 , as **commutes** is false for all of the implementation nodes.

Together, **matching** and **commutes** preserve sufficient information about the two lists to allow linearizability to be verified.

⁴The square brackets have no meaning other than being a visual indicator of which core predicates are used in the definition. (TVLA allows parametrised definitions of sets of predicates in this way — e.g. to define $\text{reach}[y, \text{next}]$ and $\text{reach}[z, \text{next}]$ at the same time.)

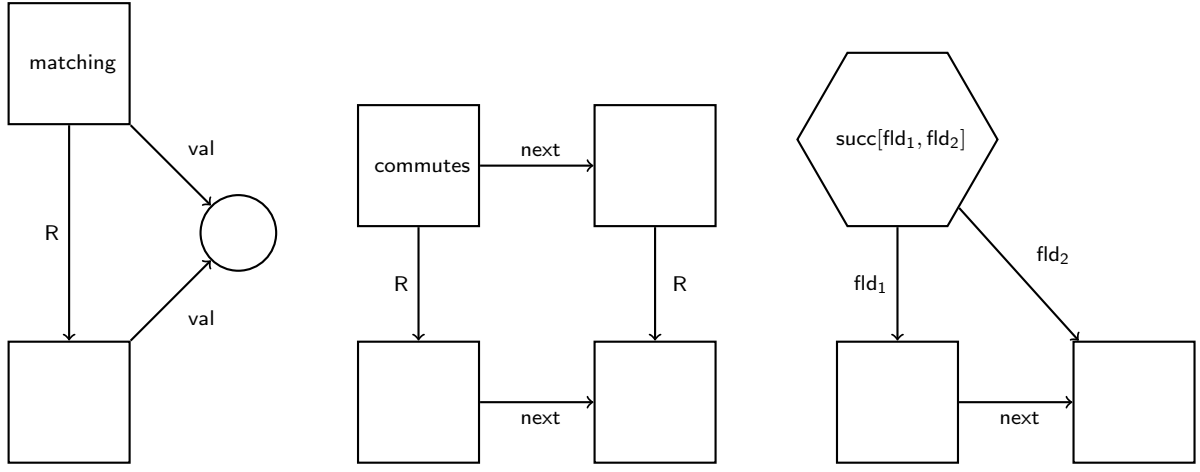


Figure 4: Three shape predicate diagrams

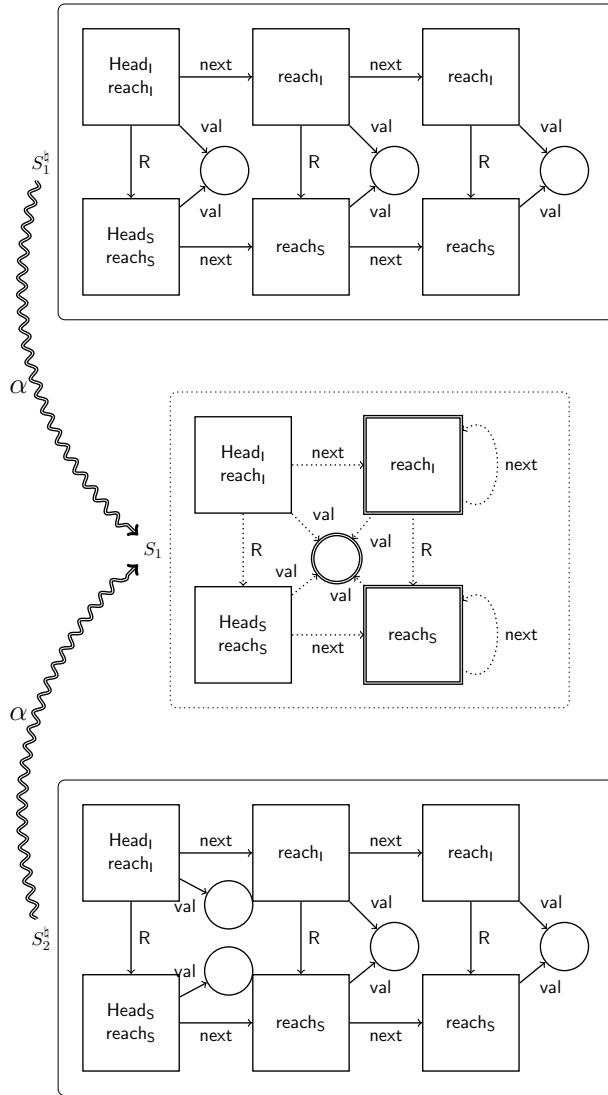


Figure 5: Canonical abstraction of two lists: the property of matching values is lost

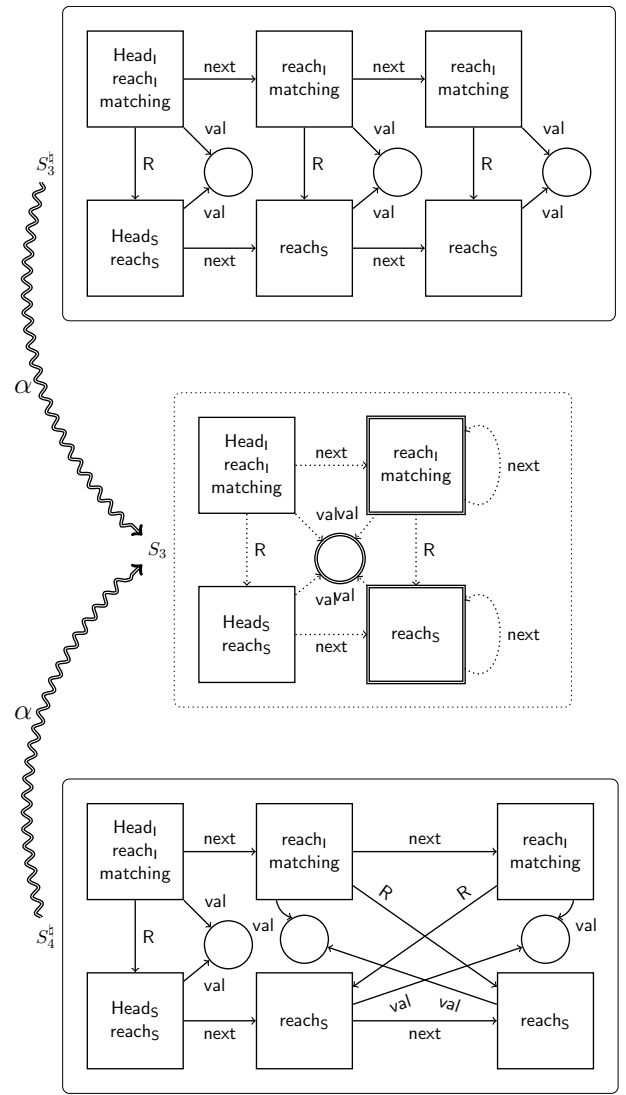


Figure 6: Canonical abstraction with “crossed” R predicates: the property of ordered values is lost

3.3 Successor triangle predicate

In the stack's Pop operation, the `ssnext` field is the next-successor of the `ss` field when it is read at line 15. This property is assumed to persist, so it is not checked before the CAS step at line 17 that attempts to set `Headi` to `ssnext`.

Figure 7 shows that this property is not retained in canonical abstraction using the predicates defined so far. Both states (shown without the data values and specification lists) have two threads performing a Pop operation — in S_5^h , both `ssnext` predicates are the next-successors of the respective `ss` predicates, but this is not the case in S_6^h ; nevertheless, both states have the same canonical abstraction (S_5). As a consequence, the CAS transition can remove an arbitrary prefix of the list because `ssnext` can be concretised at any point.

In order to preserve the relationship between the thread fields, we define an instrumentation predicate that records whether they are next-successors:

$$\text{succ}[\text{ss}, \text{ssnext}](t_1) \Leftrightarrow \exists n_1, n_2 \bullet \\ \text{ss}(t_1, n_1) \wedge \text{ssnext}(t_1, n_2) \wedge \text{next}(n_1, n_2)$$

This predicate records a “triangular” relationship between threads and nodes, as shown in the third diagram in Figure 4.

Adding this instrumentation predicate to the concrete states in Figure 7 results in different canonically abstract states. The diagrams are almost the same, but are distinguished by whether the thread summary object has is labelled with `succ[ss, ssnext]` or not.

A similar situation arises in the Push operation. The `ss` predicate is set to be the next-successor of the `n` predicate at line 6, which is assumed to be unchanged at the CAS step that sets `Headi` to `n` at line 7. Thus we similarly define the instrumentation predicate `succ[n, ss]`.

For space restrictions, we omit further discussion on the basic model constructs, notably details about transitions. A complete presentation can be found in Friggens (2013, Chapter 7).

4 Empirical Results

To perform analyses and gather empirical results, we used TVLA⁵ (Three Valued Logic Analyzer) (Lev-Ami & Sagiv 2000, Bogudlov et al. 2007), a prototype static analysis tool developed at Tel Aviv University that implements canonical abstraction.

4.1 Stack

We analysed thread-bounded and unbounded models of the stack algorithm using TVLA 3.0 α on a machine with an Intel Core 2 3.0 GHz processor and 4 GB of RAM, running Java 1.6.0 on a 32-bit GNU/Linux operating system. By default, TVLA can construct models with one thread or an unbounded number of threads, depending on whether the initial configuration has a summary or non-summary idle thread object. To obtain models with some other bounded number of threads, we defined compatibility constraints (Sagiv et al. 2002, Section 6.4.2) that discard any configuration that satisfy a formula identifying $n + 1$ or more distinct thread objects. For example,

⁵<http://www.cs.tau.ac.il/~tvla/>

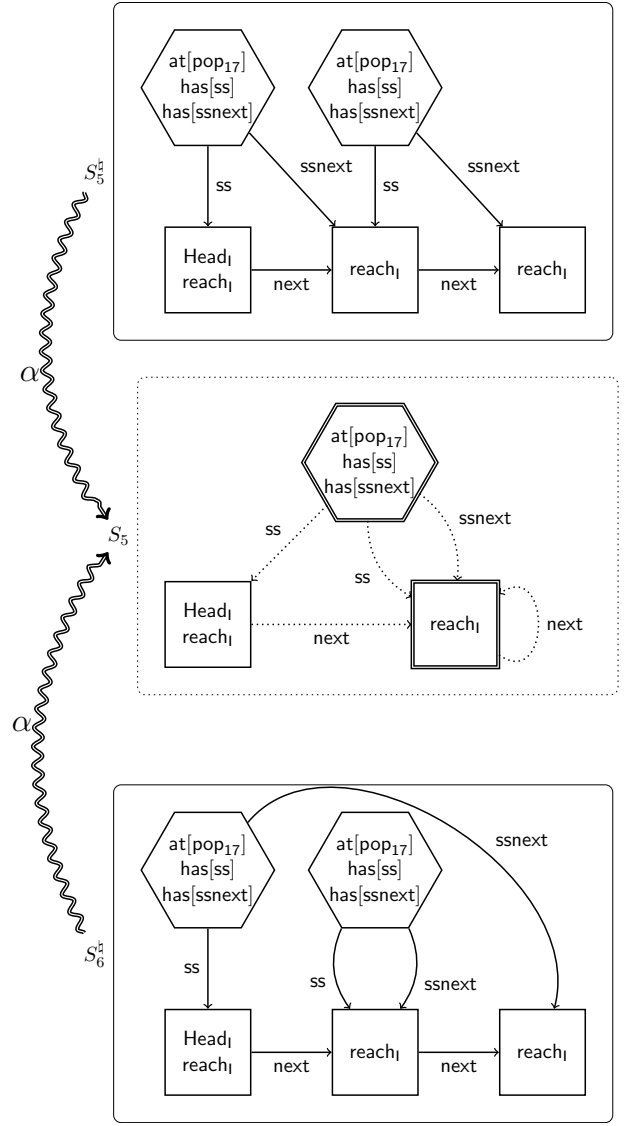


Figure 7: Canonical abstraction of threads: the relationships between fields' values are lost

bounding to two threads:

$$\exists t_1, t_2, t_3 \bullet \\ \text{is_thread}(t_1) \wedge \text{is_thread}(t_2) \wedge \text{is_thread}(t_3) \\ \wedge \neg \text{eq}(t_1, t_2) \wedge \neg \text{eq}(t_2, t_3) \wedge \neg \text{eq}(t_1, t_3)$$

Table 1 contains results of verifying linearizability with unbounded lists and data values, and up to three threads. Time, memory and statespace figures are as reported by TVLA. With four threads, the model was too large and TVLA ran out of memory.

Since TVLA does not implement techniques such as partial order reduction (Peled 1998) to reduce unnecessary interleavings of threads, we manually modified the models to restrict interleaving of transitions that only read or write to local variables. Table 2 contains results of these models, which appear to have an exponential reduction in statespace. The principal result is that linearizability of the stack algorithm is able to be verified for unbounded numbers of threads and data values, and for lists of unbounded length, using only canonical abstraction.

We note that the time for the analyses of bounded models increases exponentially - the general approach taken by TVLA of evaluating our bounding formulas becomes increasingly impractical as the number of

Th.	Heap Limit (MB)	Time (s)	Ave RAM (MB)	Max RAM (MB)	Stored States
1	800	2	16	38	148
2	800	88	178	335	7,731
3	2,048	2,931	1,089	1,946	148,191
4	2,048	—	—	—	>282,441

Table 1: Stack verification results with full interleaving

Th.	Heap Limit (MB)	Time (s)	Ave RAM (MB)	Max RAM (MB)	Stored States
1	800	1	4	6	86
2	800	18	94	252	1,312
3	800	102	173	336	6,493
4	800	535	255	479	18,564
5	2,048	6,409	502	1,184	36,749
6	2,048	143,302	480	1,052	55,069
7	4,096	2,625,113	1,182	2,633	67,334
∞	1,024	6,524	647	1,057	74,056
∞	2,048	1,934	849	1,603	74,056

Table 2: Stack verification results with restricted interleaving

thread objects being identified, and thus the length of the formula, increases. It may well be possible to implement a more direct and efficient way in TVLA for limiting numbers of specific objects; if so, it would make verifying models with bounds of greater than six threads practical.

4.2 Queues

We additionally analyzed linearizability for two non-blocking queue data structures, the original due to Michael & Scott (1998). Doherty et al. (2004) give a variation with a simplified dequeue operation; they also provide a formal verification using a theorem prover.

The canonical abstraction models are constructed similarly to the stack models, with similar shape predicates — full details are available in Friggens (2013, Section 7.9).

Table 3 contains results, using the same software and hardware as for the stack. To reduce the statespace we again added manual restrictions to interleaving for steps that only read and write to local variables. For both algorithms we verify linearizability for one or two threads, unbounded numbers of data values and lists of unbounded length. For the models with three threads, the model is too large and TVLA runs out of memory.

Deq	Th.	Heap Limit (MB)	Time (s)	Ave RAM (MB)	Max RAM (MB)	Stored States
MS	1	800	1	13	30	115
MS	2	800	393	260	476	24,271
MS	3	2,048	—	—	—	>235k
Doh	1	800	1	14	33	117
Doh	2	800	83	189	354	10,746
Doh	3	2,048	—	—	—	>230k

Table 3: Queue verification results

5 Related Work

The closest work to ours is by Amit et al. (2007), who analysed the same nonblocking data structures (plus two lock-based data structures). They also restricted interleaving of threads manually, and were able to verify linearizability for the stack algorithm with three threads and the queue algorithms with two threads (limiting to 1.5 GB of RAM). They combine canonical abstraction with an additional approach called “delta heap abstraction”: the relationship between each pair of implementation and specification nodes and their identical value is represented in the state graph by a single object. Delta heap abstraction requires each push/enqueue etc. to be for a unique value, whereas our approach can represent data values being entered into the list multiple times. Their analyses use unique predicates to distinguish each thread and its field values; this is exponentially more expensive than using the shape predicates we have defined, and does not allow unbounded numbers of threads to be considered.

This approach is made more efficient by Manevich et al. (2008), who combine canonical abstraction with heap decomposition. Heap decomposition splits the state into (overlapping) subgraphs and only stores one copy of a subgraph no matter how many states it appears in. They were able to verify linearizability for the stack algorithm with 20 threads (limiting to 2 GB of RAM), and for the second queue algorithm with 15 threads (limiting to 16 GB of RAM).

Berdine et al. (2008) combine the above approaches with an additional approach called “quantified canonical abstraction” to verify linearizability for unbounded threads. Like heap decomposition, the approach splits the state into (overlapping) subgraphs, each containing the data structure and one non-summary thread. Unlike heap decomposition, each subgraph can represent an unbounded number of identical subgraphs, thus the bounded number of subgraphs together can represent states with unbounded numbers of threads. Extending the models of Amit et al. (2007), and limiting to 2 GB of RAM, Berdine et al. (2008) were able to verify linearizability for the stack algorithm, but ran out of memory for the queue. Extending the models of Manevich et al. (2008), using heap decomposition to create smaller subgraphs, they were able to verify linearizability for both the stack algorithm (with an 80% reduction in statespace) and queue algorithm. This is the first published work to verify linearizability for unbounded threads using canonical abstraction, though it uses two additional approaches to do so.

6 Conclusions and Further Work

In this paper we have introduced shape predicates, a type of instrumentation predicate for refining canonical abstractions. Though defining triangles and squares may seem obvious in hindsight, these predicates have not been used before in the canonical abstraction literature and can prove to be powerful in constructing an appropriate abstraction. They will almost certainly be of use in a wide range of canonical abstraction applications.

We have demonstrated the utility of shape predicates by verifying linearizability for three concurrent data structure algorithms. In doing so we have demonstrated the interesting theoretical result that verification of linearizability is possible with canonical abstraction alone, and does not require delta heap abstraction or thread quantification.

The abstract models that are constructed for the

stack and two queue algorithms are finite, but still very large. Restricting the interleaving of local steps, we were able to completely verify the stack algorithm. However, for the un-restricted stack and for the restricted queues, the analyses ran out of memory for models with four or more threads. The principal problem is the exponential permutations of thread objects and list configurations. One approach to improving the performance would be to employ heap decomposition (Manevich et al. 2008) or thread quantification (Berdine et al. 2008), with which (an extension to) TVLA decomposes each state into list and thread components, storing each only once, no matter how many states the component appears in. An alternative approach would be to collapse all of the thread objects in a state into a single summary object, defining “soft invariant” instrumentation predicates (Friggens & Groves 2013) to preserve properties of the threads that would be lost otherwise.

Finally, we would like to extend the verifications of the stack and queue algorithms to other concurrent data structures. Some data structures, such as deques, have a similar property of having a close correspondence between the implementation and specification data structures, so a similar approach would be reasonable to expect. Other data structures, such as elimination stacks and sets have more difference between the implementation and specification data structures, so more ingenuity in the model construction may be required.

References

- Amit, D., Rinetzkly, N., Repts, T., Sagiv, M. & Yahav, E. (2007), Comparison under abstraction for verifying linearizability, in W. Damm & H. Hermanns, eds, ‘Proceedings of the 19th International Conference on Computer Aided Verification (CAV)’, Vol. 4590 of *Lecture Notes in Computer Science*, Springer, pp. 477–490.
- Berdine, J., Lev-Ami, T., Manevich, R., Ramalingam, G. & Sagiv, M. (2008), Thread quantification for concurrent shape analysis, in A. Gupta & S. Malik, eds, ‘Proceedings of the 20th International Conference on Computer Aided Verification (CAV)’, Vol. 5123 of *Lecture Notes in Computer Science*, Springer, pp. 399–413.
- Bogudlov, I., Lev-Ami, T., Repts, T. & Sagiv, M. (2007), Revamping TVLA: Making parametric shape analysis competitive, in W. Damm & H. Hermanns, eds, ‘Proceedings of the 19th International Conference on Computer Aided Verification (CAV)’, Vol. 4590 of *Lecture Notes in Computer Science*, Springer, pp. 221–225.
- Colvin, R., Doherty, S. & Groves, L. (2005), Verifying concurrent data structures by simulation, in J. Derrick & E. A. Boiten, eds, ‘Proceedings of the Refinement Workshop’, Vol. 137.2 of *Electronic Notes in Theoretical Computer Science*, Elsevier, pp. 93–110.
- Doherty, S., Groves, L., Luchangco, V. & Moir, M. (2004), Formal verification of a practical lock-free queue algorithm, in D. de Frutos-Escrig & M. Núñez, eds, ‘Proceedings of the 24th International Conference on Formal Techniques for Networked and Distributed Systems (FORTE)’, Vol. 3235 of *Lecture Notes in Computer Science*, Springer, pp. 97–114.
- Friggens, D. (2013), On the Use of Model Checking for the Bounded and Unbounded Verification of Non-blocking Concurrent Data Structures, Ph.D. thesis, Victoria University of Wellington.
- Friggens, D. & Groves, L. (2013), ‘Collapsing threads safely using soft invariants’, In preparation.
- Herlihy, M. P. & Wing, J. M. (1990), ‘Linearizability: A correctness condition for concurrent objects’, *ACM Transactions on Programming Languages and Systems* **12**(3), 463–492.
- Lev-Ami, T. & Sagiv, M. (2000), TVLA: A system for implementing static analyses, in J. Palsberg, ed., ‘Proceedings of the 7th International Symposium on Static Analysis (SAS)’, Vol. 1824 of *Lecture Notes in Computer Science*, Springer, pp. 280–301.
- Manevich, R., Lev-Ami, T., Sagiv, M., Ramalingam, G. & Berdine, J. (2008), Heap decomposition for concurrent shape analysis, in M. Alpuente & G. Vidal, eds, ‘Proceedings of the 15th International Symposium on Static Analysis (SAS)’, Vol. 5079 of *Lecture Notes in Computer Science*, Springer, pp. 363–377.
- Michael, M. M. & Scott, M. L. (1998), ‘Non-blocking algorithms and preemption-safe locking on multiprogrammed shared memory multiprocessors’, *Journal of Parallel and Distributed Computing* **51**(1), 1–26.
- Moir, M. & Shavit, N. N. (2004), Concurrent data structures, in D. P. Mehta & S. Sahni, eds, ‘Handbook of Data Structures and Applications’, Chapman and Hall/CRC, chapter 47.
- Peled, D. A. (1998), Ten years of partial order reduction, in A. J. Hu & M. Y. Vardi, eds, ‘Proceedings of the 10th International Conference on Computer Aided Verification (CAV)’, Vol. 1427 of *Lecture Notes in Computer Science*, Springer, pp. 17–28.
- Sagiv, M., Repts, T. & Wilhelm, R. (2002), ‘Parametric shape analysis via 3-valued logic’, *ACM Transactions on Programming Languages and Systems* **24**(3), 217–298.
- Treiber, R. K. (1986), Systems programming: Coping with parallelism, Technical Report RJ 5118, IBM Almaden Research Centre.
- Yahav, E. & Sagiv, M. (2010), ‘Verifying safety properties of concurrent heap-manipulating programs’, *ACM Transactions on Programming Languages and Systems* **32**(5), Article 18.

Document DNA: Content Centric Provenance Data Tracking in Documents

Michael Rinck

The University of Waikato
University of Waikato
Gate 1 Knighton Road
Private Bag 3105,
Hamilton 3240, New Zealand
mr97@waikato.ac.nz

Annika Hinze

The University of Waikato
University of Waikato
Gate 1 Knighton Road
Private Bag 3105,
Hamilton 3240, New Zealand
hinze@cs.waikato.ac.nz

David Bainbridge

The University of Waikato
University of Waikato
Gate 1 Knighton Road
Private Bag 3105,
Hamilton 3240, New Zealand
davidb@cs.waikato.ac.nz

Steve Jones

The University of Waikato
University of Waikato
Gate 1 Knighton Road
Private Bag 3105,
Hamilton 3240, New Zealand
stevej@cs.waikato.ac.nz

ABSTRACT

This paper presents a new content centric approach to provenance data tracking: Document DNA. We present the results and analysis of our exploratory study on the re-use and re-finding of content contained in digital documents. The study's results support our content-centric approach, as users have difficulties on keeping track of re-used content. The Document DNA presented in this paper is a distributed approach that tracks content when it is copy pasted in between documents by attaching a signature to the content. This signature evolves according to the changes made to the content, therefore allowing for tracking the changes made to content. By choosing a distributed approach, we achieve independence of central management systems. Since the Document DNA is adapted on the fly, we do not need post action analysis, making our approach very accurate. This paper includes a detailed description of the initial study, the theoretical concept and finally the prototype which implements this concept as a Microsoft Word™ add-in.

Author Keywords

Content tracking; Content evolution; Versioning; Content Management

ACM Classification Keywords

Copyright © 2014, Australian Computer Society, Inc. This paper appeared at the Thirty-Seventh Australasian Computer Science Conference, ACSC 2014, Auckland, New Zealand, January 2014. Conferences in Research and Practice in Information Technology, Vol. 147. Bruce H. Thomas and David Parry, Eds. Reproduction for academic, not-for-profit purposes permitted provided this text is included.

H.3.2 Information Storage and Retrieval: Information Storage
H.3.1 Information Storage and Retrieval: Content Analysis and Indexing

1 INTRODUCTION

In 1995, Barreau et al. [1] were the first to conduct a study on user's habits when organizing documents and when searching for documents stored on their PC. They found that at that time, people were content with the way the File/Folder structure worked. However, Vannevar Bush identified many limitations of the file/folder structure [2], and other researchers (for example Fertig et al.[7]) began to question the efficiency of the system, in particular the large amount of time spent on searching for files. In 1996, Fertig et al. [7] introduced the first of many approaches to improve the File/Folder system. Their Lifestreams system would store every file a user received or created in an ongoing timeline, which users could then browse.

However, 17 years later, the File/Folder system is still the predominant way of organizing digital documents. At the surface level, it seems that Barreau et al. were correct in their observation that the system is adequate and that more efficient ways to handle their documents are neither wanted nor needed.

However, we believe that much has changed since the original study was done and that it is time to rethink document management.

1. The first factor that has changed is the increased digital storage available to document users. Moreover, the significance of digital files in our lives has increased as well. It is therefore reasonable to assume that the amount of digital data owned by each person has also increased significantly.
2. The second factor is the increased connectivity that users are able to access. With affordable portable devices and mobile

internet connections, collaboration on digital documents is possible virtually anytime anywhere.

3. The third and final factor is the huge diversity of places in which to store and retrieve documents. The so-called cloud may include such diverse storage places as: home PCs, smart phones, work PCs, tablets and online storage solutions. The tracking of documents “in the cloud” is the aim of current research done at HP labs [13], resulting in a system called the TrustCloud.

These three factors indicate to us that there is a need for a better system of organizing and finding digital documents. This argument is supported by the fact that a large number of metadata annotation systems have been developed since Barreau’s first study. These systems aim to supply the user with an alternative structure of document organization to use for searching and sorting their files. Section 2 gives an overview of metadata annotation systems and an analysis of why these systems were not widely used.

This paper presents the findings of an exploratory study on how people manage document versions; especially on how they re-use and re-find content spread over many documents. We analyze the results of a series of interviews that confirm and substantiate our critique of the existing File/Folder system. The study shows that documents almost always contain re-used content and that users find it difficult to keep track of that content (see Section 3).

We present a new distributed approach to track content as it is copied from one document to another document by attaching a signature directly to the content (Section 4). This approach also tracks the evolution of the content, by adapting the signature according to the changes made to the content. This signature is called Document DNA, or DDNA. Our approach aims to be independent of central management systems and post-action analysis.

We conclude this paper with a summary of our work and a discussion of what needs to be done in the future.

2 DOCUMENT ANNOTATION SYSTEMS

Metadata annotation systems are the predominant way of targeting the limitations of the File/Folder structure. This approach lets users preserve a familiar environment for their documents. It also incorporates an additional dimension, the annotated metadata, which assists users when organizing and searching for documents. Document annotation systems can be divided into automatic and manual approaches. As reviewing every annotation system would go beyond the scope of this paper, we have therefore selected a number of examples to highlight the advantages and disadvantages of existing approaches.

Manual annotation

Manual annotation systems such as Desktop+ [6], VennFS [3], Archosum [8] and pStore [23] aim to capture semantic information about documents, mostly described as categories. Those categories may be represented by different objects (e.g., Venn shapes (VennFS), piles (Desktop+) or meta-objects (Archosum)). pStore even introduces a completely open framework where categories can take any form. Manual annotation systems require the user to allocate categories to files, since the detection of semantic information about documents is a process that is still prone to error.

We believe manual annotation to be unreliable as it requires the users to allocate metadata to files. However such manual annotation is itself unreliable as it depends on the user’s administrative ability and diligence. For example, users may be unable or unwilling to keep track of all annotations needed for their documents. On the other hand manual systems have the advantage of storing information that might be meaningful to the user. In particular, Xu’s [22] idea of storing each version as a new document is promising, as this allows the evolution of document information to be tracked.

Automatic annotation

Automatic annotation systems like Omnistore [10] or the Know How Sharing Agent [16] aim to collect meta-data and annotate documents automatically.

Omnistore manages documents that are distributed in a personal area network consisting of many small and mobile devices. Information about user activity is then used to annotate documents that reside on the devices. Svensson [20] observed that most automatic annotation systems log data on location, time or activity in order to provide context (as done in Omnistore). Soules et al. [20] pointed out that more than just sensor data is needed for useful annotations, as sensor data is often meaningless when determining a document’s context. Satoh et al. [16] aimed to collect more meaningful data by logging the transfer of content between websites, emails and documents (via the infrastructure of the programs used). However, this work was limited to certain software, e.g. Lotus Notes or Netscape.

A similar approach is the TrustCloud [13] by Ko et al. (HP labs), especially with Flogger integration. The Flogger (a file-centric logger) tracks and stores all file accessions and movements to ensure that the users know of every instance of their files in the cloud at all times. It does so by detecting events on files and storing the resulting information in logs kept on servers in the cloud. It does not detect any information transfer between files. A similar, but decentralized approach to annotating files is Digital Rights watermarking. Kim et al. [12] developed a watermarking algorithm that works on text documents. Their approach uses inter-word spaces and word segmenting to hide a watermark in the text document. This means that the pattern of the spacing between word groups is itself coded information. This approach has the disadvantage that the applied metadata is static and does not adapt to content changes in the documents.

Dragunov et al. [5] developed the Task Tracer, a system to automatically detect tasks and classify documents according to the tasks to which they belong. Task Tracer keeps track of copy and paste operations on the operating system level, using this information to determine connections between files. The main disadvantage of Task Tracer is its dependence on a central database to store the information. It also lacks any tools to determine what parts of files have been copy pasted, or how the documents were subsequently changed. Finally, Jensen et al. conducted a user study showing that copy and paste is the most common provenance link between files [9], indicating that a new metadata annotation system would benefit from focussing on this information.

Observations

We found that manual and automatic annotation systems alike share one major disadvantage: they depend on a central management component (such as a database) to administer the

document annotations. The annotations are stored separately and not directly in or with the documents. This makes the annotations useless as soon as the documents are stored in a place not connected to this central component. It also forces all users who collaborate on documents to install the same annotation system, as other software is unlikely to understand the format of the annotations. Following this general observation and taking the other observed disadvantages into account, we propose the following requirements for our annotation system:

1. The annotation must be directly stored in or with the document.
2. The annotation must be adapted automatically when the document is created or manipulated.
3. The annotation must not point out false relations, or miss correct relations.
4. The format of the annotations must be open and easily reusable.
5. The annotation has to enable the user to track where instances of their content have been used.
6. The annotation has to enable the user to determine how their content evolved from previous versions, i.e., which are the previous versions and in what way do they differ.

3 EXPLORATORY STUDY

Our user study had two goals. Firstly, to explore what issues current users of digital documents have with the File/Folder system. Secondly, we aimed to verify if the requirements we defined after our review of existing annotation systems would be sufficient to solve those issues. We therefore conducted a series of 15 minute interviews at the workplaces of our participants. We discussed their ways of managing documents and learned about the issues they had in their everyday work processes.

We chose our participants from University staff and a local law firm. The only requirement was that the participants needed to have five or more years of experience handling digital documents. We also wanted to have a diverse group of participants and therefore selected participants from different areas of work. This resulted in the participation of seven academics (computer science), four staffmembers concerned with administrative or management tasks, three staffmembers working at a library, four language teachers, one lawyer and one legal secretary.

We prepared a number of questions but were also open to following up on anything interesting we would discover while talking to the participants. We also kept track of statistical information such as age, gender and profession of our participants. Our interviews were structured around the following questions:

1. How many years have you been using digital documents in a professional environment?
2. What is your current most used document editor?
3. How often do you re-use digital content, on a scale from 1 (never) to 5 (very often)?
4. How often do you need to find other instances of re-used content, on a scale from 1 (never) to 5 (very often)?
5. How do you organize your documents?

6. What problems do you encounter in the organization of your documents?

The interviews were held at the workplaces of the participants where possible, so that they could demonstrate to the interviewer how they managed their documents.

Results

Our study had eight male and twelve female participants. One participant was between 20 and 29 years old, six were in the age group of 30-39, eight in the age group of 40-49 and five in the age group of 50-59 years.

Digital Documents and Editors.

The participants had been using digital documents for five to thirty years (average 17.25, median 20). Most participants could not name a single editor they used most but rather named a range of editors they use every day. Figure 1 shows the results: Microsoft Word was the favored document editor for most participants. However, eight participants who mentioned Microsoft Word also mentioned another document editor. These results also indicate that most document content is text.

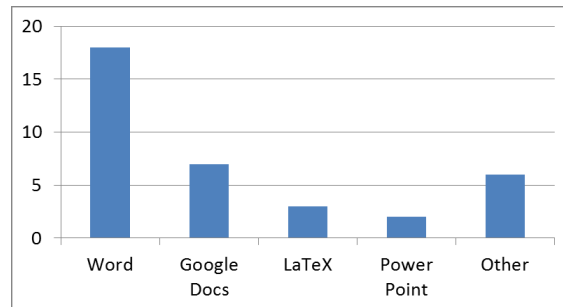


Figure 1: Document editors used by the 20 participants (multiple answers allowed)

Reusing Content.

The interviewer defined content as being re-used if it was taken by the participant from another digital source. We explained that a digital document could be any digital file whose contents were accessed by the participant. No distinction was made between content that was changed when re-used, or content that was re-used in unchanged form. We also included the re-use of complete documents for other purposes in this definition.

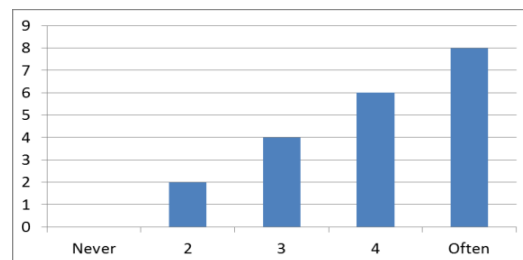


Figure 2: Frequency of content re-use

Participants' answers about reusing content were recorded on a Likert scale from one to five, where 1 meant "never" and 5 meant "very often" (see Figure 2 for results). Two participants chose a

position in-between 3 and 4. We attributed one to 3 and one to 4. None of the participants had never re-used documents or document parts. The majority of participants re-used documents often and very often (4 and 5). The average of all the answers is 4 (often).

The answers for how often participants needed to find other instances of re-used content had greater variation (see Figure 3). We again chose a Likert scale to record the participants' answers. Five participants chose to answer in-between numbers (e.g., 3.5), so we split their vote for the figure (0.5 on 3 and 0.5 on 4). The average of all the answers is 3.225.

For some participants, the re-use (creating or finding) depended on the context of the documents. For example, academics stated that they very rarely re-use or try to find re-used teaching content, but had the opposite experience with academic content.

Experiences with File/Folder System.

Unsurprisingly, the File/Folder system was still used by all 20 participants. However, 18 of the participants noted issues with the File/Folder system. This was expressed either in the direct statement that their File/Folder system fails when they try to find documents they stored, or it was mentioned indirectly in reference to issues resulting from the limitations of the File/Folder system, for example the lack of versioning or synchronization.

When they mentioned the File/Folder system, we also inquired how participants used it. The file and folder names were always very important to the participants. One participant even used the file names for exact versioning. This participant's file names included dates, version numbers and purposes.

The depth of the File/Folder system used by our participants ranged from zero, meaning just the Desktop, to 5 or more. Exact numbers were difficult to get in these instances, as users are usually not aware which of their self-created folders is the deepest. Interestingly, both the participants with low-depth file systems and the participants with high-depth file systems claimed to have difficulty locating files. We also had one example each of low- and high-depth folder systems which presented no issues for their respective users. However, every participant stated that the maintenance of their File/Folder system required a considerable amount of work that they would rather spend on different tasks. In one of our interviewed work environments, a separate document was kept to record locations of other documents.

The second observation we made was the sheer number of systems used to keep files and to organize them. Figure 4 shows the number of places in which participants keep documents. Only one participant kept all their files in one place, and 11 of the 20 participants had three or more places where they would keep documents. Often participants would describe the different systems they used when collaborating with other people on digital documents.

Participants usually described those systems in direct connection to the cases in which they used them. The pattern we observed was that participants who used many systems either had a heterogeneous workgroup or many collaborators outside their work group. We defined a heterogeneous workgroup as a workgroup whose members had diverse preferences for systems used in connection with their documents.

Problems Encountered with Document Organization.

Participants reported a great variety of issues. We already mentioned the effort needed to maintain an established folder system. This workload was reported to become unbearable as soon as participants interacted with people who had conflicting preferences as to how the File/Folder System should be organized. Participant 14 also reported an instance where it was impossible to pass on the work of a retired coworker due to the work being saved in the wrong folder system, which then became permanently locked after the retiree's account was deleted. Also, duplicates continued to be a problem for most of the participants who used a sophisticated folder system, even though those participants were aware of the options of links to documents.

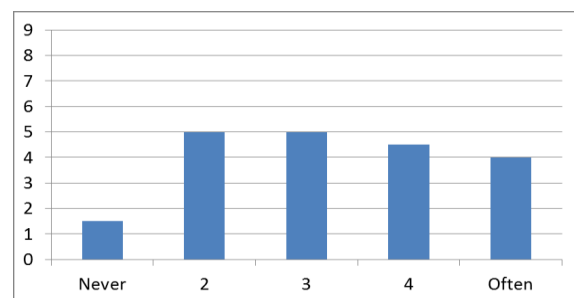


Figure 3: How often do participants want to (re)find occurrences of the same content

Another issue mentioned by several participants was Versioning. Dedicated versioning systems such as SVN or Git were only mentioned and used by one participant from the area of computer science. Versioning issues were not necessarily connected to duplicates, but were also triggered by working in different environments on the same document. For example, the same document was accessed via a home machine, a work machine and a tablet PC. In general, versioning failed due to the attempt to manage it manually, instead of having a dedicated system. One participant in a management position stated: "Versioning is impossible to maintain in our work group, it does not exist." This was again due to group members having different preferences for handling versioning.

Participants who used the same document in more than one system, or shared access to the same document with other users, mentioned the issue of synchronization. They recognized the availability of tools for synchronizing documents, but were either not able to use them due to incompatibility between systems, or lacked the knowledge or time to set them up.

The last reported issue was unique to the employees of the law firm. They have a large body of legal documents that include many intentional near duplicates and lots of re-used content. Whenever a legal phrase changes, every document containing that legal phrase needs to be found and adapted according to the change that was made. This proves to be such a difficult task that lawyers are advised to check all the provided legal documents before using them, as they may contain errors. One participant also mentioned that letters from other law firms would frequently contain such errors, indicating that this issue is a widespread one.

Discussion

No gender-specific differences were detected in our interviews, nor did we observe differences based on the age of the interviewees. The long years of personal experience with digital documents in a professional environment mean that the problems identified could not be explained simply by unfamiliarity with tools and systems.

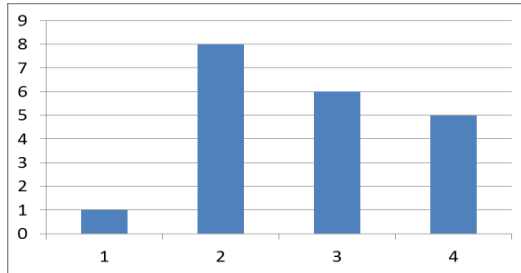


Figure 4: Number of places/computers in which digital documents are kept

18 of the 20 participants experienced problems while using the File/Folder system. Most frequently mentioned issues were (1) maintenance of the File/Folder system, (2) versioning of documents, (3) synchronizing documents, and (4) keeping track of instances of the same content. This verifies our assumptions about document management as outlined in the introduction and confirms the need for a new approach to help users organize their documents.

When we interviewed the participants, we realized that their differing work environments had a large influence on how many systems they used and how many issues they experienced with those systems. We observed ten participants in heterogeneous groups, while the other ten were either digital loners or in homogenous groups. Heterogeneous groups were made up of users with very different approaches to document management; this was usually due to the different backgrounds of the users, or to the fact that the users came from different work environments which overlapped where they collaborated. These participants encountered two main issues: Firstly, they encountered difficulties in agreeing on the way to use shared systems. Secondly, they often had no access to or knowledge of the systems that their collaborators used.

Whilst the first issue is hard to solve from a computer science perspective, the second is open to a technical solution. The issue may be remedied by providing a system that needs no separate introduction at the workplace nor does it need a central connection point.

The digital loners or homogenous user groups encountered fewer issues (3 out of 10 had none). They were mostly challenged by synchronizing or versioning efforts between workplaces or systems. Again, this may be remedied by a solution that is inherent to the document, instead of being applied to it.

The high level of re-used content in the participants' documents implies that the focus of a new document management system should not be on the documents, but on the actual content within the documents. This is underlined by the fact that participants

usually referred to specific content snippets instead of documents when asked about re-finding content.

The next section introduces our approach in targeting those issues whilst fulfilling the requirements set out in Section 2.

4 DOCUMENT DNA

When researching for systems that could fulfill our requirements, we realized that a concept from outside Computer Science may be applicable. If we were to think about files (documents) as life forms, then we can assume that the DNA of life forms behaves like annotations. These are extremely powerful annotations, as they store every bit of information needed for the life form, including heritage (i.e., versioning) information. Through the DNA it is possible to track where certain attributes (i.e., document content) originated and how they evolved over time. Our annotation approach is inspired by this analogy and is, consequently, named Document DNA (DDNA).

DNA and Phenographs

To model our annotations after conventional DNA, we need to better understand how DNA is currently used to identify relations between life forms. Phenographs are a way to depict relationships between life forms. The phenograph in Figure 5 shows polymers as life forms, and depicts sequential changes of a single nucleotide (A, C, G or T) resulting in several new life forms. For example, to follow just one path, the original ATCA was changed to ATCG and then to ATGG. As we plan to design the document DNA model in an analogous way to the biological DNA, we hope to be able to create trees similar to phenographs that will depict the relationships between documents (see Fig. 6).

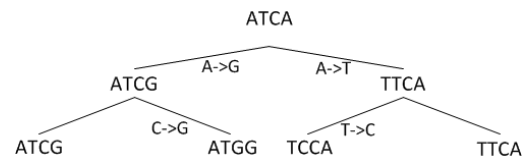


Figure 5: Phenograph by Lesk [12]

The Document DNA Approach

In keeping with the DNA comparison, we want to be able to track which source documents contributed content to a specific target document. Figure 6 illustrates our approach. Document A has been edited, and some further content from Document B has been added to Document A. When Document A is now saved, the DDNA would represent both sources of content (similar to parentage of a life form).

To describe our annotation approach in detail, we need clear definitions of the objects of our work. We therefore now introduce definitions of the core concepts used in our approach: document, document state, action, and session.

Documents

In our work, each digital document is defined by a set of triples consisting of the object (O), the content (C) and a temporary history (Z).

Definition 1: (Document). Document $D = [O, C, Z]$.

- *O*: The object is the container holding the content and any additional information concerning the content, for example formatting or character count.
- *C*: The content is the information the document contains, stripped of formatting or style. In a Word document this would mean the characters.
- *Z*: The temporary history of the document is used when the document is accessed and manipulated. Acts of manipulation are called actions. Every action is recorded in *Z*.

Following our example of Document A (in Figure 6), *O* represents the container of the document, *C* is the content of the document (textual or otherwise) and *Z* contains the edit actions in A and the paste action from document B. Note that the copy action would be part of the history *Z* of Document B, not of Document A.

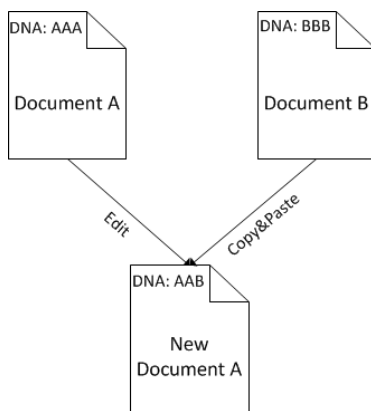


Figure 6: Document DNA

Document States, Actions and Sessions

Definition 2: Actions are basic activities that users apply to digital documents. They include:

- *Insert content* - A_i
- *Delete content* - A_d
- *Manipulate content* - A_m
- *Select content* - A_{se}
- *Copy content* - A_c
- *Paste content* - A_p
- *Save document* - A_{sa}

Actions are typically reflected in the document itself. Following the example in Figure 6, we have a copy and a paste action and a number of insert/delete/manipulate actions representing the editing of the document. All of those actions are contained in the *Z* component of the new Document A.

Different actions have different effects on object *O*, content *C* and temporary history *Z*. Here we list the effects for the seven actions identified above:

$$\begin{aligned} A_i(D) &= A_i([O;C;Z]) \\ &= [O + \text{details and format of inserted content}, \end{aligned}$$

$$C + \text{inserted content}, Z + \text{insert}]$$

$$\begin{aligned} A_d(D) &= A_d([O;C;Z]) \\ &= [O - \text{details and format of deleted content}, \\ &\quad C - \text{deleted content}, Z + \text{delete}] \end{aligned}$$

$$\begin{aligned} A_m(D) &= A_m([O;C;Z]) \\ &= [O + \text{details and format of manipulation}, \\ &\quad C, Z + \text{manipulate}] \end{aligned}$$

$$A_{se}(D) = A_{se}([O;C;Z]) = [O, C, Z + \text{select}]$$

$$A_c(D) = A_c([O, C, Z]) = [O, C, Z + \text{copy}]$$

$$\begin{aligned} A_p(D) &= A_p([O, C, Z]) \\ &= [O + \text{details and format of pasted content}, \\ &\quad C + \text{pasted content}, Z + \text{paste resource}] \end{aligned}$$

$$A_{sa}(D) = A_{sa}([O, C, Z])$$

$$= [O, C, \text{empty } Z \text{ and write history into DDNA}]$$

The activities of opening or closing a document are not regarded as actions, since they are not relevant to the versioning. When content is copied from one document, that content is held in a separate buffer (such as the Microsoft Clipboard). This content can now be pasted into another document by applying the paste command to the document.



Figure 7: Document States

A session always starts and ends with a saved document. There are at least two documents involved in each session, the starting document and the (final) saved document. The actual number of starting documents is not limited. Every document that is used in a session, and that is not the final saved document, is regarded as a starting document.

When actions are applied to a document, the state of the document may or may not change, depending on the action.

Definition 3: A document state is defined by the last entry in *Z* and whether the document was saved or not.

- *Temporary State:* The document will be in the temporary state, if the last entry in *Z* is either selection or copy. If the document is saved in this state, it will take on the form of the last consistent state or saved state.
- *Consistent State:* All other actions transform a document into the consistent state. The consistent state represents what the document will look like when saved, closed and reopened later on.
- *Saved State:* When saved, a document is transformed into the saved state. This also means that a freshly opened document is in the saved state. Selections or content held in a copy buffer will not be restored when reopened.

In a state diagram, we indicate different document states by differently shaped circles, see Figure 7.

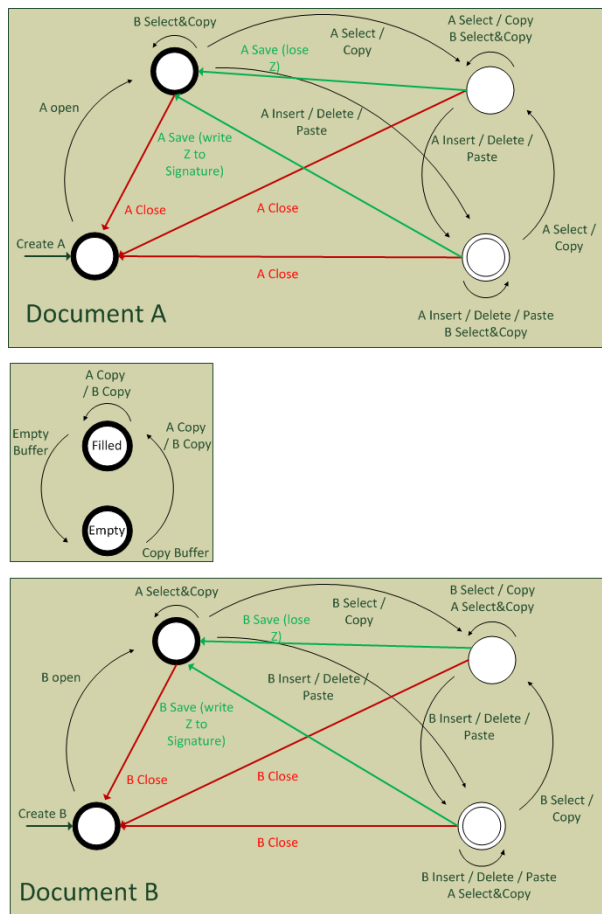


Figure 8: State Changes

When a document is saved (i.e., it enters the saved state), a new physical file is created and stored. Since storage is not a concern, we can keep unlimited versions of a document. It does not seem efficient to keep a version for each separate change or added character, so we decided that keeping a document every time a piece of work was saved should be sufficient. This means that if a user were writing a letter and saved it three times during the writing, there would be not one but three documents. This means that every 'save' action is treated as a 'save as ...' action.

Figure 8 illustrates all possible state transitions of two documents A and B for actions such as selecting, copying, pasting, opening and closing. Note that we also need to model the editor's buffer (e.g., the MS Word Clipboard) as it holds information to be pasted into a document.

Sessions

The last step required is to define the points at which the DDNA is adapted to reflect the changes made to a document. For performance reasons this cannot happen after every action (e.g., after every keystroke). We introduce Sessions:

Definition 4: A Session starts when a document is opened and ends when the document is saved. The Session represents all the actions between those events.

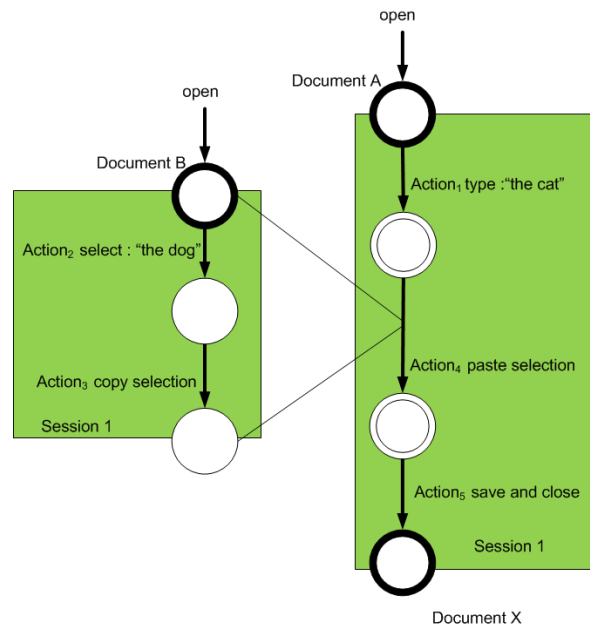


Figure 9: Document Session

The final saved document is always created as a new file and the document has a new unique DDNA. This new DDNA represents the history of the document inherited from the starting documents. The new DDNA is created by processing the actions that are recorded in Z. After saving, Z is cleared. Since a select and copy action may result in several paste actions, sessions may overlap. However, all other actions (i.e., other than copy-paste) belong to single sessions. Figure 9 illustrates the session of our starting example, with actions provided. First, Document A is edited by the user inserting text "the cat" (see first action after opening document A on the right in Fig. 9). Then the user copies the text "the dog" from Document B and pastes it into Document A. For simplicity here, we do not show the clipboard. Document A is then saved as the New Document A, here named Document X.

Following these definitions, the DDNA is a representation of Z that is updated with every saved state. We now follow with a description of our first prototype aiming to implement the DDNA.

5 MICROSOFT WORD ADD-IN PROTOTYPE

The DDNA for our prototype has four parts. The first three parts are needed to allocate the DDNA to the correct content. The fourth part represents Z from Section 4.

1. A UUID to uniquely identify a document. This UUID is static and never changes.
2. A series of timestamps in the form of *DateTime.Ticks* supplied by C#. This series is augmented with every save as a new tick is added.
3. The range of the content. This will be necessary when a copy-paste takes place. The range contains the starting

character position and the ending character position of the content.

4. A DNA signature representing the actual content. This signature should evolve according to the changes made to the content. We call this the DNA.

These parts are put together in this format:

```
<DDNA>
  <UUID>uuid</UUID>
  <Ticks>ticks</Ticks>
  <Range>range</Ranges>
  <DNA>dna</DNA>
</DDNA>
```

Documents will start with such a simple DDNA. The static *UUID* allows us to track a document even it is renamed or the content is drastically changed, as the *UUID* never changes. The ticks allow us to determine the hierarchy (according to the time of creation) of several documents with the same *UUID*. The ticks also allow us to track the closest version available that has been shared as a previous version, since that document will have only the ticks that are shared by all subsequent documents.

However, if content gets pasted into a document, it will bring its own DDNA with it from the source document. This DDNA will be added to the one already available for the document and the *range* identifiers of both DDNAs will be modified. The *range* identifiers will allow us to identify which content belongs with which DDNA. *Range* identifiers also change if the content is changed within the document to reflect the *range* of the new content.

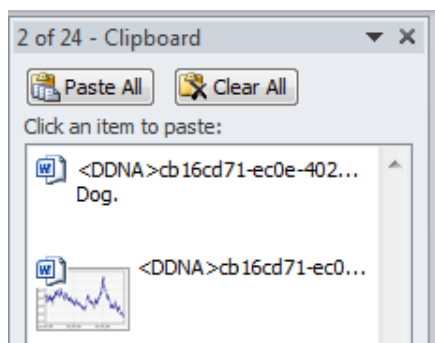


Figure 10: Clipboard containing DDNA

The DNA part of the DDNA is represented using a lossy compression function of the content. It works such that if two documents are equal in all but two words, they will then have an almost identical DNA, but if they differ completely, their DNAs will also be completely different. Currently we explore the quality of different functions in the DDNA context.

We decided to use Microsoft Word as the platform for our prototype since our interview series confirmed it as being the most widely used editor. We also wanted to have a solution that was easy to apply to users' workspaces without interfering with their everyday work. Therefore, we decided to use the AddIn option within Microsoft Word. We decided to store the DDNA information in the document properties as a custom property, since there it is accessible both for our AddIn and for us to check manually.

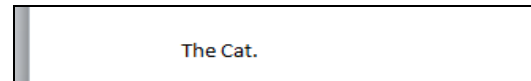


Figure 11: Initial document (before pasting).

We made use of the *repurpose command* option available for Microsoft Word to automatically add and maintain the DDNA. This option allows us to add code for the commands that users execute, therefore automating the maintenance of the DDNA within the work processes of the users.

The first step is to supply every document with a unique DDNA. To do so, we repurposed the *save* command. Whenever the user hits *save*, the AddIn checks if the saved document has a DDNA. If the document does not have a DDNA, the AddIn creates one and adds it. If a DDNA is found, the *range* and the *ticks* properties will be adapted according to the changes made since the last *save*. We chose this point since we think that the *save* action marks the point in time when the user completes a significant amount of work. We also added a subroutine that saves the previous version of the document in an archive folder. We did this both to supply the user with the means to return to previous versions, and to have more data with which to evaluate our prototype later.

The DDNA needs to be transferred with every *copy paste* cycle, as shown in our concept. We repurposed both the *copy* and the *paste* commands in MS Word to achieve this. When a user copies content from a document, the appropriate DDNA is added to the content. This is visible in the clipboard (see screenshot in Figure 10). Here we revisit the example from Figure 6 and also show additionally that images can be handled the same way as text.

Figure 11 shows a document before the contents of the clipboard shown in Figure 10 are pasted and Figure 12 shows the same document afterwards.

When the *paste* command is executed, the added DDNA information is automatically removed from the content. The DDNA will then be permanently added to the document if it is saved. We keep track of the appropriate range information by logging at which character positions *paste*, *insert* or *delete* actions happened.

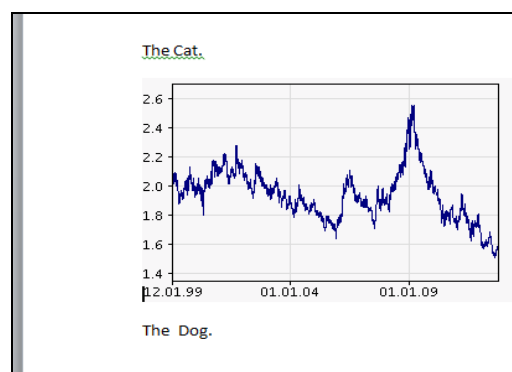


Figure 12: Document after pasting

Our prototype currently enables us to detect instances of the same content in all available documents, even if the content has been modified. We are also able to determine the version hierarchy of different instances of the same content. The next development will

permit detection of precise differences between content instances simply by comparing the DDNA, as well as enabling reconstruction of the operations that led to those differences.

Discussion

Our prototype currently allows us to tackle two of the issues found in the user study. The first was that participants found it difficult to determine the most recent version of documents on which they had collaborated. This issue is remedied by simply listing all documents that share the same DDNA and selecting the document with the most recent timestamp included in the DDNA.

The second issue we are able to solve with our prototype is the re-finding of other instances of the same content, for example for update purposes. We simply need to take the DDNA representing the content snippet and search for other documents containing this DDNA.

These two key features enable users to spend less time organizing their files when collaborating and instead allows them to focus on the collaboration.

A final feature of the prototype that needs attention is the ability to compare two versions of the same content snippet and measure the degree of semantic difference between those two snippets with high accuracy.

7 CONCLUSIONS

In this paper, we introduced a novel decentralized approach for annotating documents with metadata to track the evolution of the digital content within the document, as well as the sources of that content.

Summary of Contributions.

We first compared and categorized recent approaches for metadata annotation on document. We found that there has been no shortage of research activity in this field in the last 17 years, but some issues remain unsolved. We discussed the results of an interview series we conducted to further investigate the issues users encountered in the organization of their documents. We then introduced our DNA-inspired approach, which aims to combine the advantages of automated meta-data annotation with those of manual annotation, and to correct for the deficiencies of these two categories of system. We then defined the objects and concepts of our research in detail. Finally, we described our prototypic implementation of the DDNA concept.

Future Work

The next steps need to include the following two tasks: (1) Execution of a longer-term user study to evaluate the document DNA with users “in the wild” (i.e., at their workplaces). (2) Improving our text compression method to permit a small signature, and then evaluating that method.

For the first task, we need to design a study to enable long term observation of single users and how they handle their data. Another possibility would be to observe work groups and how collaboration is simplified by using our system.

The second task of qualitative instead of quantitative observation of change is more challenging. We will investigate work in the areas of hashing and fingerprinting [15, 17, 18, 21] in addition to lossy compression methods currently in use. We will define measurements on the distinctiveness of content instances, as well as methods to make those measurements. We also need to

determine how to measure successfulness in document organization, for example how quickly do users find the files for which they search.

We believe that DDNA can make a significant contribution to users’ efforts in organizing and re-finding content that is shared across many documents. We further envision that DDNA can be used for research purposes, for example to find out which documents form the kernel of the most frequently used content. This information could be used to form clusters and automatically support the users in the organisation of their files. We also recognize that DDNA might be applicable to the field of DRM, however, that is not our focus.

ACKNOWLEDGMENTS

We thank all volunteers from our interview series. We would also like to thank *anonymised* for funding this research with a *anonymised* Doctoral Scholarship.

REFERENCES

1. Barreau, D. and Nardi, B.A. (1995). Finding and reminding: File organization from the desktop. *SIGCHI Bulletin*, 27(3), 329-339.
2. Bush, V. As we may think. *Atlantic Monthly*, 176:101-108, 1945.
3. Chiara, R. D., Erra, U. and Scarano, V. Vennfs: A venn-diagram file manager. In *Proceedings of the Seventh International Conference on Information Visualization*, 2003.
4. de la Cruz, F. and Davies, J. Horizontal gene transfer and the origin of species: lessons from bacteria. *Trends in Microbiology*, 8(3):128-133, 2000.
5. Dragunov, A.N., Dietterich, T.G., Johnsrude, K., McLaughlin, M., Li, L., Herlocker, J.L.. TaskTracer: A Desktop Environment to Support Multi-tasking Knowledge Workers. *International Conference on Intelligent User Interfaces*. p. 75-82, 2005.
6. Fallin, K. Entropy: *Managing data in an electronic world*, 2003. Undergraduate research project, University of Calgary.
7. Fertig, S., Freeman, E. and Gelernter, D. Lifestreams: An Alternative to the Desktop Metaphor. *CHI'96 Conference on Human Factors in Computing Systems*. Video Program (Vancouver, Canada, April 1996). ACM Press, 1996.
8. Hopkins, I. and Vassileva, J. Beyond keywords and hierarchies. *Journal of Digital Information Management*, 3:139-145, 2005.
9. Jensen, C., Lonsdale, H., Wynn, E., Cao, J., Slater, M., Dietterich, T. G. The Life and Times of Files and Information: A Study of Desktop Provenance. *CHI '10*. ACM Press (2010), 767-776.
10. Karypidis, A. and Lalis, S. Omnistore: A system for ubiquitous personal storage management. In *IEEE International Conference Pervasive Computing and Communications*, pages 136-147. IEEE Computer Society, 2006.
11. Karypidis, A. and Lalis, S. Automated context aggregation and file annotation for pan-based computing. *Personal and Ubiquitous Computing*, 11:33-44, 2007.
12. Kim, Y.-W., Moon, K.-A. and Oh, I.-S. “A Text

- watermarking Algorithm based on Word Classification and Inter-word Space Statistics”, *IEEE, Seventh International Conference on Document Analysis and Recognition Volume II*, August 03-06, 2003
13. Ko, R.K.L., Jagadpramana, P. and Lee, B.S. “Flogger: A file-centric logger for monitoring file access and transfers within cloud computing environments,” in *IEEE TrustCom/IEEE ICSS/FCST*, International Joint Conference of. Los Alamitos, CA, USA: IEEE Computer Society, 2011, pp. 765–771.
 14. Lesk, A.M. *Introduction to Bioinformatics*. Oxford University Press, 3rd edition, 2008.
 15. Rajaraman, A. and Ullman, J. *Mining of Massive Datasets*. 2011. Available online at <http://infolab.stanford.edu/~ullman/mmds.html>, last visited 06/10/2011.
 16. Satoh, K. and Okumura, A. Documentation know-how sharing by automatic process tracking. In *Proceedings of the 4th International Conference on Intelligent User Interfaces*, 1999.
 17. Schleimer, S., Wilkerson, D. S. and Aiken, A. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, 2003.
 18. Shivakumar, N. and Garcia-Molina, H. Finding near-replicas of documents on the web. In *The World Wide Web and Databases*. Springer Berlin / Heidelberg, 1999.
 19. Signer, B. What is wrong with digital documents? A conceptual model for structural cross-media content composition and re-use. In *Conceptual Modeling - ER 2010*. Springer Berlin / Heidelberg, 2010.
 20. Soules, C.A.N. and Ganger, G.R. Why can't I find my files? New methods for automating attribute assignment. In *Proceedings of the 9th conference on Hot Topics in Operating Systems - Volume 9*, 2003.
 21. Stein, B. Principles of hash-based text retrieval. In *Proceedings of the 30th Annual International ACM SIGIR conference on Research and Development in Information Retrieval*, 2007.
 22. Svensson, M. Contextual metadata in practice. In *Advances in Multimedia*, 2009. MMEDIA '09.
 23. Xu, Z., Karlsson, M., Tang, C. and Karamanolis, C. Towards a semantic-aware file store. In *Proceedings of the 9th conference on Hot Topics in Operating Systems - Volume 9*, 2003.

Exploring the applicability of Reservoir methods for Classifying Punctual Sports Activities Using On-body Sensors

Doug P. Hunt¹

Dave Parry¹

Stefan Schliebs¹

¹ School of Computing and Mathematical Sciences,
AUT University,
Private Bag 92006, Auckland, New Zealand 1142,
Email: dp hunt, dparry, sschlieb@aut.ac.nz

Abstract

This paper explores the use of a reservoir computing (RC) method, Echo State Networks (ESN) to classify inertial sensor motion data collected from sensors worn by horse riders into punctual activities of interest within a scripted movement environment. RC methods incorporate both temporal and spatial aspects within the model and therefore may have applicability classifying signals with the varying temporal signatures often seen across activity instances even when performed by the same subject. ESN's, as one of a number of RC methods has a potential advantage in this case of being able to directly incorporate the inertial data into the reservoir without the need to segment this data into sliding windows. This is part of a wider set of work to build a wearable coach for technique feedback for Equestrian sport. Our use of RC methods on inertial data to classify punctual human activities is novel.

Keywords: Echo State Networks, Reservoir Computing, Spatio-temporal data processing, Punctual Activity Classification, Activity Classification, Machine Learning, Equestrian sport

1 Introduction

Human activity recognition using on-body sensors presents a number of challenges (Avci et al. 2010). In many situations this includes a lack of knowledge of overall context such that while it may be possible to distinguish a gesture such as pronating the wrist, it is difficult, without some idea of overall context, to conclude reliably that the gesture is associated with, for example, opening a door by twisting the door handle or turning a car key in a vehicle ignition. One method used by researchers to resolve this dilemma is to embed the sensor within some other item such as a car key, pen or baseball bat (Verplaetse 1996) that has a particular use that constrains the context. Another method, using a more generalised sensor, is to assume or predicate (Lukowicz et al. 2004, Ward, Lukowicz, Troster & Starner 2006) a particular context or domain so that the choice of meaning of the gesture is constrained by the predicated domain. This is our own approach and in our case our predicated domain is that of Equestrian sport. Sporting domains have

some additional benefits as a result of often being strongly defined by rules and traditions.

Another challenge in human activity recognition is the variability in both the spacial and temporal aspects of a particular action both across subjects and even within a single subject who performs an activity (or action) more than once (Ward, Lukowicz & Troster 2006). Some obvious examples of spatial variability includes differences between left-handed and right-handed activities and differences in technique such as between the drive shot of a professional golfer and an amateur. Examples of obvious temporal differences include taking 75 seconds to mount your horse because the horse is moving away from you or mounting in 7 seconds (or less) as the horse is standing still.

Human activities may be broadly distinguished into durative or punctual activities. Durative activities usually occur over a longer period of time and have some sort of repeating rhythmic component. Some simple examples of durative activities include running, walking, rowing, grooming a horse and cycling. Punctual activities tend to be shorter and happen once rather than multiple times and so often do not have a repeating rhythmic component to the signal. Some simple examples of punctual activities include bowling a ball in cricket, hitting a ball in baseball, a backhand shot in tennis and mounting a horse. Most of the activity recognition literature looks at durative activities. This work looks at punctual activities using current state-of-the-art temporal pattern recognition methods.

This paper is divided into 5 main parts. In section 2 we describe the application domain including our goal so that other practitioners can understand our motivation and assess the relevancy of this work. In addition we define the activities to be classified, with an illustration and identify the entities involved with our goal scenario. In section 3 we describe the experimental set up including our approach, the equipment used, the participant, the activity setting/scenario, the data obtained and its main characteristics/metadata such as sample rate and precision, the preprocessing applied to the data prior to classification and a description of the classification engine. In section 4 we present the results obtained and in section 5 we discuss our conclusions and future work.

In reporting this work we have followed the recommendations in “*How to do good research in activity recognition*” (Plötz 2010).

2 Application Domain

The goal of our research is to construct an Activity Classification System (ACS) to classify activities of interest within Equestrian sport. Data input into the

Copyright ©2014, Australian Computer Society, Inc. This paper appeared at the Thirty-Seventh Australasian Computer Science Conference (ACSC2014), Auckland, New Zealand, January 2014. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 147, Bruce H. Thomas and David Parry, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.

system is collected from horse riders using wearable inertial sensors. The ACS is envisaged to be part of a wider, wearable coaching system that would be designed to provide horse riders with postural and riding feedback as they train. We envisage such a system as having a small number of generalised sensors on the rider's body at strategic positions that are used for multiple purposes.

In this paper and planned follow on work we have set out to classify the activities *Mounting* and voluntary *Dismounting*. Mounting is when the rider gets on the horse and dismounting is when they get off the horse. At this stage we are only seeking to classify voluntary dismounts and we exclude involuntary dismounts or falls.

As with all activities, identifying the precise beginning and ending of a mount/dismount is not trivial and so a definition of the activity with a recognisable start and end is required. For this work, our activity definitions are:

Mount A stirrup mount from the time when a rider with one leg in the stirrup, lifts the second leg off the ground or mounting block in order to mount until the time when they are seated in the saddle.

Dismount The time from when a rider leans forward (prior to dismounting) until they are standing on the ground.

Neither definition is inclusive of all possible classes of either mounting or dismounting but we consider them sufficient to cover a reasonable percentage of the mounts and dismounts that are likely to be encountered in equestrian sport. As far as we are aware, there are no other generally accepted definitions for these activities, for the purposes of activity classification.

The entities involved are the horse rider and the horse. In this case the horse is a built for purpose wooden model that allows multiple mounts and dismounts in a laboratory setting without having to deal with horse welfare or safety issues.

3 Experimental Set-up

3.1 Approach

Mounts and dismounts are punctual rather than durative activities. Punctual activities are short, simple activities such as opening a drawer, sitting down, picking up a cup, bowling a ball in cricket, hitting a ball in baseball, a backhand shot in tennis or shooting a hoop in basketball. Durative activities are of longer duration and usually have a repeated element such that there is a rhythmic or cyclic nature to the activity such as walking, running, rowing, cycling or grooming a horse. Most of the ACS research for wearable sensors that has successfully moved out of a scripted activity, laboratory environment into the real world has either been based on classifying durative activities or has made use of additional non-inertial sensors such as cameras and sound in addition to inertial sensors to reliably classify activity.

Much of the successful work classifying durative activities has used sliding window techniques to break the raw input into fixed blocks or windows and then calculates a number of signal derivatives for each window. Those window derivatives are then used as inputs to the ACS. Example activities include walking, running and being still. These being successfully classified based on windowed properties of the sensor signal such as signal standard deviation (Lee & Mase



Figure 1: Mounting a horse using a stirrup mount

2002, Lester et al. 2005), spectral frequency (Lukowicz et al. 2004, Lester et al. 2005), spectral entropy (Bao & Intille 2004, Ermes et al. 2008, Lester et al. 2005), integrals, means and variances (Lester et al. 2005). Sliding window techniques have had some success with scripted punctual activities but have been markedly less successful in real world situations.

In this work we use a Reservoir Computing (RC) technique, called Echo State Networks (ESN) to directly classify the pre-processed input data without the need to window this data.

3.2 Data Collection

Data was recorded from two laboratory and 55 real life riding sessions from 20 participants (and their horses) over a three month period using a commercially available, six degrees of freedom inertial sensor from SparkFun (SparkFun Electronics Inc 2008). Data collection was done as part of a Masters project (Hunt 2009) and all data was collected within Sweden. Each session was videoed so that activities could be manually classified by participants, riding domain experts and the research team.

The data for this set of experiments was taken from one of the two laboratory sessions and during this session the participant, who was an experienced rider, wore the sensor on her right wrist using a simple stretchable Velcro bandage for attachment. The participant self-described herself as right-handed. The “horse” used during the laboratory sessions was a built-for-purpose wooden framed horse (Diana) of approximately 16 hands in height (163cm at the “shoulder”), draped with a standard European riding saddle and stirrups. During this particular laboratory session the participant mounted and dismounted 17 times following a proscribed script. During the laboratory sessions the video camera was fixed into position using clamps so that the researcher was free to move around if needed.

3.2.1 Sensor

The SparkFun 6DoF inertial sensor contains a Freescale MMA7260Q triple-axis accelerometer, two InvenSense IDG300 500° per second gyroscopes and both a Honeywell HMC1052L and a HMC1051Z magnetic sensor.

The sensor outputs readings from a 12 bit analogue to digital converter that gives a reading range between 0 and 1023.

The sensor outputs are:

1. a sample start character “A”
2. an unsigned 15 bit serialised sample number
3. three axis of magnetic readings
4. three axis of accelerometer readings
5. pitch, roll and yaw readings
6. a sample stop character “F”

Giving a total of 12 data fields per sample. For example:

```
A,0,569,498,504,577,344,576,467,446,462,F
A,1,569,493,503,567,342,571,466,458,464,F
A,2,569,496,504,0,340,1023,465,456,464,F
```

Sensor readings were sampled at 10Hz and broadcast via Bluetooth to an on-body receiver for logging and later analysis. The accelerometer in this sensor has a settable scale and for this session it was set to record $\pm 2G$'s.

3.2.2 Session Script

The script asked the participant to start and finish each mount/dismount pair at the same spot in the laboratory, within three metres of the wooden horse but clear of all obstacles. Prior to each

mount/dismount pair the participant clapped her hands over her head two times as a synchronisation signal (to enable the inertial data to be synchronised with the video). After each set of claps and upon mounting the participant was asked to pause for approximately five seconds by standing or sitting still.

3.3 Data Description

The participant was asked to mount and dismount as closely as possible to her normal technique and apart from the requested pauses was not asked to keep to any particular time schedule. The duration of each mount and dismount is, never-the-less, reasonably consistent with a significant but gradual shortening of duration as the participant gets used to and more proficient at the script. The first mount takes 11 seconds while the last mount takes 6.5 seconds with progressive shortening in between. The first dismount takes 9 seconds and the last dismount takes 6.7 seconds again with progressive shortening. There is also a small shortening of the interval between each mount/dismount pair with the first interval being 11.3 seconds and the last being 7.2 seconds. However there is also a longer interval after the first two pairs.

The interval between each mount and dismount (when the participant is sitting on the horse) is much more consistent (probably as a result of the script). The first and second mount to dismount intervals are both 4 seconds and the last mount to dismount interval is 3.4 seconds.

The periods prior to the mount/dismount series records the sensor on a bench after being turned on, being fitted to the participant's wrist and then the participant waiting around while the researcher checked and adjusted equipment such as the video camera. The period after the series is essentially the participant taking off the sensor and it being placed back on a bench.

Figure 2 depicts the recorded time series after pre-processing and manual labelling. The three panels each show the 3-dimensional recordings from the accelerometer, gyroscope and magnetometer respectively. The alternating background stripes show the activity undertaken during recording (mounting, dismounting and null class). The figure depicts all 17 mounts and dismounts.

From figure 2, careful observation shows that there is a slight drift upwards over time with the Gyroscope data. This drift is relatively common with lower priced Gyroscopes and this tendency has been ignored in our data analysis. In addition, the y axis of the Magnetometer data also drifts upwards over time, while the x and z axis do not show the same level of drift. The presence of the drift on only a single axis is possibly due to the sensor moving slightly on the participant's wrist as they do things. In addition, some level of drift is common with consumer level Magnetometers such as this one but normally the drift would be present on all axis. Both instances of signal drift have been ignored in this work.

3.4 Preprocessing and Labelling

3.4.1 Editing & Error Checking

The raw data files are hand edited to remove extraneous set up data and commands that were logged before and after the main data file. Column labels are inserted as the first line of the cleaned file. An error checking routine is then run against the file to check for missing and out of range data. This routine

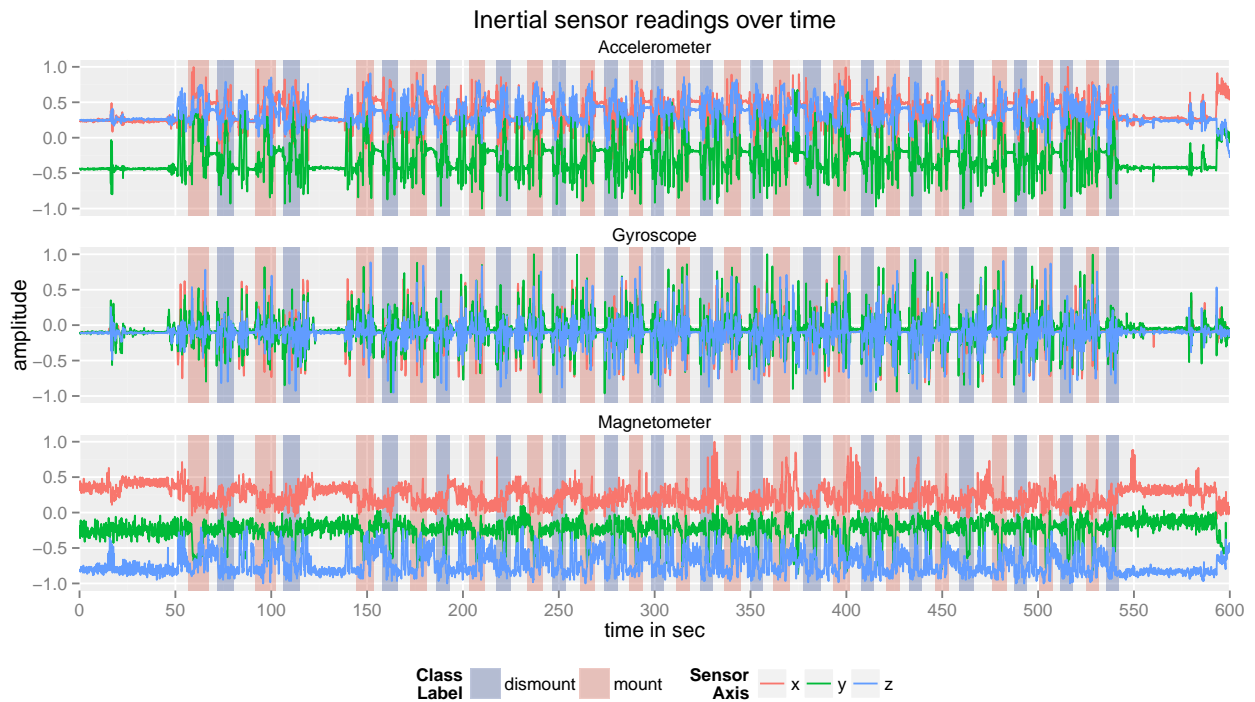


Figure 2: Input Data Set

also strips off the start and end characters (A & F) and changes the end-of-line character sequence. The raw files have an unusual LF-CR end-of-line (eol) sequence and this is replaced with CR-LF. Once the 15 bit sample number has been checked for missing or duplicate samples it is replaced by an unsigned integer that does not overflow. No out of range, invalid or missing data was found in this file.

3.4.2 Synchronisation & Labelling

The overhead hand clap leaves a distinctive signal in the sensor data and provides an obvious counterpart in the video. The sensor data is searched visually to identify the first two sets of overhead hand claps and the time between the sets of claps is measured. The video is then run and the first two sets of hand claps are identified. The time between clap sets is checked to ensure the same claps are being used. The sensor data and video are then synchronised based on a detailed visual evaluation of the first set of hand claps using both the sensor signal and a frame by frame view of the video. Once synchronisation is established it is checked against later hand claps to ensure correctness and negligent temporal drift.

Once synchronisation is complete the video is used to find the frame where the occurrence of each activity starts and finishes. The video frame numbers are then translated into an applicable sample numbers from the sensor stream, based on the synchronisation point, and class labels are added to the sensor data based on our activity definitions.

3.4.3 Data Cleaning

The sensor readings are normalised to a range between -1 and 1. The Echo State Network (ESN) prefers input in the range $[-1, 1]$ and so normalisation gives an opportunity to extract maximum infor-

mation from the signal. Prior to normalisation we remove the 0.1% upper and lower quantiles. Removing the outliers and replacing them with the signal mean value allows us to maximise the signal range at the cost of a very small number of changes to the signal. Outlier removal is somewhat controversial and we discuss this in section 5.

3.5 ESN Model Description

In this study, we have chosen an Echo State Network (ESN) as our classifier. ESN were first introduced in (Jaeger 2001). ESN have been employed for a wide range of spatio-temporal real-world problems such as speech recognition (Jaeger et al. 2007, Verstraeten et al. 2006), financial forecasting (Ilies et al. 2007) and the prediction of chaotic dynamics (Jaeger & Haas 2004). Here we briefly outline the concept of the method and refer to the excellent review on ESN and related reservoir techniques presented in (Lukoševičius & Jaeger 2009) for further details. For the sake of consistency with previous descriptions of ESN, we adopt the nomenclature defined in the review article mentioned above.

The ESN attempts to learn a functional mapping from a (possibly multi-dimensional) input time series $\mathbf{u}(t) \in \mathbb{R}^{N_u}$ to a target time series $\mathbf{y}_{\text{target}}(t) \in \mathbb{R}^{N_y}$ based on a training data set $\{\mathbf{u}(t), \mathbf{y}_{\text{target}}(t)\}$ with $t = 1, \dots, T$ where T is the size of the training set. The ESN is a neural network consisting of N_u input neurons, N_x reservoir (hidden) neurons and N_y output neurons. The reservoir neurons are either fully or sparsely interconnected with connection weights specified by a weight matrix $\mathbf{W} \in \mathbb{R}^{N_x \times N_x}$. Matrix \mathbf{W} is initialized with random (uniform) weights and then scaled by the spectral radius $\rho(\mathbf{W})$ which is the largest eigen value of \mathbf{W} . Generally, all N_u input neurons are connected to all reservoir neurons via

connection weights defined by a separate input matrix $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_x \times N_u}$. The weights of the input matrix are initialized as either -1 or 1 and then scaled by a scaling factor.

At time step t , the input $\mathbf{u}(t)$ is fed into a neural network. The output of all reservoir neurons in the network is computed:

$$\mathbf{x}'(t) = f(\mathbf{W}_{\text{in}}\mathbf{u}(t) + \mathbf{W}\mathbf{x}(t-1)) \quad (1)$$

$$\mathbf{x}(t) = (1-a)\mathbf{x}(t-1) + a\mathbf{x}'(t), \quad (2)$$

function f being a neuron activation function usually defined as the (element-wise) hyperbolic tangent $\tanh(\cdot)$, and factor $a \in \mathbb{R}$ being a leaking rate that controls the contribution of the previous neural output to its current state.

Since recurrent neural networks are difficult to train through gradient-descent based learning methods, ESN propose an elegant and efficient solution for imposing a desired input-output behaviour onto the network. Input and output vectors at time step t are concatenated and then linearly transformed into the final output of the network:

$$\mathbf{y}(t) = f_{\text{out}}(\mathbf{W}_{\text{out}}[\mathbf{u}(t)|\mathbf{x}(t)]) \quad (3)$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_y \times (N_u + N_x)}$ is a weight matrix connecting all reservoir and all input neurons with N_y output neurons, $[\cdot]$ represents the vertical concatenation of vectors and f_{out} is the activation function of the output neurons which is usually chosen as the identity.

The learning task is defined as an optimization problem in which the difference between $\mathbf{y}(t)$ and $\mathbf{y}_{\text{target}}(t)$ is minimized. Arguably the most popular method of computing matrix \mathbf{W}_{out} is linear regression or its regularized extension called ridge regression:

$$\mathbf{W}_{\text{out}} = \mathbf{Y}_{\text{target}}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \alpha^2\mathbf{I})^{-1} \quad (4)$$

where $\mathbf{I} \in \mathbb{R}^{N_x}$ is the identity matrix and $\alpha \in \mathbb{R}$ is a regularization factor that has to be carefully tuned for optimal results. Matrix $\mathbf{Y}_{\text{target}} \in \mathbb{R}^{N_y \times T}$ contains all vectors $\mathbf{y}_{\text{target}}$ concatenated into a matrix. We note that the connection weights \mathbf{W} are not modified by the learning rule and only the output weights \mathbf{W}_{out} are updated during training.

The role of the reservoir is the transformation of the input signal into a high-dimensional intermediate feature space represented by the neural network output at any time t . Although linear methods are then used to transform the feature vector into a desired target output, the mapping of the input $\mathbf{u}(t)$ to output $\mathbf{y}(t)$ is of non-linear nature.

The concept of generating intermediate feature vectors is also exploited in other kernel-based machine learning algorithms, most prominently the Support Vector Machine (SVM). By choosing a straightforward linear learning rule, the training process becomes highly efficient. ESN allow the exploitation of the interesting characteristics of recurrent neural networks without the need of mathematically and computationally complex training algorithms.

4 Results

We used an evolutionary algorithm, i.e. a Particle Swarm Optimiser (PSO) to do a search of the parameter space for the ESN, to find a suitable configuration to use for our experiment. We optimize the regularization parameter α , the number of reservoir neurons

N_x , the scaling factor of the input weights, the leaking rate a and the spectral radius $\rho(\mathbf{W})$. The ranges for these ESN parameters are generally accepted sensible ranges and were taken from (Lukoševičius 2012) and are shown in table 1. We ran the PSO with 14 particles over 50 generations and within each generation the ESN was initialized and trained five times to take some account of the stochastic nature of the initialization process. Each ESN was trained on the first 50% of the time series and the entire series was used for testing. The root mean square error between target and actual network output on the test set was used as a fitness measure for the PSO.

The results of this search process can be seen in figure 3. As is visible from the diagram, four of the five parameters (Number of neurons, Input Scaling, Leak Rate and Spectral Radius) had settled into a small range within 20 generations. Regularization, the fifth parameter, appears less critical for this problem and a range of configurations reported satisfying test errors.

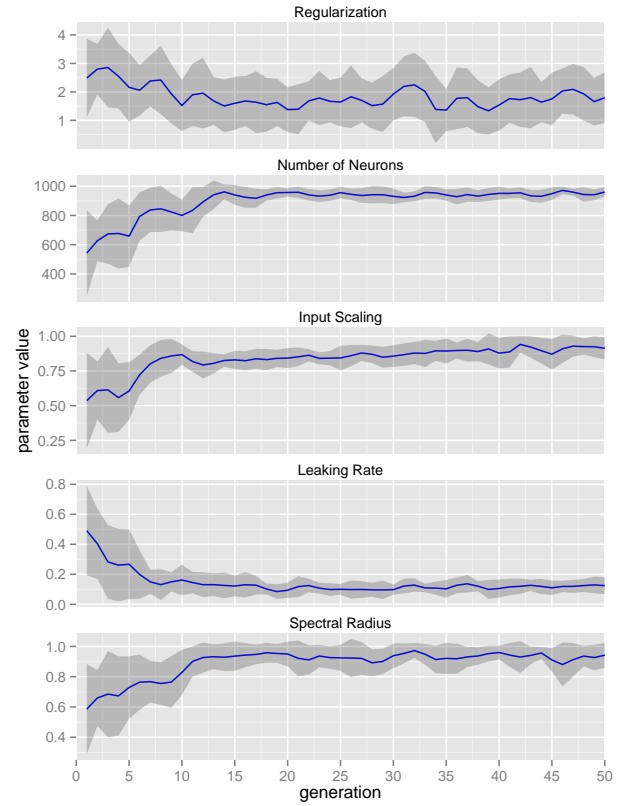


Figure 3: Optimising the ESN parameters

ESN parameters & ranges		
Regularisation	α	0 to 5
Number of Neurons	N_x	100 to 1,000
Input Scaling		0 to 1
Leaking Rate	a	0 to 1
Spectral Radius	$\rho(\mathbf{W})$	0 to 1

Table 1: PSO Parameter Ranges

The selected parameters from the PSO that were used to run the ESN were Regularisation (2.7), Neurons (939), Input Scaling (0.9147), Leaking Rate (0.05937) and Spectral Radius (1.0).

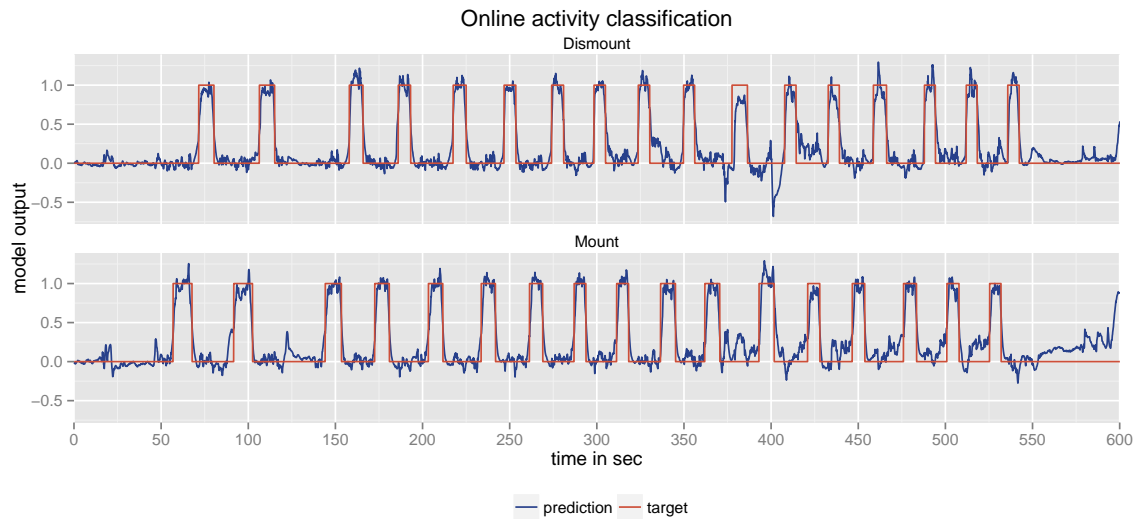


Figure 4: ESN Results

Fifty percent of the data file was used for training and the full file was used for testing. The file was divided in two parts with no account for where that separation point fell in terms of the classes. In this case, as a result of where the mounts and dismounts occur, the first half of the file includes the first 8 mounts and the first 7 dismounts. The output from the test run is shown in figure 4.

Using a cut off point set at 0.5 of the continuous output $y(t)$ has all mounts and dismounts successfully classified with one false positive mount classified during the sequence when the participant takes off the sensor. No false positive dismounts were classified. Within each mount and dismount there is a slight lag between the class label and the classified label in most cases with an associated drop off towards the end of each class. This is an expected characteristic of the reservoir property as the reservoir needs some time to establish a recognised pattern.

Class Label	Predicted Label		
	0	1	2
0	3298	53	45
1	103	1281	0
2	58	0	1163

Table 2: Confusion Matrix of Results

5 Conclusions and future work

5.1 Conclusion

The ESN classifier has worked well in this particular situation, using data from a scripted activity in a laboratory setting collected from a single participant. The false positive mount possibly indicates that the ESN classifier would benefit from additional training that provides a wider variety of signals that are outside of the desired classes. This is supported by some follow on work (not shown here) where we used the last eight mounts and dismounts as training data with the full set of mounts for testing and in this case we

provided the “taking the sensor off” segment as part of the training and no false positives were classified.

Our use of overhead claps for synchronisation and scripted pauses in activity introduce significant signal artefacts into our data. While it is not possible with ESN to tell what properties of the signal are used by the classifier it is probable that our artefacts make a significant contribution to classification. For example, the initial spike in the classifier response in the wider gap between the second dismount and the third mount may be indicative of the ESN starting to respond to the synchronisation signal. In addition, the clear pause in activity after mounting consistently occurs just prior to dismounting and so this artefact is undoubtedly contributing to dismount classification success. We would be unwise to conclude that our current classifier is suitable for use on data that does not contain artificial artefacts.

The current pre-processing methods that we use to normalise the signal between -1 and +1 will have created additional signal artefacts including magnifying the drift in the Magnetometer and Gyroscope data. Our removal of outliers has enabled us to amplify the central portion of the signal but the outliers are real data and so by removing them we have changed the underlying activity signatures. These issues, without resolutions, will make it difficult to generalise the results across participants and across other sensors.

This data was collected from a single individual using a single (wooden) horse on the same day using the same equipment. This provides unrealistic consistency. In the real world not only do riders not follow a script when mounting and dismounting they also come in all sizes, temperaments and skill levels; their horses come in all sizes, temperaments and training levels; additional equipment may be involved such as a rider holding a crop while mounting and differing techniques may be used while mounting. All of this cautions us against simplistically concluding that we are close to having a simple, reliable method of classifying this (or any other punctual activity) that works across riders and situations.

This reported experiment is a idealised situation, designed to provide the best chance of successful classification and so while we are pleased that these re-

sults are positive they need considerable further development before we can safely conclude that RC methods are well suited to classifying punctual human activities based on inertial data. Despite this caution, these results are positive enough that we and perhaps other researchers are willing to follow this path further.

5.2 Future work

This work is part of a larger set of work designed to resolve some of the issues mentioned in our conclusions as well as additional issues. Some of our future plans include:

- Comparing the ESN classifier with more traditional kernel based classifiers such as Support Vector Machine.
- Testing to see if the classifier will generalise across participants.
- Testing to see if the ESN classifier can generalise across activities in different places and time for the same individual.
- Training and testing the ESN on data collected during real world situations.
- Designing methods of translating our sensor data from the form in which it was captured (sensor dependent) into a standardised format more useful for input into the ESN in a way that allows us to draw the maximum information out of the data while still allowing it to be comparable between data collection sessions. This includes alternate methods of pre-processing the data to filter out noise, to account for drift and methods for including outliers.

References

- Avci, A., Bosch, S., Marin-Perianu, M., Marin-Perianu, R. & Havinga, P. (2010), Activity Recognition Using Inertial Sensing for Healthcare, Well-being and Sports Applications: A Survey, in '2010 23rd International Conference on Architecture of Computing Systems (ARCS)', pp. 1–10. 0033.
- Bao, L. & Intille, S. (2004), 'Activity recognition from user-annotated acceleration data', *Pervasive Computing* pp. 1–17.
- Ermes, M., Parkka, J., Mantyjarvi, J. & Korhonen, I. (2008), 'Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions', *IEEE Transactions on Information Technology in Biomedicine* **12**(1), 20–26. 0205.
- Hunt, D. (2009), A heuristic method to distinguish horse rider mounts using a single wrist mounted inertial sensor, Master's thesis, Auckland University of Technology, Auckland, New Zealand.
- Ilies, I., Jaeger, H., Kosuchinas, O., Rincon, M., Sakenas, V. & Vaskevicius, N. (2007), 'Stepping forward through echoes of the past: forecasting with Echo State Networks'.
URL: http://www.neural-forecasting-competition.com/downloads/methods/27-NN3_Herbert_Jaeger_report.pdf
- Jaeger, H. (2001), The "echo state" approach to analysing and training recurrent neural networks, Technical report, Fraunhofer Institute for Autonomous Intelligent Syst.
- Jaeger, H. & Haas, H. (2004), 'Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication', *Science* **304**(5667), 78–80.
- Jaeger, H., Lukosevicius, M., Popovici, D. & Siewert, U. (2007), 'Optimization and applications of echo state networks with leaky- integrator neurons', *Neural Networks* **20**(3), 335–352.
- Lee, S.-W. & Mase, K. (2002), 'Activity and location recognition using wearable sensors', *IEEE Pervasive Computing* **1**(3), 24–32. 0303.
- Lester, J., Choudhury, T., Kern, N., Borriello, G. & Hannaford, B. (2005), A hybrid discriminative/generative approach for modeling human activities, in 'IJCAI', Vol. 5, pp. 766–772. 0271.
- Lukoševičius, M. (2012), A practical guide to applying echo state networks, in 'Neural Networks: Tricks of the Trade', number 7700 in 'Lecture Notes in Computer Science', Springer, Bremen, Germany, pp. 659–686. 0003.
- Lukoševičius, M. & Jaeger, H. (2009), 'Reservoir computing approaches to recurrent neural network training', *Computer Science Review* **3**(3), 127–149.
- Lukowicz, P., Ward, J., Junker, H., Stäger, M., Tröster, G., Atrash, A. & Starner, T. (2004), 'Recognizing workshop activity using body worn microphones and accelerometers', *Pervasive Computing* pp. 18–32.
- Plötz, T. (2010), How to do good research in activity recognition, in 'How To Do Good Research In Activity Recognition; Workshop in conjunction with Pervasive 2010', Heksinki, Finland, p. 4.
- SparkFun Electronics Inc (2008), 'IMU 6 degrees of freedom - v4 with bluetooth capability', Web.
URL: <http://www.sparkfun.com/products/8454>
- Verplaetse, C. (1996), 'Inertial proprioceptive devices: self-motion-sensing toys and tools', *IBM Systems Journal* **35**(3), 639–650. 0124.
- Verstraeten, D., Schrauwen, B. & Stroobandt, D. (2006), Reservoir-based techniques for speech recognition, in 'Proceedings of the International Joint Conference on Neural Networks, IJCNN', pp. 1050–1053.
- Ward, J. A., Lukowicz, P. & Troster, G. (2006), Evaluating performance in continuous context recognition using event-driven error characterisation, in 'Location and Context-Awareness', Springer, pp. 239–255.
- Ward, J., Lukowicz, P., Troster, G. & Starner, T. (2006), 'Activity recognition of assembly tasks using body-worn microphones and accelerometers', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **28**(10), 1553–1567.

A Comparative Study of RFID Technology Measuring Efficiency and Acceptance when Capturing Attendance

Steven Tucker

Peter Darcy

Bela Stantic

School of Information and Communication Technology
Griffith University
Gold Coast QLD Australia

Email: Steven.Tucker@griffithuni.edu.au, {P.Darcy, B.Stantic}@griffith.edu.au

Abstract

The use of Barcodes and Radio Frequency Identification (RFID) technology has become a ubiquitous means of inventory and asset tracking. When considering the application of monitoring people in an enclosed environment, for example in a classroom or an examination setting, previously employed RFID-enabled solutions have yielded high costs and poor user acceptance. Previous studies have also shown that an important factor which has impacted adoption is privacy issues surrounding Ultra High Frequency (UHF) systems. In this paper, we compare attendance recording techniques and technologies to determine the optimum method focusing on the price, efficiency and user acceptance. The three approaches we have examined include manual recording, barcode scanning and Low-Frequency RFID capturing over a fixed period conducted as a technology integration pilot study. From our initial results, we have found that a low cost RFID reader and tags approach is most favoured for user acceptance, drastically reduced the recording time compared to manual methods and is comparative to the cost of barcode systems.

Keywords: RFID, User Acceptance, Efficiency.

1 Introduction

The process of capturing attendance has in the past been a manual process of calling out names and awaiting a response which resulted in unacceptable consumption of time. To more efficiently address the process, both Barcode and Radio Frequency Identification (RFID) technologies have been employed to manage the task. Barcode technology uses a printed series of lines in conjunction with an optical reader to determine an items unique identifier while RFID uses wireless technology between a reader and the tag being interrogated. While barcode systems have have the potential to be more efficient than a manual roll call, however in past studies concerns that have been raised included the technologies requirement for line-of-sight, if a card is partially covered, angled or worn it can often result in misreads. RFID systems that have been deployed have often opted for long range Ultra-High Frequency (UHF) devices

which have not only been costly but raised concerns around privacy as the technology continually makes readings, effectively tracking participants (Pateriya & Sharma 2011).

In this paper we will compare and analyse three methods and technologies used for attendance recording to identify the optimum method in consideration of speed, user acceptance and cost. The three methods trialled were Manual Roll Call, Barcode reading and RFID scanning. The architecture was made up of a database driven web application, client machines with a web browser and the Barcode and RFID readers. Readers were attached via usb to the client computer which was logged into the application, upon receiving a valid read the application would record the attendees identification and current time stamp in the data base for later review and analysis.

At the conclusion of the trial our results showed the RFID technology was most favoured for user acceptance, was the most speed efficient method and was comparative to the cost of the Barcode system.

The remainder of this paper is organised in the following sequence: Section 2 provides the required background information for both the methods previously used for attendance tracking, as well as information about the technologies deployed which include Barcode and RFID technologies. Section 3 delivers a succinct account of related work previously studied which then leads to our methodology outlined in Section 4. The results and analysis of the technology integration pilot study are delivered in Section 5. Finally Section 6 discusses our conclusions on which method provided the optimum solution and the future work we intend to study following on from this research.

2 Background

Recording attendance has traditionally been a manual procedure where by either participants sign in or a process of announcing names and recording the response is undertaken during the attendance period. The time required for manual attendance monitoring procedures can be significant, particularly for large groups. Various means of speeding up the process have been suggested such as recording only a random 10% of attendees when roll taking is used as an incentive to increase attendance (Shimoff 2001). When the full data is required however, such is the case for statistical analysis, electronic means of recording have been utilised to reduce not only the time required for recording attendance, but also to streamline or even eliminate post recording work such as data entry. The technologies we will be examining in this research are Barcode and RFID technologies.

2.1 Barcode Technology

Barcode Technology incorporates a printed series of solid lines and spaces which are interpreted by a reader into a string of alphanumeric values to identify an object. The Barcode Reader is most often a laser scanner where the light reflected back off the printed Barcode is interpreted by the reader. The solid lines reflect less light, while the spaces reflect more light (Woodland et al. n.d.), (Sriram et al. 1996).

Barcode Technology is a familiar and accepted technology largely due to its almost universal use in supermarkets and retail in general. Barcode technology is used across a vast number of applications from personal identification tags to inventory management to tracking items in transit.

2.2 RFID Technology

Radio Frequency Identification (RFID) incorporates the wireless transfer of data between a reader and one or more identifying tags (Want 2006). When entering the range of a reader, a tags Electronic Product Code (EPC) is reported back to the reader to identify the tag (Chawathe et al. 2004). RFID Technology can be divided into short and long range systems. Low Frequency (LF, 125-134 KHz) and High Frequency (HF, 13.56 MHz) systems are short range, whereas Ultra-high frequency (UHF, 860-960 MHz) is long range (Nikitin et al. 2007). The range of the RFID interrogation zone spans 3 meters to 100 meters depending on the type of tag and reader employed (Chawathe et al. 2004).

The primary purpose of RFID Technology is to accurately and efficiently identify an object using its associated tag. RFID is employed across a wide range of industries and use cases such as Stock Management in the Retail sector, Smart Cards for Financial transactions and Human Identification items such as passports (Ilie-Zudor et al. 2011).

2.3 Related Work

Systems have been implemented and evaluated using Barcode or RFID to automate attendance tracking with the goal of increasing efficiency. Barcode systems use statically located readers or (often several) portable hand-held units, in both cases it is typically the subjects themselves handling the scanning. It has been observed that statically located readers have caused bottlenecks when there are many attendees all attempting to register a scan, this is exacerbated by failed reads due to the Barcode systems line-of-sight requirement, where any covering of or incorrectly aligned Barcode would result in a misread. Portable units raised concerns such as possible theft, absentees providing their Barcode to attendees to scan, and misuse of a reader for scanning unrelated Barcodes (Casey & Kille 2011). The bottleneck problem could be reduced by increasing the number of available readers, this would however reduce the ability to monitor the scanning to eliminate false scans as well as increase the expense of deployment.

RFID Technology has been deployed using short and long range systems for use in automated attendance tracking, though long range systems are preferred as they can scan large numbers of tags autonomously (Chang 2011).

Long range UHF systems have a number of problems which has discouraged wide scale adoption. Some of these problems include the high cost of equipment, possible signal interference, reading collisions and multiple reads of the same tag. (Wu et al. 2006)

Long range systems also have greater Privacy Concerns than short range. UHF Long range systems would have readings of attendees for their duration within the read area, and thus act to track an attendee rather than the approach of a Short range system which simply takes a snapshot at a given time (Silva et al. 2008).

A Short Range RFID system coupled with secure communication and strict server security does not suffer the same privacy concerns as a Long Range system.

2.4 Motivation

It is often desirable to collect attendance data for an event, class or exam, whether for further analysis, reflection and planning or to encourage greater attendance. Collecting this data for large groups can be tedious and can cause unacceptable disruption and consumption of precious time. The goal of our technology integration pilot study is to identify the optimum method for capturing attendance, balancing efficiency to address the time constraint, user acceptance of the technology to encourage participation and to keep the cost of deployment to a minimum. Our pilot study concentrated on three methods which met some basic constraints of the environment and which could be deployed simply, cheaply and immediately.

3 Methodology

In this paper we will compare the results of our technology integration pilot study to determine the optimum method by which to record attendance. The recording methods analysed are Manual Roll Call, Barcode scanning and Low-Frequency short range RFID capturing. To determine the optimum, each method is analysed on the basis of three key measurements, efficiency, user acceptance and cost. The following section outlines the motivation of our analysis, after which the systems architecture and requirements are described.

3.0.1 Manual Roll Taking

The traditional method of manual roll taking met all environmental and cost constraints, however the disruption and time consuming nature of this approach has often meant that it has been unpopular for both the attendees and the coordinator. Regardless of its short comings, Manual Roll Taking was included in the study to provide a base line for comparison. Any alternative method must at minimum have a greater efficiency and user acceptance to be considered viable.

3.0.2 Barcode Technology

The use of Barcode Technology for the study was a natural choice due to the low cost of the scanning equipment coupled with the existing Barcodes allocated to each student that appear on their student identification cards. Although a single scanning unit can cause bottlenecks with larger classes, the requirement of supervision to prevent problems discussed earlier, such as absentees providing their barcode to attendees to register attendance, resulted in the use of a single scanning point at a very low cost and reduced the likelihood of mistreatment or breakage of the equipment.

User acceptance of Barcode Technology was expected to be high as the identification cards Barcodes were already in use by the students and so would be

familiar. It was expected that the rate of scanning would be more efficient than traditional roll call.

3.0.3 Short range RFID Technology

The use of RFID Technology was trialled primarily on the basis of efficiency. As RFID does not require line of site to obtain a reading, it was thought that overall read time would be shorter as attendees would be less likely to encounter read issues. Short range Low Frequency equipment was selected over Long range Ultra High Frequency due to cost, potential privacy issues and possible user acceptance problems as previously discussed.

The cost of an RFID reader was comparable to the cost of the Barcode scanner, however tags had to be purchased and distributed to all the students. While the tags supplied were low cost, it may not be necessary in future to supply tags as students cards may be distributed with RFID chips to be used for security access. As is the case with the Barcode system, if the student cards are issued with the required technology then there will not be an extra expense to provide tags for the purpose of tracking attendance.

It was not clear what level of user acceptance would be attained with RFID compared to a bar code system, and so this Technology was of particular interest in the trial.

3.1 Architecture

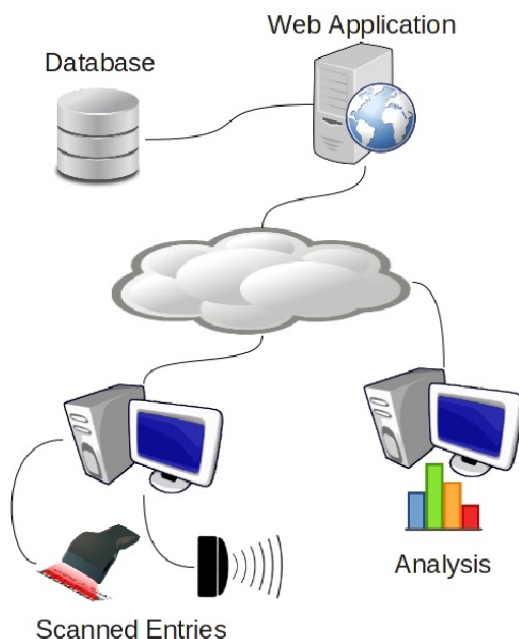


Figure 1: Architecture overview

As shown in Figure 1, the system architecture is made up of client machines connecting via a network to a web based application. In the classroom the client machine reads the attendees identification through either Barcode code or RFID technologies, the reading is then sent to the web application which in turn writes the record to the database including both the attendee and the current time. Out side of the class room, a client machine is used to access the web application to retrieve real time reports and analysis on the attendance data. To ensure secure communication between the clients and the server, the application is password protected and runs over a Secure Socket Layer (SSL).



Figure 2: Barcode reader



Figure 3: RFID scanner and tag

3.2 Assumptions

We have made three assumptions regarding the trial processes. The first assumption is that each attendee will have with them their student identification card or assigned RFID tag as required in class. The second assumption is that the organiser will have in their possession either the Barcode Reader or the RFID Scanner as needed. The last assumption is that there is access to a network connected computer with a web browser at the venue so that it is possible to connect with the web application.

4 Experimental Results and Analysis

The following section describes the framework we assembled to undertake the technology pilot study as well as the methodology followed. To begin with we look at the environment in which the Technologies operated, next we detail the results of each method in the context of efficiency, user acceptance and cost.

4.1 Environment

Our web based application was coded using a Linux, Apache, MySql and Php (LAMP) stack. The server Operating System is the Debian GNU/Linux distribution Release 6 "Wheezy". The Operating System for the client machines was Windows XP with Service Pack 3 installed. Both the Barcode Code Scanner and the RFID Reader operated as standard input devices running over USB. The RFID Reader and tags operate at 125Khz. The web browsers used to access the web application were Firefox and Chrome.

4.2 Efficiency

A key measurement for determining the optimum method for attendance recording is undoubtedly speed efficiency. For a technology to be considered for adoption it must at minimum have a far greater time

efficiency than manual attendance recording. Each method trialled had its capture duration recorded for two lectures with a class of approximately 60 students, the averaged results of which are represented in Figure 4.

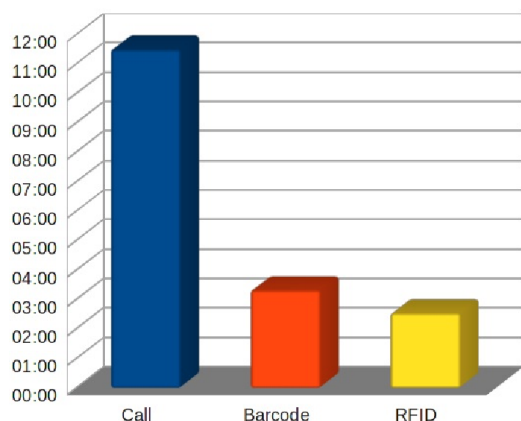


Figure 4: The aggregated results of the recording duration of each method

4.2.1 Manual Roll Call

The duration of Roll Call was determined by manually recording the time of commencement and of completion. The times were recorded by a second person so as not to interfere with or allow bias by the roll taker. It is important to note that while the average time taken to record attendance using this method came to 11 Minutes 30 Seconds, an extra task of data entry was performed after each class, and while this did not effect the class, it had the impact of effectively doubling the attendance tracking effort. The results show that the time needed for manual roll call has a significant impact on the time remaining for the class and as such is rarely adopted for large audiences.

4.2.2 Barcode Technology

The results show a vast improvement in time efficiency over Manual Roll Call with an average recording duration of 3:19 Seconds. The duration of Barcode approach was retrieved by database queries calculating the difference between the time stamps of the first and last reads, the scanning required students to approach the the reader at the front of the room and present their barcode. The capture time for each record was fairly consistent for Barcodes, while Manual Roll Call varied greatly. The varying times for Manual Roll Call can be attributed to slow responding attendees and absentees. A call is repeated a number of times when there is no response to ensure that the person is absent rather than having not heard the call. Also the callers often encounter names they find difficult to pronounce, and this also causes delays which the bar code system is immune from.

4.2.3 RFID Technology

RFID claimed the greatest time efficiency with an average recording duration of Two Minutes Thirty Two Seconds. Like the barcode system, students were required to approach the reader at the front of the room, this time however they passed their tag over the reader. The performance increase over Barcode reading is a result of not requiring line of sight and hence encountering less misreads.

4.3 Technology Adoption

As part of the technology integration pilot study a survey was conducted to measure the level of user acceptance for each method. The survey used a Likert-type approach with a scale of 1 to 5 where 1 is strongly disagree and 5 is strongly agree. There were twelve questions used for evaluation, the aggregated results of which are shown in Figure 5. The survey questions were divided into four areas, Perceived Usefulness, Perceived Ease of Use, Intention to Use and Attitude Towards Usage. The survey includes questions such as "Using ... enables me to get my attendance checked quicker" and "Overall, I find ... very useful" for each of the approaches given a grade between 1 and 5. Additionally, a space was provided at the end for any additional comments regarding the technology.

The survey showed a significant bias in all categories in favour of RFID, even though in this trial it required attendees to manage an extra item (the rfid tag). Interestingly attendees still rated RFID higher for ease of use than Barcode Technology which they were already familiar with from existing use. As part of the survey, space was provided for general feedback, of those that responded the majority reported that it was primarily the speed of RFID that was appealing. A number of participants also commented that there would be increased convenience if the RFID tag was integrated into the student card so they only had one item to manage.

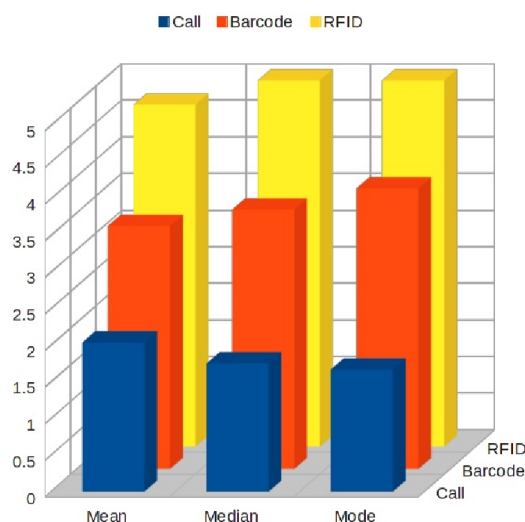


Figure 5: The aggregated results of the Likert-style survey for user acceptance of attendance recording methods

4.4 Cost

4.4.1 Barcode Technology

The only purchase required to trial the Barcode Technology was a Scanner as students had already been supplied with a unique bar code by way of their student cards. We purchased the 80mm USB Barcode Scanner shown in Figure 2 for AUD\$24.92

4.4.2 RFID Technology

To trial the RFID Technology we purchased both a reader and a sufficient number of tags. The reader as shown in Figure 3 is a 125Khz EM4100 RFID USB Proximity ID Card Chip Reader purchased

for AUD\$9.60. The RFID tags were purchased for AUD\$0.27 each (\$26.90 per pack of 100).

5 Conclusion

In this paper we have carried out a comparative analyses on three methods of attendance capturing, Manual roll call, Barcode and RFID technologies. The focus for comparison was based on three key indicators, speed efficiency, user acceptance and cost (including equipment and deployment costs). Our results have shown RFID Technology performed greatest for both speed efficiency as well as user acceptance while remaining affordable with a cost of deployment comparative to that of Barcode technology.

To further leverage the benefits of an optimal attendance capturing technology, it would be interesting in future work to analyse the captured data for useful applications such as attendance prediction for class planning, and recognising attendance patterns to aid content delivery planning.

References

- Casey, M. M. & Kille, M. W. (2011), 'Effective Administration of a Large Undergraduate Physics Class: From Enrolment to Assessment Feedback', *ArXiv e-prints*.
- Chang, C. H. (2011), Smart classroom roll caller system with iot architecture, in 'Innovations in Bio-inspired Computing and Applications (IBICA), 2011 Second International Conference on', pp. 356–360.
- Chawathe, S. S., Krishnamurthy, V., Ramachandran, S. & Sarma, S. E. (2004), Managing RFID Data, in 'VLDB', pp. 1189–1195.
- Ilie-Zudor, E., Kemny, Z., van Blommestein, F., Monostori, L. & van der Meulen, A. (2011), 'A survey of applications and requirements of unique identification systems and rfid techniques', *Computers in Industry* **62**(3), 227 – 252.
- Nikitin, P., Rao, K. & Lazar, S. (2007), An overview of near field uhf rfid, in 'RFID, 2007. IEEE International Conference on', pp. 167–174.
- Pateriya, R. K. & Sharma, S. (2011), The Evolution of RFID Security and Privacy: A Research Survey, in 'Communication Systems and Network Technologies (CSNT), 2011 International Conference on', pp. 115–119.
- Shimoff, E. (2001), 'Effects of recording attendance on grades in introductory psychology', *Teaching of psychology* **28**(3), 192.
- Silva, F., Filipe, V. & Pereira, A. (2008), Automatic control of students' attendance in classrooms using rfid, in 'Systems and Networks Communications, 2008. ICSNC '08. 3rd International Conference on', pp. 384–389.
- Sriram, T., Vishwanatha Rao, K., Biswas, S. & Ahmed, B. (1996), Applications of barcode technology in automated storage and retrieval systems, in 'Industrial Electronics, Control, and Instrumentation, 1996., Proceedings of the 1996 IEEE IECON 22nd International Conference on', Vol. 1, pp. 641–646 vol.1.
- Want, R. (2006), 'An introduction to rfid technology', *Pervasive Computing, IEEE* **5**(1), 25 – 33.
- Woodland, N. J., Ventour, N. J. & Silver, B. (n.d.), 'Classifying Apparatus and Method. Patent 1952. US 2612994.'
- Wu, N., Nystrom, M., Lin, T. & Yu, H. (2006), 'Challenges to global rfid adoption', *Technovation* **26**(12), 1317 – 1323.

A Trigger Counting Mechanism for Ring Topology

Sushanta Karmakar¹

Subhrendu Chattopadhyay¹

¹ Department of Computer Science and Engineering,
Indian Institute of Technology Guwahati, India, 781039
Email: sushantak@iitg.ernet.in, subhrendu@iitg.ernet.in

Abstract

Consider a distributed system with n processors, which receive triggers from the outside world. The Distributed Trigger Counting (DTC) problem is to raise an alarm if the number of triggers over the system reaches w , which is an user specified input. DTC is used as a primitive operation in many applications, such as distributed monitoring, global snapshot etc. In this paper, we propose an algorithm for the DTC problem in a ring topology with a message complexity of $O(n^2 \log(w/n))$ and each node in the system receives $O(n \log(w/n))$ number of messages. We also discuss about the possible tuning of the algorithm which results better complexities.

Keywords: Distributed algorithm, distributed monitoring, distributed trigger counting, ring topology

1 Introduction

Distributed trigger counting (DTC) is an important problem in distributed systems. Consider a distributed system with n processes. Each process receives some triggers (signals) from an external source. The DTC problem is to detect the state when the number of triggers received by the system reaches w which is a user defined input to the system. Note here that w may be much larger than n . The sequence of processors receiving the w triggers is not known a priori to the distributed system. Our goal is to propose an algorithm for the DTC problem under a known topological setting. More specifically, in this paper we propose an algorithm for the DTC problem in a ring network.

The DTC problem arises in many applications in distributed systems. Some major application areas can be distributed monitoring, global snapshot etc. Monitoring is an important aspect in wireless networks as well as in wired networks. A wireless sensor network is typically used to monitor physical or environmental conditions such as border surveillance, forest fire detection, traffic management in highways etc. In traffic management, one may be interested in detecting whether the number of vehicles on a highway exceeds a certain threshold. Similar applications may be found in border surveillance and forest

fire detection as well. In distributed system a global snapshot state is said to be valid provided all the in-transit messages are recorded. It was shown by Garg et al. (2006) that the problem of detecting whether all the in-transit messages have been recorded can be reduced to the DTC problem. Awerbuch (1985) proposed the concept of synchronizers which is a tool to transform a synchronous distributed algorithm to another that runs on an asynchronous distributed system. Here a distributed system is required to generate a signal (or pulse) when all the messages generated after the previous signal have been delivered. This problem can also be formulated as a variant of the DTC problem. The performance of an algorithm for the DTC problem is often measured by the following two metrics.

- *Message complexity:* It is the total number of messages sent by all the nodes of the systems.
- *MaxRcvLoad:* It is the maximum number of messages received by any node in the system.

The DTC problem was studied by Garg et al. (2006) for a general distributed system. In their work, they proposed two algorithms for the DTC problem: a centralized algorithm and a tree-based algorithm. The centralized algorithm has a message complexity of $O(n \log w)$. Its MaxRcvLoad has a complexity of $O(n \log w)$. This is a tight bound for the centralized algorithm. The tree based algorithm has a message complexity of $O(n \log n \log w)$. Again the MaxRcvLoad of the tree based algorithm has a complexity of $O(n \log n \log w)$. In the work the authors also showed that any deterministic algorithm for the DTC problem must have a message complexity of $\Omega(n \log(w/n))$. Huang et al. (2007) proposed a novel solution to the problem of efficient detection of an aggregate predicate over cumulative triggers over a time-varying window in a distributed monitoring system. They provided a queueing theory based analysis of the problem which provides the user the power to trade-off between desired accuracy and communication overhead. However, their approach is based on a single coordinator. Hence the algorithm is not fully distributed. In another work (Chakaravarthy, Choudhury, Garg & Sabharwal 2011) a randomized distributed algorithm was proposed for trigger counting in a tree based topology. The algorithm has a message complexity of $O(n \log n \log w)$ and MaxRcvLoad complexity of $O(\log n \log w)$ with high probability. Later in another work (Chakaravarthy, Choudhury & Sabharwal 2011) the authors proposed an improved algorithm for the DTC problem having message complexity of $O(n \log w)$ and MaxRcvLoad of $O(\log w)$. However, this work has the limitation that the algorithm is a randomized algorithm and it does not provide any bound on the messages sent per node.

Copyright ©2014, Australian Computer Society, Inc. This paper appeared at the Thirty-Seventh Australasian Computer Science Conference (ACSC2014), Auckland, New Zealand, January 2014. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 147, Bruce H. Thomas and David Parry, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.

Table 1: Summary of DTC algorithms

<i>Algorithm</i>	<i>Messages</i>	<i>MaxRcvLoad</i>	<i>Nature</i>
Tree-based (Garg et al. 2006)	$O(n \log n \log w)$	$O(n \log n \log w)$	randomized
Centralized (Garg et al. 2006)	$O(n \log w)$	$O(n \log w)$	centralized
(Chakaravarthy, Choudhury, Garg & Sabharwal 2011)	$O(n \log n \log w)$	$O(\log n \log w)$	randomized
(Chakaravarthy, Choudhury & Sabharwal 2011)	$O(n \log w)$	$O(\log w)$	randomized
(Emek & Korman 2010)	$O(n(\log n \log w)^2)$	$O((\log n \log w)^2)$	deterministic
Unidirectional Ring (this paper)	$O(n^2 \log \frac{w}{n})$	$O(n \log \frac{w}{n})$	deterministic

Also this algorithm assumes a clique of n nodes as the topology. In fact they concluded that designing a deterministic algorithm with message complexity of $O(n \log w)$ and $MaxRcvLoad$ of $O(\log w)$ for any arbitrary network is an open DTC problem.

In this paper, we propose a deterministic distributed algorithm for the DTC problem in a unidirectional ring network. A ring of nodes is an interesting topology and is used in solving many distributed computing problems such as leader election, mutual exclusion, distributed snapshot etc. Also if a solution of a problem in distributed computing can be obtained for a ring topology then the same can be applied to have a solution for the same problem under arbitrary topology by means of embedding a virtual ring topology over the network (e.g. Eulerian graph). Therefore for many reasons a ring topology is important and in this paper we propose a solution to the DTC problem assuming a ring topology. We also provide an outline of how the DTC problem can be solved in a general network using our proposed solution to the problem in a ring network.

The main challenge of the DTC problem is that outside triggers may be detected by any node and one would like to minimize the communication overhead for determining when a total of w external triggers have been counted. It is important to see that there is a tradeoff between minimizing the communication overhead and having a timely detection of the w triggers. The simplest way to minimize the number of messages is to let every node detect triggers and when any one has detected w triggers, the global alarm signal is sent. However, in this simple scheme there is obviously a big risk that the system will be seriously delayed in sending the global alarm signal which has to be sent as soon as w or more triggers are received by all the nodes. The obvious way to avoid such delays is to send a message as soon as a trigger is detected. However this will generate a lot of messages (potentially $O(w)$). Hence there should be some compromise between these two extremes. In an earlier work (Garg et al. 2006) it was shown that any deterministic algorithm for the DTC problem must have a message complexity lower bound of $\Omega(n \log(w/n))$. However this result was analytical and no algorithmic instance has been found so far satisfying this lower bound.

Let there be n nodes in a unidirectional ring. The nodes are denoted as $v_1, v_2, v_3, \dots, v_n$. Any node v_i can send a message to v_{i+1} and receive a message from v_{i-1} . Specifically, v_n sends a message to v_1 . We assume an asynchronous model of computation and communication (via messages). We assume that the channels are reliable and FIFO. There is no node or link failure. Also messages are not corrupted or spuriously introduced. It is assumed that each node has a unique identifier. Out of the n nodes, there is one node v_n designated as the *master*. All other nodes (v_i such that $i \neq n$) act as slaves. The algorithm is stated in the form of guarded statements. A guarded statement is of the form $g \rightarrow a$ where g is the guard, and a is the action. The action a is executed if and

only if g is true. The program for any node i contains a sequence of statements $\{S_1, S_2, \dots, S_n\}$ where each S_j is of the form $g_j \rightarrow a_j$. The j -th statement of the program at node i is denoted by $S_j(i)$. The guard corresponding to $S_j(i)$ is denoted by $g_j(i)$ and the action corresponding to $S_j(i)$ is denoted by $a_j(i)$. Also it is assumed that the guarded actions are atomic. The main result of our algorithm is as follows. The distributed trigger counting algorithm over a ring of n processors has a message complexity of $O(n^2 \log \frac{w}{n})$ and $MaxRcvLoad$ of $O(n \log \frac{w}{n})$ where w is the number of triggers to be counted.

The rest of this paper is organized as follows. Section 2 contains the related work on the DTC problem. The proposed algorithm for the DTC problem in a unidirectional ring is presented in Section 3. Section 4 contains the analysis for proving the correctness and message complexity of the proposed algorithm. Section 5 discusses the tuning of the algorithm with respect to some parameters. The trade-off between message complexity and delay in raising the alarm is discussed in Section 6. Solving the DTC problem under arbitrary network is discussed in Section 7. Finally we conclude in Section 8.

2 Related Work

Many of the earlier works primarily consider the DTC problem in a centralized setting and solve it using randomized algorithm. A randomized algorithm was proposed by Emek & Korman (2010) for the DTC problem with message complexity of $O(n(\log \log n)^2 \log w)$ and average message complexity of $O((\log \log n)^2 \log w)$. In this paper it is assumed that the input is constructed by an adaptive adversary whose decisions may depend on previous coin tosses of the randomized protocol but not on future ones. Here a separator decomposition of a tree of n nodes is constructed over which the events are aggregated. By changing the internal parameters of the randomized protocol they have obtained a deterministic protocol with message complexity of $O(n(\log n \log w)^2)$ and $MaxRcvLoad$ of $O((\log n \log w)^2)$.

A fundamental class of problems called “thresholded counts” was introduced by Keralapura et al. (2006) where the goal is to return the aggregate frequency count of an event, that is continuously monitored by distributed nodes with a user-specified accuracy, whenever the actual count exceeds a given threshold value. They studied the cases under static as well as dynamic threshold values. Cormode et al. (2011) defined a function monitoring problem as a 4-tuple $(\kappa, f, \tau, \epsilon)$ where κ denote the number of nodes, f denote the function that is being monitored by the nodes, τ denote the threshold such that if $f \geq \tau$ then the system generates some alarm 1 and the alarm is 0 if $f \leq (1 - \epsilon)\tau$. The authors give lower and upper bounds of communication cost for the $(\kappa, f, \tau, \epsilon)$

```

MasterProcess(TriggerCount  $w$ )    // for master node  $v_n$ 

Initialization:  $TargetTriggerCount$   $w' = w$ ;  $C = 0$ ;  $sflag = true$ ;  $fsorFlag = false$ ;
                   $eorFlag = false$ ;  $\psi$ ;  $\tau$ 

( $S_1$ )     $sflag \rightarrow \psi = w'/2$ ;  $\tau = w'/2n$ 
          send  $\langle start - of - round, \tau \rangle$  to  $v_1$ 
           $fsorFlag = true$ ;  $sflag = false$ 

( $S_2$ )    upon receiving  $\langle Trigger \rangle \wedge fsorFlag \rightarrow C = C + 1$ 

( $S_3$ )    upon receiving  $\langle Coin, factor \rangle \wedge fsorFlag \rightarrow C = C + factor \times \tau$ 

( $S_4$ )     $fsorFlag \wedge (C \geq \psi) \rightarrow efactor = 0$ 
          Send  $\langle end - of - round, efactor \rangle$  to  $v_1$ 

( $S_5$ )    upon receiving  $\langle end - of - round, efac \rangle$  from  $v_{n-1} \rightarrow$ 
           $w' = w' - (C + efac \times \tau)$ 
          if  $(w' > 0)$  then  $sflag = true$ 
          else  $sflag = false$ 
               $fsorFlag = false$ 
              send  $\langle TargetReached \rangle$  to  $v_1$ 
    
```

Figure 1: Pseudocode of master node for DTC algorithm for a ring

problem with different f . The algorithms proposed by the authors are randomized. One example of f can be the aggregate function. Here each input trigger i is associated with a value α_i . The goal is to raise an alarm when the aggregate of these values crosses a threshold.

A decentralized and randomized algorithm called LayeredRand algorithm was proposed by Chakaravarthy, Choudhury, Garg & Sabharwal (2011). In this work, the nodes are organized in a tree topology and they communicate only with the nodes in the adjacent layers. The algorithm proceeds in multiple rounds and in each round it achieves a trigger count which is approximately half of the required count. In another work, Chakaravarthy, Choudhury & Sabharwal (2011) proposed an approximate algorithm with the assumption $w \leq 2^n$, and it exhibits a message complexity of $O(n \log n \log w)$ and MaxRcvLoad of $O(\log n \log w)$. Similar works were done in (Korman & Kuten 2007, Emek & Korman 2011). No prior work on the DTC problem considered a ring as the underlying topology.

3 Algorithm for DTC in a Unidirectional Ring

In this section, we describe an algorithm for the DTC problem in a unidirectional ring topology. Its message complexity is $O(n^2 \log(w/n))$ and MaxRcvLoad is $O(n \log(w/n))$. We assume that the triggers which come from the outside world are distributed to the n nodes of the ring randomly. The distribution of the triggers among n nodes is arbitrary and not known apriori. The algorithm consists of a number of rounds. The total number of triggers to be counted by the system is denoted by w . The system has n number of nodes connected in a logical unidirectional ring configuration. The nodes in the ring are denoted as v_i ($1 \leq i \leq n$). There is one node v_n which is designated as the *master* node. All the other nodes in the ring are called *slaves*. Each *slave* node has the following state variables: C is a counter indicating the number of received triggers that have not yet contributed to the distributed trigger counting and initialized to 0; $fsorFlag$ is a boolean flag which indicates the start

of the DTC algorithm, and it is initialized to *false*; $cflag$ is a boolean flag which indicates whether a node has received a *Coin* message, and it is initialized to *false*; $coinCount$ is an integer that denotes the number of *Coin* messages sent by the node; $eorFlag$ is a boolean flag that indicates the end of the current round and it is set to *true* on receipt of a *end-of-round* message. For the *master*, there are some other state variables in addition to the aforesaid variables. They are: w' is the remaining number of triggers yet to be counted by the system and is initialized to w ; $sflag$ is the start of round flag for each round and it is initialized to *true*. The *master* node also maintains two variables, ψ and τ , which are known as *global threshold* and *local threshold*. Each slave node gets the value of τ for the current round from its previous node in the ring. To reduce the number of messages, each slave node sends a *Coin* message only when its trigger count crosses the local threshold, τ . Similarly the *master* node sends a *end-of-round* message only when the total trigger count at the master crosses its global threshold, ψ .

For any node v_i , v_{i-1} is called the predecessor of v_i and v_{i+1} is the successor of v_i . Note that the predecessor of v_n is v_1 . Similarly the predecessor of v_1 is v_n . The trigger counting algorithm proceeds in multiple rounds. At the start of each round, the system should know the number of triggers which are yet to be counted to raise an alarm. This can be known by subtracting the sum of all the triggers received in the previous round from w' . Each round of the algorithm is started by the *master* node v_n by sending the *start-of-round* message to v_1 . Node v_n calculates ψ and τ , and sends τ to v_1 along with the *start-of-round* message. It updates its state variables as follows: $fsorFlag(v_n) = true$ and $sflag(v_n) = false$. When a *slave* v_i gets $(start-of-round, \tau')$ message it also calculates its local threshold as $\tau = \tau'$. It sets its local state variables as follows: $fsorFlag(v_i) = true$, $eorFlag(v_i) = false$. Since it has not yet sent any *Coin* message, v_i sets $coinCount(v_i) = 0$. It also sends a $(start-of-round, \tau)$ message to v_{i+1} .

With $fsorFlag(v_i) = true$ ($1 \leq i \leq n$), any node v_i increments its local counter $C(v_i)$ on receiving a *Trigger* message. So each node v_i independently


```

SlaveProcess()           // code for node  $v_i$  where  $i \neq n$ 

Initialization:  $C = 0$ ;  $fsorFlag = false$ ;  $cflag = false$ ;  $\tau$ ;  $coinCount = 0$ ;  $factor$ ;
                   $eorFlag = false$ 

( $S_6$ ) upon receiving  $\langle start - of - round, \tau' \rangle \rightarrow \tau = \tau'$ 
      Send  $\langle start - of - round, \tau \rangle$  to  $v_{i+1}$ 
       $coinCount = 0$ 
       $fsorFlag = true$ ;  $eorFlag = false$ 

( $S_7$ ) upon receiving  $\langle Trigger \rangle \wedge fsorFlag \rightarrow C = C + 1$ 

( $S_8$ ) upon receiving  $\langle Coin, fac \rangle \wedge fsorFlag \wedge \neg cflag \rightarrow C = C + fac \times \tau$ 
       $cflag = true$ 

( $S_9$ )  $fsorFlag \wedge (C \geq \tau \vee cflag) \wedge \neg eorFlag \wedge coinCount \leq n \rightarrow factor = C/\tau$ 
      Send  $\langle Coin, factor \rangle$  to  $v_{i+1}$ 
       $C = C - factor \times \tau$ 
       $coinCount = coinCount + 1$ 
       $cflag = false$ 

( $S_{10}$ ) upon receiving  $\langle end - of - round, efactor \rangle \rightarrow$ 
       $eorFlag = true$ 
       $factor = C/\tau$ 
       $C = C - factor \times \tau$ 
       $efactor = efactor + factor$ 
      Send  $\langle end - of - round, efactor \rangle$  to  $v_{i+1}$ 

( $S_{11}$ ) upon receiving  $\langle TargetReached \rangle \rightarrow fsorFlag = false$ 
      send  $\langle TargetReached \rangle$  to  $v_{i+1}$ 

```

Figure 2: Pseudocode of slave node for DTC algorithm for a ring

counts the number of triggers it receives. Whenever v_i finds that $C(v_i) \geq \tau$ or $cflag(v_i) = true$ then it sends a *Coin* message to v_{i+1} ($i \neq n$). Note here that $cflag(v_i) = true$ only if v_i has received a *Coin* message from its previous node in the ring. In this case v_i computes the $factor = C/\tau$ and sends the $factor$ along with the *Coin* message to its successor node v_{i+1} in the ring. So a *Coin* message essentially transfers $\tau \times factor$ amount of triggers from v_i to the successor v_{i+1} . Node v_i also reduces the value of its counter C appropriately and remembers the send of a *Coin* message by incrementing the variable $coinCount$. Then it sets $cflag(v_i) = false$. It is clear from S_9 in the pseudocode of Figure 2 that a slave can send at most n number of *Coin* messages in a round of the algorithm.

If a node v_i ($i \neq 1$) receives a *Coin* message from its predecessor and $fsorFlag(v_i) = true$ and $cflag(v_i) = false$ then v_i increments its local trigger count $C(v_i)$ by $\tau \times fac$, where fac is the multiplicative factor received along with the *Coin* message. Also v_i sets $cflag(v_i) = true$. Therefore *Coin* messages move from one node to another and eventually reaches the master node v_n . When the master receives a *Coin* message, v_n increments $C(v_n)$ by $\tau \times fac$, where fac is the the multiplicative factor that v_n received from its predecessor. When the master finds that $C(v_n) \geq \psi$ then v_n initiates the end of the round by sending a *end-of-round* message to v_1 . During the propagation of the *end-of-round* message from one node to another, nodes continue to receive triggers from the outside world. These triggers are counted in a similar way.

If a slave v_i receives (*end-of-round, efactor*) message then it first set $eorFlag(v_i) = true$ such that no further *Coin* message is sent in this round. It computes $factor = C/\tau$ and adds this with the received multiplicative factor $efactor$, and sends this modi-

fied $efactor$ to v_{i+1} along with a *end-of-round* message. In this way the (*end-of-round, efactor*) message moves from one slave to another and eventually reaches the master. When the master receives (*end-of-round, efac*) message from v_{n-1} it computes the remaining number of triggers yet to be counted in future rounds. This is given by $w' = w' - (C(v_n) + efac \times \tau)$. If the remaining number of triggers yet to be counted is more than zero then v_n sets its variable $sflag(v_n) = true$ so that it can start another round (by S_1 of Figure 1).

4 Correctness and Analysis

Lemma 1. A slave node sends at most n number of *Coin* message in each round.

Proof. By S_9 , a node v_i sends a *Coin* message to v_{i+1} if the G_9 is true. Again G_9 is true if $coinCount \leq n$. By S_6 , $coinCount = 0$ at the start of each round. Also by S_9 , if v_i sends a *Coin* message to v_{i+1} then it increments its $coinCount$. Hence in each round, v_i can send at most n number of *Coin* messages. \square

Lemma 2. If master node v_n gets the (*end-of-round, efac*) message in a round and E is the number of triggers contributed by all nodes except v_n during the *end-of-round* phase then $E \geq 0$.

Proof. By S_4 , the master node v_n sends a (*end-of-round, efactor*) message to v_1 with $efactor = 0$. Any node v_i , on receiving a (*end-of-round, efactor*) message, computes its own contribution as $factor = C/\tau$. It then adds this $factor$ with the $efactor$ and forwards the (*end-of-round, efactor*) message to v_{i+1} with the modified $efactor$ value. Therefore eventually v_n will receive the (*end-of-round, efac*) message from v_{n-1} . Note here that the contribution of triggers

by v_n itself in this phase is done through the variable $C(v_n)$. Hence $E = efac \times \tau$. Since $efac \geq 0$ and $\tau \neq 0$ therefore $E \geq 0$. \square

Lemma 3. *In each round, at least $\psi = w'/2$ number of triggers are counted by the system where w' is the remaining number of triggers yet to be counted at the beginning of the round.*

Proof. By S_7 (or S_2) and S_8 (or S_3) it is clear that $C(v_i)$ is updated when v_i receives a *Trigger* or a *Coin* message. Also by S_4 , the master node v_n sends a (*end-of-round*, *efactor*) message to v_1 only if $fsorFlag(v_n) = true$ and $C(v_n) \geq \psi$. Again by Lemma 2, the number of triggers contributed by all nodes except v_n during the propagation of *end-of-round* message (from the time it is sent by v_n to the time it is received by v_n) is $E \geq 0$. Hence total number of triggers counted in a round is given by $C(v_n) + E$. Since $C(v_n) \geq \psi$, therefore the lemma is proved. \square

Theorem 1. *The distributed trigger counting takes $O(\log \frac{w}{n})$ rounds.*

Proof. Here we will analyze how the value of w is decreasing from one round to the next round. By Lemma 2, it depends on $E \geq 0$. Initially $w' = w$. By S_5 , the remaining number of triggers to be counted in future rounds is given by $w' = w - (C(v_n) + E)$. Upper bound of the number of rounds can be obtained when $E = 0$. Here w' decreases as $w, \frac{1}{2}w, (\frac{1}{2})^2w, (\frac{1}{2})^3w, \dots, (\frac{1}{2})^kw$. The number of rounds continue till $w' \geq 0$.

Let after $(r + 1)$ rounds $\tau < 1$ for the first time. So, in the r -th round $\tau \geq 1$. Therefore $\frac{(1/2)^r w}{n} \geq 1$. Therefore $r = O(\log \frac{w}{n})$. At $(r + 1)$ -th round, $\tau \leq 1$. Hence $w'/2n \leq 1$. Hence $w' \leq 2n$. Since $\tau \leq 1$, if a node gets 1 trigger then its local threshold is crossed and a *Coin* message is sent. Therefore to count $w' \leq 2n$ number of triggers, at most $2n$ *Coin* messages will be required. In worst case this will require $2n$ rounds. Hence the total number of rounds is $O(2n + \log \frac{w}{n}) = O(\log \frac{w}{n})$ since $w \gg n$. \square

Theorem 2 (Partial Correctness). *If the DTC algorithm has terminated then at least w triggers have been counted.*

Proof. Let us assume that the DTC algorithm has terminated. However at least w triggers have not yet been counted. Therefore by S_5 , $w' > 0$. So v_n sets $sflag(v_n) = true$. Therefore G_1 becomes enabled. So eventually G_6 will be enabled. However this contradicts the assumption that the algorithm has terminated. Therefore $w' \leq 0$. Hence at least w triggers will be counted. \square

Theorem 3 (Termination). *The DTC algorithm for a ring network eventually terminates.*

Proof. By Lemma 1, each slave node sends at most n number of *Coin* messages in a round. After this, no slave will send a *Coin* message in the current round even if it crosses its local threshold or receives *Coin* message from its predecessor. Also if $C(v_n) \geq \psi$ then v_n sends a *end-of-round* message to v_1 . Each slave node v_i , on receiving a *end-of-round* message, forward it to its successor. Eventually when v_n gets back a *end-of-round* message, it does not send any further

end-of-round message in the current round. Here v_n checks is $w' > 0$. By Lemma 3, eventually $w' \leq 0$ will hold. Therefore v_n will set $sflag(v_n) = false$ and $fsorFlag(v_n) = false$. Hence G_1, G_2, G_3, G_4 are all *false*. Also by S_{11} , $fsorFlag(v_i) = false$ for any slave node v_i . Hence G_7, G_8, G_9 are *false*. Since G_1 is *false*, therefore v_n does not send *start-of-round* message. Here G_6 is *false*. Similarly G_{10} and G_{11} are also *false*. Therefore all the guards are *false*. Therefore the algorithm eventually terminates. \square

Theorem 4. *The message complexity of the DTC algorithm is $O(n^2 \log \frac{w}{n})$.*

Proof. In a round, each slave node can send n number of *Coin* messages. Also each of them is forwarded by the other slaves towards the master node. Hence in each round $O(n^2)$ number of messages are sent overall by all the nodes. By Theorem 1, the algorithm takes $O(\log \frac{w}{n})$ rounds. Hence the overall message complexity is $O(n^2 \log \frac{w}{n})$. \square

Theorem 5. *The MaxRcvLoad of the DTC algorithm in a ring is $O(n \log(w/n))$.*

Proof. In a round each node v_i receives $O(n)$ number of messages. Since the algorithm has overall $O(\log \frac{w}{n})$ rounds, the MaxRcvLoad per node is $O(n \log(w/n))$. \square

5 Tuning the Algorithm

The algorithm can be tuned by appropriately setting the parameters ψ and τ . In this case we have chosen $\tau = \psi/n$. If we choose $\tau = \psi$ then we may achieve a better message complexity and MaxRcvLoad. Let us assume that $\psi = w/k$ and $\tau = \psi/l$. Here we can claim that,

$$\begin{aligned} \text{No of rounds} = \mathcal{R} &= O\left(\log_{\frac{k}{k-1}}(w/n)\right) \\ &= O\left(\log_{\frac{w}{w-\psi}}(w/n)\right) \end{aligned}$$

It can be observed that the round complexity is primarily dependent on ψ . This is due to the fact that one round completes only when all the nodes together counts a global threshold, ψ , number of triggers. The local threshold, τ , does not affect the number of rounds. The variation of the number of rounds with respect to ψ is shown in Figure 3. We consider three different values of w . It is observed that as ψ increases, the number of rounds required decreases.

The overall message complexity of the algorithm can be given as,

$$\mathcal{M} = O\left(\frac{n\psi}{\tau} \left(\log_{\frac{w}{w-\psi}}(w/n)\right)\right)$$

It is obvious from the above equation that \mathcal{M} is directly proportional to $l = \psi/\tau$. In the proposed algorithm in this paper, $l = n$. If we assume $l = 1$ (i.e. $\psi = \tau$) then we can achieve a better overall message complexity of $O(n \log_{\frac{w}{w-\psi}}(w/n))$. Figure 4 depicts the variation of message complexity with respect to $l = \psi/\tau$. It is clear that as l increases the messages complexity increases. This is due to the fact that l can increase only when ψ is large and τ is small. In this case, each node crosses its local threshold (due

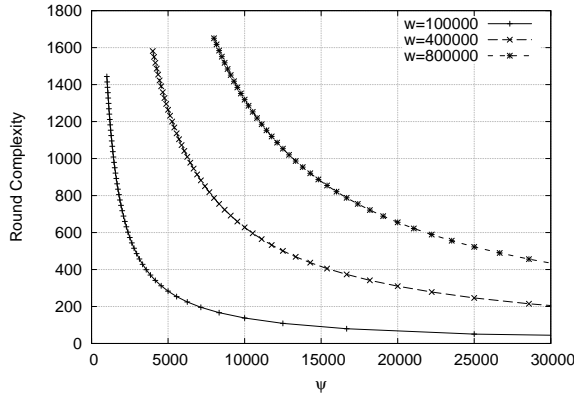


Figure 3: The variation of number of rounds with respect to ψ

to the receive of triggers) many more times and thus more number of *Coin* messages are sent in a round. This increases the overall message complexity of the system.

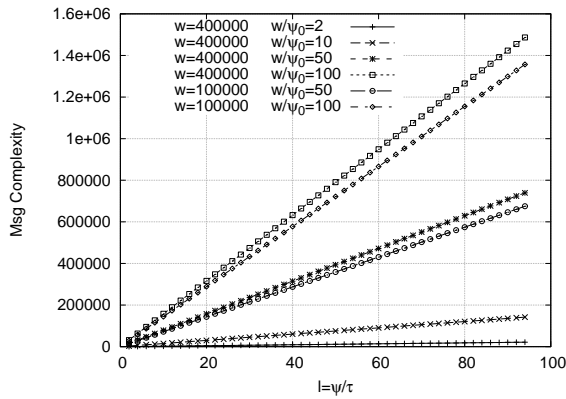


Figure 4: The variation of message complexity with respect to l

Similarly in Figure 5, we see the variation of message complexity with respect to τ . Here we keep w and ψ fixed and observe the variation of message complexity with respect to τ . It is found that as τ increases, message complexity falls sharply initially. However as the value of τ approaches ψ , the rate of fall in message complexity is small and not as sharp as earlier. This is an indication that we may not improve the message complexity by arbitrarily increasing the τ .

6 Message Complexity and Delay Trade-off

There is a trade-off between the message complexity of distributed trigger counting and the time of raising an alarm when at least w triggers have been counted. The simplest way to minimize the number of messages is to let every node detect triggers and when any one has detected w triggers, the global alarm signal is sent. However, in this simple scheme there is obviously a big risk that the system will be seriously delayed in sending the global alarm signal which has to be sent as soon as w or more triggers are received by all the nodes. The obvious way to avoid such delays is to send a message as soon as a trigger is detected.

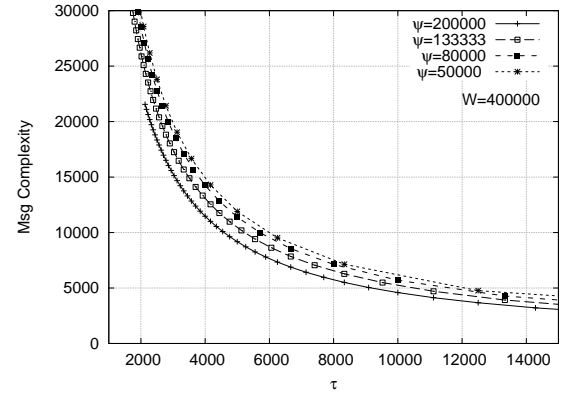


Figure 5: The variation of message complexity with respect to τ

However this will generate a lot of messages (potentially $O(w)$). Hence there should be some compromise between these two extremes.

Let t_0 be the time when w -th trigger enters the system (any node). Let t_1 be the time at which v_n sends the *TargetReached* message to v_1 . We define the delay of generating the alarm as $\mathcal{T} = |t_1 - t_0|$. In the following we give a measure of \mathcal{T} .

Let e_1 be the $(w-1)$ -th trigger and e_2 be the w -th trigger. Now there can be two cases:

Case 1: e_1 and e_2 enter the system in same round.

Every trigger that enters the system is counted by the *master* node through *Coin* message or excess triggers during *end-of-round*. The maximum delay between the times e_1 and e_2 are recorded at the *master* node is the delay between the receive of two successive *Coin* messages by the *master* v_n . Essentially this is equal to the time required by τ triggers to arrive at any arbitrary node (where trigger count is zero). Let this time be denoted by Δ . In this case $\mathcal{T} = \Delta$. Here we assume that processing delay and communication delay is negligible.

Case 2: e_1 and e_2 enter the system in two different successive rounds. In this case, $\mathcal{T} = \Delta + \Delta_e$ where Δ_e is the amount of time required to collect all the excess triggers during the *end-of-round* phase.

If τ is large then delay will be large. However message complexity will be smaller. The reverse will happen if τ is small.

7 DTC in Arbitrary Network

To solve the DTC problem over an arbitrary network, we need to embed a logical unidirectional ring over the graph. For a certain class of graphs (Eulerian graphs), this can be done by forming an Eulerian circuit over the graph. Even though a Hamiltonian circuit provides the perfect ring embedding, the graph may not have an Hamiltonian circuit. Also finding a Hamiltonian circuit is NP-complete. Therefore our approach relies on Eulerian circuit construction. Makki (1999) proposed a distributed algorithm for construction of an Eulerian circuit. The algorithm has a message complexity of $(1+r)(|\mathcal{E}| + n)$ where $0 \leq r < 1$. Here \mathcal{E} denotes the set of edges of the graph. Our approach is to pre-process the input graph, provided it is Eulerian, to find an Eulerian circuit using the

above algorithm. Next on the virtual ring thus obtained, we apply the proposed trigger counting algorithm. Hence the overall message complexity of the DTC problem in an arbitrary network, which is Eulerian, is $O(|\mathcal{E}| + n^2 \log \frac{w}{n}) = O(n^2 \log \frac{w}{n})$. Also MaxRcvLoad for DTC in this case remains $O(n \log \frac{w}{n})$ since average message complexity of Eulerian circuit construction is $O(n)$.

8 Conclusion

We have presented a distributed algorithm for the DTC problem in a ring network with message complexity of $O(n^2 \log(w/n))$ and MaxRcvLoad of $O(n \log(w/n))$. One of the important issues is that the DTC algorithm may generate an alarm after counting w' number of triggers where $w' > w$. One of the future works may be to minimize the difference between w' and w . Proposing an algorithm for the DTC problem in an arbitrary network with optimal complexity can be a challenging work.

References

- Awerbuch, B. (1985), ‘Complexity of network synchronization’, *Journal of ACM* **32**(4), 804–823.
- Chakaravarthy, V. T., Choudhury, A. R., Garg, V. K. & Sabharwal, Y. (2011), An efficient decentralized algorithm for the distributed trigger counting problem, in ‘Proceedings of the 12th International Conference on Distributed Computing and Networking’, Springer-Verlag, Berlin, Bangalore, India, pp. 53–64.
- Chakaravarthy, V. T., Choudhury, A. R. & Sabharwal, Y. (2011), Improved algorithms for the distributed trigger counting problem, in ‘Proceedings of the 25th IEEE International Parallel and Distributed Processing Symposium’, IEEE Computer Society, Washington, DC, USA, Anchorage, Alaska, USA, pp. 515–523.
- Cormode, G., Muthukrishnan, S. & Yi, K. (2011), ‘Algorithms for distributed functional monitoring’, *ACM Trans. Algorithms* **7**(2), 1–20.
- Emek, Y. & Korman, A. (2010), Efficient threshold detection in a distributed environment, in ‘Proceedings of the 29th ACM Symposium on Principles of Distributed Computing (PODC)’, ACM, New York, USA, Zurich, Switzerland, pp. 183–191.
- Emek, Y. & Korman, A. (2011), ‘New bounds for the controller problem’, *Distributed Computing* **24**(3–4), 177–186.
- Garg, R., Garg, V. K. & Sabharwal, Y. (2006), Scalable algorithms for global snapshots in distributed systems, in ‘Proceedings of the 20th Annual International Conference on Supercomputing (ICS)’, ACM, New York, USA, Cairns, Queensland, Australia, pp. 269–277.
- Huang, L., Garofalakis, M., Joseph, A. D. & Taft, N. (2007), Communication-efficient tracking of distributed cumulative triggers, in ‘Proceedings of the 27th International Conference on Distributed Computing Systems’, IEEE Computer Society, Washington, DC, USA, Toronto, Canada, pp. 54–63.
- Keralapura, R., Cormode, G. & Ramamirtham, J. (2006), Communication-efficient distributed monitoring of thresholded counts, in ‘Proceedings of the ACM SIGMOD International Conference on Management of Data’, ACM, New York, USA, Chicago, IL, USA, pp. 289–300.
- Korman, A. & Kutten, S. (2007), Controller and estimator for dynamic networks, in ‘Proceedings of the 26th ACM Symposium on Principles of Distributed Computing (PODC)’, ACM, New York, USA, Portland, Oregon, USA, pp. 175–184.
- Makki, S. A. M. (1999), ‘Eulerian tour construction in a distributed environment’, *Computer Communications* **22**(7), 621 – 628.

Formal Approach for Generating Privacy Preserving User Requirements-Based Business Process Fragments

Mohamed Anis Zemni¹

Amel Mammam²

Nejib Ben Hadj-Alouane³

¹ Ecole Nationale des Sciences de l'Informatique (ENSI), UR/OASIS
Campus Universitaire de la Manouba,
2010 Manouba, Tunisia,
Email: mohamedaniszemni@gmail.com

² Institut Mines-Telecom/Telecom SudParis, CNRS UMR 5157 SAMOVAR
9 Rue Charles Fourier,
91011 Evry, FRANCE,
Email: amel.mammam@telecom-sudparis.eu

³ Ecole Nationale d'Ingenieurs de Tunis (ENIT), UR/OASIS
BP 37, Le Belvedere,
1002 Tunis, Tunisia,
Email: nejib_bha@yahoo.com

Abstract

A business process fragment is a portion of a business process, more commonly designed for reuse purposes. Fragments are intended to be declared as safe from a privacy perspective, when manipulated in an open context. Privacy is related to the authority to have a view on some sensitive information. A business process privacy-preserving fragmentation is the task of decomposing business processes into significant fragments, which can be reused in the future in order to build new business processes while preserving the sensitive information from leakage. This paper presents a design-time two-phases approach to decomposing existing business processes into significant fragments while preserving the integrity of data items that navigate within the process. The first phase is based on the so-called Formal Concept Analysis (FCA) technique handling semantic activity clustering according to designers requirements, while dealing with the privacy constraints. The second phase manipulates clusters of activities and generates ready-for-reuse fragments. Some experiments that demonstrate the feasibility of the proposed approach are also provided.

Keywords: Business Process, Fragmentation, Reuse, Privacy, Semantics, Requirements-Driven.

1 Introduction

In the industry of business process management (Ter Hofstede et al. (2003)), organizations and business entities are more and more focused on improving the quality of their services. At the same time, they experience a need to maintain a high degree of efficiency, with respect to delivery deadlines and productivity; all this, within the context of a continuously increasing competition. A key strategy consists in efficiently implementing and developing modern busi-

ness processes relying on new Web technologies. A business process, as defined by Ouyang et al. (2007), consists of a set of operations, more commonly called activities, organized in a given manner so as to produce a specific service. Implementing new activities and business processes completely from scratch may be a very tedious and time consuming task. Reusing already existing business process *fragments* can reduce the business process development time and enhance its robustness (Schumm et al. (2010)). In fact, the reuse of business process fragments can be an important component of any flexible design strategy resulting in a reduction of process development periods (Markovic & Pereira (2008)). Schumm et al. (2010) cite two ways to design business process fragments: (1) from scratch (Eberle et al. (2009)) or (2) extracted from existing process models. In our work we focus on the second approach which is used to be done essentially manually. The work introduced by Schumm et al. (2011) demonstrates that, apart from improving the quality of the resulting business processes, reusing the fragments allows to avoid (i) the redesign of certain existing fragments, and (ii) the implementation and optimization of all business process artifacts from scratch.

Business process developers, however, need to pay special attention to the data privacy concerns. Indeed, nowadays, the individuals are becoming more and more concerned about the privacy of their personal data that may appear within the process boundaries. As defined by Sweeney (2002), privacy is related to the authority to receive some sensitive information, e.g., personal information. Such important information is called sensitive and should not be disclosed. Though sensitive information may occur in a business process for the purpose of performing its key functionalities, they are not intended for sharing or publishing. Consequently, the fragments that are to be reused, must be individually and conjointly safe. One has to make sure that (i) each fragment produced from a business process decomposition approach is safe, and (ii) two fragments from which sensitive information can be inferred should not be coupled together when a new business process is built.

In a previous work (Zemni et al. (2012a,b)), we have presented an initial and informal approach for business process decomposition while maintaining the

Copyright ©2014, Australian Computer Society, Inc. This paper appeared at the Thirty-Seventh Australasian Computer Science Conference (ACSC2014), Auckland, New Zealand, January 2014. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 147, Bruce H. Thomas and David Parry, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.

privacy of sensitive information. The approach generates *any* fragment whose activities involve common functionalities. Fragments may contain superfluous activities which are not interesting for the designer. Moreover, the approach relies mainly on grouping semantically close activities while structural concerns still early stage. In this paper, we seek to improve on the fragmentation mechanism by attempting to provide requirements-driven algorithms, as well as formal definitions that make the approach formal. We also prove the correctness of the privacy-preserving mechanism. Here, a requirements-driven approach is presented in order to provide useful and reusable business process fragments. Indeed, approach relies on the fragmentation of existing business processes to retrieve fragments that may suit designers' requirements.

The structure of the paper is articulated as follows: in the next section, we give a motivating example with a detailed illustration of *reuse issues* that may occur during the fragmentation process (Section 2). Section 3 provides formal definitions for business processes and business process fragments along with *privacy* and *semantics* concerns that allow us to formalize and prove the decomposition approach. Section 4.1 introduces our requirements-driven clustering mechanism to generate *clusters* of *semantically* close activities, and deals with *privacy*. A detailed *proof* is also given in section 4.2 to show that the privacy is well respected. We then present an algorithm to generate *privacy-aware* and *reusable* fragments from the composition of the previous clusters 4.3. Section 5 evaluates the effectiveness of the proposed approach and gives the description of the most significant related statistics. We finally conclude with the related work and the results summary in sections 6 and 7, respectively.

2 Working Example

In this part, we specify in details the main issues related to the fragmentation through a practical case study. For this, we consider an inter-department collaboration scenario where a business process designer, from a given department, desires to provide relevant fragments of his own process to third designers, from another department, who needs to build a new process. Figure 1 depicts a "surgery performing" process case study.

Represented in Business Process Model and Notation (BPMN)¹, this process is roughly defined as a set of activities, represented with rounded boxes, a set of data items, represented with note boxes, a set of events, represented with circles, and, a set of gateways, represented with diamonds. Activities, gateways, and events are linked to each other with control flows, i.e., represented with solid arrows, and data items are linked to activities by means of data flows, i.e., represented with dashed arrows. Control flows depict the execution order of the elements they link, while dashed arrows represent the data items routed in between activities, i.e., as either inputs or outputs.

The process represented in figure 1 performs as follows: a surgery order triggers the process execution. Indeed, a message event, i.e., which is a start event, receives a '*surgery order number*'. Activity (*a*₁) uses the '*surgery order number*' to retrieve the *surgery information* as well as the *patient's SSN* (Social Security Number). Note that the *surgery information* contains the type of surgery, whether it is urgent

or not, etc. Activity (*a*₂) receives the *patient's SSN* and retrieves the *patient's information*. After that, two branches are called concurrently: (i) selecting a free surgeon (activity (*a*₅)) and selecting a free surgery room (activity (*a*₆)), and (ii) asking for patient's history (activity (*a*₃)) and compiling patient's record (activity (*a*₄)). Activities (activity (*a*₅)) and (activity (*a*₆)) use the *surgery information* and check the surgeon and surgery room availabilities, while activities (activity (*a*₃)) and (activity (*a*₄)) check for surgery and history inconsistencies, e.g., whether any contraindication exists between the *surgery information* and the *patient's record*, or if the patient take some drugs he must stop before the surgery. The process terminates immediately if any inconsistency is detected. Once both branches finish their execution, activity (*a*₇) confirms the patient for surgery. When the receive message event receives the *surgery report*, the patient's record (activity (*a*₈)) is updated (i.e., using the *surgery information*, the *patient's SSN*, and the *surgery report*), the *order patient follow up* activity (activity (*a*₉)) is executed, and the process is ended.

Generally speaking, this process can be published in an open context to be reused as part of a new business process to perform more complex functionalities. For instance, the process may be part of a "Surgery Performing and Reimbursement" process to perform the costs reimbursement tasks by the social security in addition to the functionalities involved by the surgery performing process.

This process, however, is not necessarily *safe* while it may reveal some sensitive information compiled from data associations. For instance, the association between the data item *patient's information* (activity *a*₂ output) and the data item *patient's record* (activity *a*₃ output), is sensitive. Indeed, this association may lead to *patient's illnesses* disclosure. In this paper, it is dealt with information, more specifically data items, that are routed between activities. Recall that information enclosed within the activities cannot be viewed, as activities are seen as black boxes and are then safe. More specifically, it is dealt with non-sensitive data items that are safe when they are considered separately, but turn out to be critical when they are associated together. Note that activities, also, are safe when they are taken separately, where for a given activity, inputs and outputs do not form sensitive associations. Indeed, it is the *association* between data items which may be *sensitive*. Consequently, when the process is coupled with malicious activities, the latter may probably make use of the data items and infer sensitive information. For example, some additional activities (or even a single one) may be added to the surgery performing process, using the *patient's information* data item and the *patient's record* data item manipulates them and generating the *patient's illnesses*.

Moreover, the process may be uninteresting for reuse, as a whole. For example, let the designer be interested only in preparing surgery and manipulating patient's record (i.e., history) within the desired "Surgery Performing and Reimbursement" process. The designer already has his own patient's follow up activities, and thus do not need them from the current process. Consequently, it would be wiser to catch only portions that semantically match the designer's requirements. Therefore, activities *a*₉, should not be kept for reuse. This solution, however, is intuitive, essentially manual, and needs business investigations, as for many existing decomposition approaches (as stated by Khalaf (2008)). This task also requires a

¹<http://www.omg.org/spec/BPMN/2.0/>

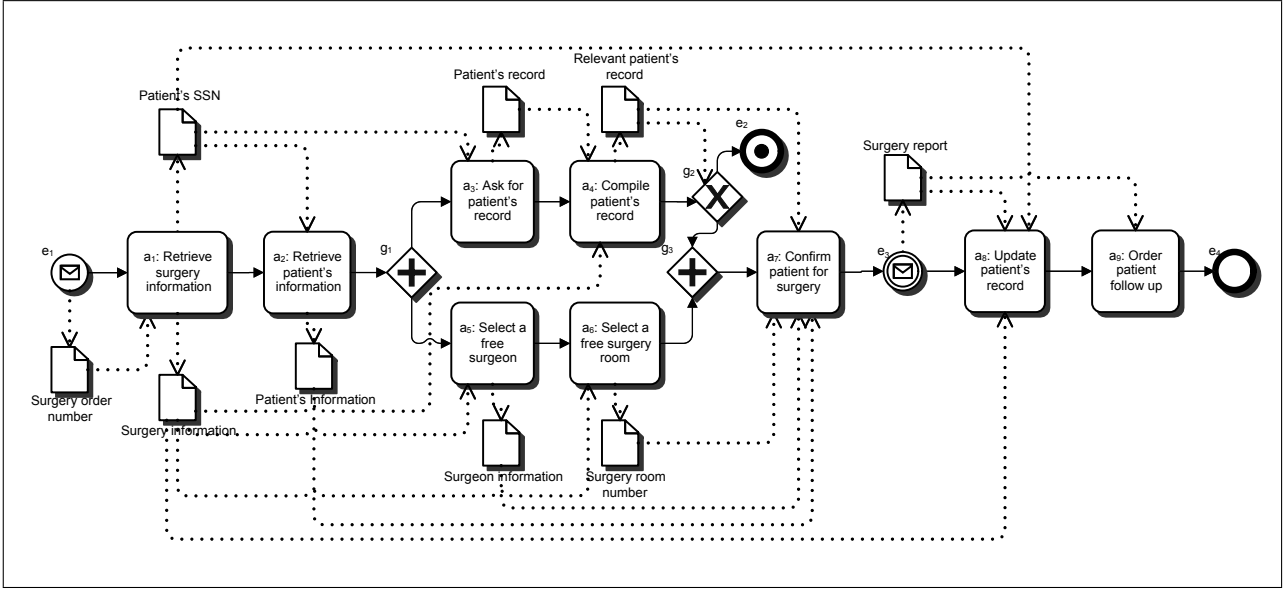


Figure 1: An Example of Surgery Performing Process.

good understanding of the initial process to enable retrieving the interesting portion for the designer. This turns out to be a difficult task and thus needs automation especially to deal with big and complex processes.

3 Fragmentation Preliminaries

In this section, we present the necessary formal notions needed to understand and perform the fragmentation task, in order to generate well-formed and privacy-aware fragments whose functionalities are consistent with those required by the designer.

3.1 Business Process and Business Process Fragment Models

As stated by Mancioppi et al. (2011), a complete process model definition is an important artifact in order to enable performing the fragmentation task.

Ouyang et al. (2007) have introduced a business process (process for short) as a directed graph that is defined as a collection of activities, events, and gateways, linked with control flows. For our purposes, this definition is further refined, to clearly specify the various parts and aspects of a process, namely data elements, i.e., consisting of data items and data flows. We formally define a business process as follows:

Definition 1 (Business Process) A Business Process is a tuple $P=(O, A, G, C_f, D, D_f)$, where $O \subseteq (A \cup G)$ is a set of objects composed of a set of activities and events A , and a set of gateways, G . $C_f \subseteq (O \times O)$ is the control flow relation to link objects to each other, D is the set of data items handled by the activities, and $D_f \subseteq (D_{fin} \cup D_{fout})$ is the data flow relation to link objects to their corresponding input data items $D_{fin} \subseteq (O \times D)$, and activities to their output data items $D_{fout} \subseteq (A \times D)$.

The relation D_{fout} does not involve gateways as they generate no data items. Note that events and activities are set in the same activity set A as they act alike. In the rest, we use “activities” instead of “activities and events”.

We define a path as a linear and connected portion of a business process describing a set of sequential

activities, i.e., whose execution is sequential, starting from a given activity, traversing a set of sequential activities, and leading to another activity, and where each couple of consecutive activities are linked by means of a control flow. Let $Paths$ be the set of all possible paths defined in a process such that $Paths = \{(act_1, \dots, act_n) | \bigwedge_{i=1..n} act_i \in A \wedge \forall i. (1 \leq i \leq (n-1) \Rightarrow (act_i, act_{i+1}) \in C_f)\}$. A process should contain no unconditional control flow cycles leading the process to run indefinitely. Consequently, there should not exist any path ending with an activity it has started by.

Let the following formally defined property that each business process has to respect:

Property 1 (Unconditional Cycle Free)
 $\forall (a_1, \dots, a_n). ((a_1, \dots, a_n) \in Paths \Rightarrow a_1 \neq a_n)$.

A business process fragment (fragment for short) as defined by Schumm et al. (2010) is a *connected* sub-process designed for reuse purposes. It is composed of at least one activity, and of several edges, representing control and data flows. A fragment involves some functionalities and is intended to be composed with other ones to build new processes. A fragment can depict dangling control flows, i.e., with either no source or sink activities specified. More formally:

Definition 2 (Business Process Fragment) A Business Process Fragment is a tuple $f = (O, A, G, C_f, D, D_f, fn)$, with O, A, G, D , and D_f as in Definition 1. $C_f \subseteq (O \cup \{\perp\})^2 - \{(\perp, \perp)\}$ is the complete and/or dangling control flow relation to link objects to each other, and fn is a set of terms that describe the fragment’s involved functionalities.

Note that character \perp is used to denote dangling control flows, i.e., representing missing start or end of a control flow.

As for the case of a complete process, a fragment should contain no unconditional cycles. This property is necessary to ensure that business processes that are made of fragments also respect property 1.

Let the following connectivity property that should be ensured when generating fragments:

Property 2 (Connectivity) A fragment f is connected iff $\forall (a_1, a_2). (a_1, a_2 \in f.A \Rightarrow (a_1, \dots, a_2) \in Paths \vee (a_2, \dots, a_1) \in Paths \vee \exists a_3. ((a_3, \dots, a_1) \in Paths \wedge (a_3, \dots, a_2) \in Paths))$.

That is, a fragment is connected if and only if each couple of activities belong to a path when they are placed on the same branch or belong to two different paths with a common activity when they are placed on different branches. The connectivity condition is necessary to ensure that when a fragment is used in a new process, all activities can be executed.

Generally speaking, a fragment denotes a complete process when it encloses no dangling control flows, i.e., $C_f \subseteq O^2$.

3.2 Privacy-Preserving Mechanism

As we have already tackled in the motivating example, when publishing a fragment, the association between some non-sensitive data items, involved in this fragment, can disclose some sensitive information. For example, “*patient’s illnesses*” can be inferred from the association between the data items “*patient’s information*” and “*patient’s record*”. The association between such couples of data items are called *sensitive* and should never be published in an open context. Therefore, data items that are involved in a sensitive association are called *in conflict*. To get over such issues, we propose the following privacy constraint definition that should be taken into account when performing the fragmentation task.

Definition 3 (Privacy Constraints) Privacy constraints, denoted $C_N \subseteq D^2$, is a set of data item couples that should not figure in the same fragment.

Consequently to the definition, activities that output conflicting data items should never figure in the same fragment; they are called *in conflict* with each other. To assert that two activities are conflicting, only the data they produce are taken into account. Indeed, when an activity is selected for a fragment, its input data, i.e., that are in conflict with other ones in the fragment, are not yet received and then may be substituted with other ones.

For instance, $C_N = \{(surgery\ information, patient's\ information), (patient's\ information, patient's\ record), (patient's\ information, relevant\ patient's\ record)\}$, in Figure 1, depicts the privacy constraint set. Then, activities a_1 and a_2 , activities a_2 and a_3 , and activities a_2 and a_4 are pairwise in conflict.

The proposed privacy constraints leads to the following privacy-awareness property that we have to ensure on each fragment f :

Property 3 (Privacy-awareness) A fragment f is safe iff $\forall (a_1, a_2). (a_1 \in f.A \wedge a_2 \in f.A \Rightarrow ((f.D_{f_{out}}[\{a_1\}]^2 \times f.D_{f_{out}}[\{a_2\}]) \cup (f.D_{f_{out}}[\{a_2\}] \times f.D_{f_{out}}[\{a_1\}])) \cap C_N = \emptyset$.

That is, each fragment should not contain any couple of activities that output conflicting data items.

3.3 Activities to Functional Requirements Similarity

In our work, we are interested in any fragment verifying Definition 2 and whose activities verify some

²Given the relation $R \subseteq X \times Y$, then, $R[X_1] = \{y | y \in Y \wedge \exists x. (x \in X_1 \wedge (x, y) \in R)\}$, and $R^{-1}[Y_1] = \{x | x \in X \wedge \exists y. (y \in Y_1 \wedge (x, y) \in R)\}$.

given functional requirements, denoted Q , in addition to privacy concerns. To this aim, we focus on any activity textual description that gives the main functionalities involved by the activities. If no activity description is provided, then, activities’ labels and data items’ labels are retrieved. Generally speaking, activity descriptions are made of a set of terms that describe the functionalities involved by the activity.

Let \hat{a} be the textual description of activity a , and \hat{Q} be the textual description of requirements Q . In our work, activity descriptions, \hat{a} , as well as the functional requirements, \hat{Q} , are pretreated (e.g., by removing stopwords (Fox (1992)), by stemming terms (Porter (1997)), and by unifying synonyms), and weighted (e.g., using term weighting mechanisms (e.g., TF/IDF³ (Salton & Buckley (1988)))). This respectively generates weighted description vectors \vec{a} and \vec{Q} . Let $w(t, \vec{a})$ be a function that returns the weight of term t in the activity description \hat{a} . Note that function, w , returns 0 if term, t , does not belong to the activity description, \hat{a} . For example, $\hat{a}_5 = \{ 'select', 'free', 'surgeri', 'surgeri', 'inform', 'surgeri', 'inform' \}$ is an activity description, and $\vec{a}_5 = ((0.105, select), (0.105, free), (0.037, surgeri), (0.056, inform))$ its corresponding vector, where $w(surgeri, \vec{a}_5) = 0.037$.

To include or not an activity a in a fragment, we have to compute the similarity between the activity vector \vec{a} and the requirements vector \vec{Q} . To this aim, we use Vector Space Model (VSM), introduced by Salton et al. (1975). Let α be a similarity threshold fixed by the designer, above which, the activity would be part of the fragment. Guidelines for fixing an appropriate threshold is out of the scope of this paper.

Generally speaking, an activity may correspond to all the query terms or *only part of them*. The latter occur when the query terms involve complex functionalities which cannot all be met by a single activity. Consequently, it would be wiser considering only terms of the activity descriptions for the similarity computation instead of the union of both query and activity description terms, as set in the literature. This enables to focus on the functionalities in Q that are involved in a given activity. The similarity function is formally defined as follows.

Definition 4 (Similarity Function) Given a functional requirement vector \vec{Q} and an activity vector \vec{a} , the similarity between \vec{Q} and \vec{a} is defined by:

$$sim(\vec{Q}, \vec{a}) = \frac{\sum_{j \in dist(\hat{a})} w(j, \vec{Q}) \times w(j, \vec{a})}{\sqrt{\sum_{j \in dist(\hat{a})} w(j, \vec{Q})^2} \times \sqrt{\sum_{j \in dist(\hat{a})} w(j, \vec{a})^2}}$$

with $dist(\hat{a})$ is a function which returns distinct terms in \hat{a} .

For instance, $sim(\vec{Q}, \vec{a}_5) =$

$$\frac{0 * 0.105 + 0 * 0.105 + 2 * 0.037 + 0 * 0.056}{\sqrt{2^2} * \sqrt{0.105^2 + 0.105^2 + 0.037^2 + 0.056^2}} = 0.36$$

where $\vec{Q} = ((2, patient), (2, surgeri), (2, record), (2, reimburs))$ is the requirements vector. Note that the requirement terms are all weighted 2 meaning that they are as important as each other.

³TF/IDF is a term weighting method which reflects the importance of a term for a document among a corpus of documents. Higher weights are assigned to terms occurring frequently in a particular document, but rarely on the remainder of the document collection. In our work, documents represent activities.

4 Privacy-Aware Business Process Fragmentation

In this section, we propose a two-phases requirement-driven fragmentation. The first phase involves an algorithm in which we incorporate privacy constraints and semantics to generate semantically close and privacy-aware clusters of activities. A cluster of activities is a group of coupled activities that cooperate together closely to achieve the same goal. We also provide a proof demonstrating the correctness of the approach w.r.t. Property 3. The second phase involves an algorithm that derives, from clusters, reusable fragments. Indeed, clusters of activities are garnished with structural concerns (gateways, flows, etc...) so as to draw connected and reusable fragments.

4.1 Clustering Algorithm

The process clustering consists in selecting activities that are semantically close to the functional requirements and classifying them w.r.t. the functionalities they are involving, and this, while maintaining the sensitive information privacy. The clustering is based on the so-called Formal Concept Analysis (FCA) (Ganter & Wille (1999)). The latter is a data analysis technique, used for classifying similar objects within object collections, w.r.t. their common attributes. Our work adapts the FCA to the process activity clustering. That is, activities are mapped onto objects, and activity descriptions are mapped onto attributes. We extend this technique so as to compute the similarity between the activities and the required functionalities. This enables forming clusters consisting of activities that cooperate to involve the same functionalities. We also constrain the technique with the privacy constraints.

The following are the extended FCA element definitions that the clustering algorithm, Algorithm 1, relies on.

Definition 5 (Formal Context) A formal context is a triplet, $C = (A, T, w)$ involving a set of process activities, A , a set of terms, T , that has been retrieved from the process activity descriptions, and a weight function, $w : T \times A \rightarrow \mathbb{R}$, that returns the weight of a term, $t \in T$, for an activity $a \in A$.

Definition 6 (Clustering System) $S = (C, D_{f_{out}}, C_N)$ is a clustering system where C is the formal context, $D_{f_{out}}$ is the output data flow relation, and C_N are the privacy constraints.

Definition 7 (Galois Correspondence) A Galois correspondence involves two functions, Θ and Δ , for a clustering system $S = (C, D_{f_{out}}, C_N)$ and functional requirements Q .

$\Theta : \mathcal{P}(A) \rightarrow \mathcal{P}(T \cap Q)$ is defined over the power set $\mathcal{P}(A)$, and returns the maximal set of terms, among Q , that are shared by all the activities, where Q is the set of requirements' terms. That is, for a given $A_i \in \mathcal{P}(A)$, where $\forall (a_1, a_2). (a_1 \in A_i \wedge a_2 \in A_i \Rightarrow ((D_{f_{out}}[\{a_1\}] \times D_{f_{out}}[\{a_2\}]) \cup (D_{f_{out}}[\{a_2\}] \times D_{f_{out}}[\{a_1\}])) \cap C_N = \emptyset$), then, $\Theta(A_i) = \{t \in (T \cap Q) | w(t, \vec{a}) \neq 0, \forall a \in A_i\}$.

$\Delta : \mathcal{P}(T \cap Q) \rightarrow \mathcal{P}(A)$ is defined over the power set $\mathcal{P}(T \cap Q)$, and returns relevant-enough activities (that are maximal), according to a fixed threshold α , that share all the terms in $(T \cap Q)$. That is, for a given $T_j \in \mathcal{P}(T \cap Q)$, $\Delta(T_j) = \{a \in A | \text{sim}(\vec{Q}, \vec{a}) \geq \alpha\}$.

Definition 8 (Formal Concept/Cluster) Given a clustering system $S = (C, D_{f_{out}}, C_N)$, a formal concept is a tuple $Con = (T_j, A_i)$, where $\Theta(A_i) = T_j$, and, $\Delta(T_j) = A_i$. A_i is called a cluster made of relevant-enough activities collaborating to process the functionalities represented with T_j .

Algorithm 1 depicts the requirement-driven privacy-aware clustering process. It takes as input parameters a clustering system S (Definition 6), and the functional requirements Q . The algorithm returns the set of formal concepts $conSet$ from which clusters are derived.

Algorithm 1 Requirement-driven Privacy-aware clustering

```

1: function CLUSTERING(FormalContext  $C$ ,
   Output  $D_{f_{out}}$ , Privacy  $C_N$ , Requirements
    $Q$ ):FormalConceptSet
2:   declare FormalConceptSet  $conSet \leftarrow new$ 
   FormalConceptSet()
3:   begin
4:   for all  $A_i \in \mathcal{P}(C.A)$  do
5:     if  $notConflicting(A_i, C_N, D_{f_{out}})$  then
6:       TermSet  $T_i \leftarrow \Theta(A_i)$ 
7:       if  $\Delta(T_i) = A_i$  then
8:          $conSet \leftarrow conSet \cup \{new \text{ Formal-}$ 
           Concept  $(T_i, A_i)\}$ 
9:       end if
10:    end if
11:  end for
12:  return  $conSet$ 
13: end
14: end function

```

The formal context, depicted in definition 5, represents the clustering basis. It involves the activities to classify, the classification criteria, i.e., the terms, and the weights of terms for a given activity. Note that we have extended the relation between terms and activities from binary relations to values ones, i.e., given by the function w . The clustering system involves the process corresponding formal context, the privacy constraints it is subject to, as well as output data flows to decouple conflicting activities during the clustering task.

Given that the clustering task is activity-centric, the algorithm applies on every element of the power set $\mathcal{P}(C.A)$ ⁴. The algorithm uses the Galois correspondence functions (Definition 7), Θ and Δ (lines 6,7), that we have extended with semantic concerns. Function Θ returns a set of terms that represent the functionalities that may probably be involved by the activities (e.g., $\Theta(\{e_1, a_1, a_5, a_6\}) = \{surgeri\}$). The Θ parameter is privacy-aware as the privacy is checked when selecting the subset A_i by means of function $notConflicting$ (line 5) which returns *true* if the activities in A_i are not conflicting given C_N . Function Δ applies over $(T \cap Q)$. This permits to focus the similarity computation on the terms in Q that may be involved by the process activities. Δ returns the activities that are close to the functional requirements Q , w.r.t. threshold α . For instance, $\Delta(\{surgeri\}) = \{e_1, a_1, a_5, a_6\}$, where activities a_4, a_7, e_2, a_9, a_8 are not returned, i.e., even if they involve the term *surgeri* in their description, as they are not relevant enough according to threshold $\alpha = 0.35$ (e.g., $\text{sim}(\vec{Q}, \vec{a_9}) = 0.114 < \alpha$). Note

⁴ $C.A$ is the activity set, A , for a formal context, C . The same thing applies to similar notations.

that Δ , as it exists in the literature, would return $\{e_1, a_1, a_5, a_6, a_4, a_7, e_2, a_9, a_8\}$ for $\Delta(\{surgeri\})$.

Using functions Θ and Δ , we generate the formal concepts (Definition 8). A formal concept in our work represents a cluster of close activities, that share the same functionalities. Indeed, for a given formal concept, *all* activities involve the functionalities described by the terms in Q or part of them, and the terms are significant enough for *only* those activities according to the fixed threshold α . For instance, $(\{patient, sergeri\}, \{a_1, a_7, a_8\})$ is a formal concept, but $(\{patient, sergeri\}, \{a_1, a_4, a_7, a_8, a_9\})$ ⁵ is not a formal concept as $\Delta(\{patient, sergeri\}) = \{a_1, a_7, a_8\}$ does not contain a_4 nor a_9 .

Note that, *without* considering the privacy constraints, the clustering task would return the cluster $\{a_2, a_3, a_4, a_7, a_8\}$, i.e., involving patient's manipulation, while a_2 should not be put in the same cluster as a_3 and a_4 , as explained in section 3.2.

In the following subsection, we demonstrate that Algorithm 1 verifies property 3.

4.2 Correctness of the Clustering Algorithm

In order to validate the proposed approach, we have verified the correctness of Algorithm 1 using the B formal method and its refinement concept. Before describing how we used this method to prove the correctness of the algorithm, we give a brief introduction of the B method.

Introduced by Abrial (1996), B is a formal method for safe project development. B specifications are organized into abstract machines that encapsulate state variables on which operations are expressed. The set of the possible states of the system are described using an invariant which is a predicate in a simplified version of the ZF-set theory, enriched with many relational operators. Refinement is the process of transforming a specification into a less abstract one. In B, we distinguish behavioral and data refinement. The behavioral refinement, used in this paper, includes weakening of preconditions, the replacement of parallel substitution with a sequence one, etc. To ensure the correctness of a B specification, a set of proof obligations is generated for each B component. At the abstract level, these proofs aim at verifying that the invariant of the system is satisfied after the execution of each operation. Refinement proofs permit us to check the correctness of each concrete operation with respect to its abstraction. We assume that readers are familiar with B method and more details can be found in Abrial (1996).

To establish the correctness of Algorithm 1, we have adopted the B architecture depicted in Figure 2:

- At the abstract level, we build a B machine *FragProcess* that defines a set of types which correspond to *Activities*, *Objects* (*Data items*) and *Terms*. The process is described through 3 variables defined as follows:

1. *output*: this function gives the data items used by an activity as output.
2. *desc*: this function gives the set of terms corresponding to each activity (it corresponds to \hat{a}).
3. *conflict*: this relation stores the sets of the data items couples that are in conflict (privacy constraints C_N).

⁵This formal concept is generated using the FCA technique as it exists in the literature

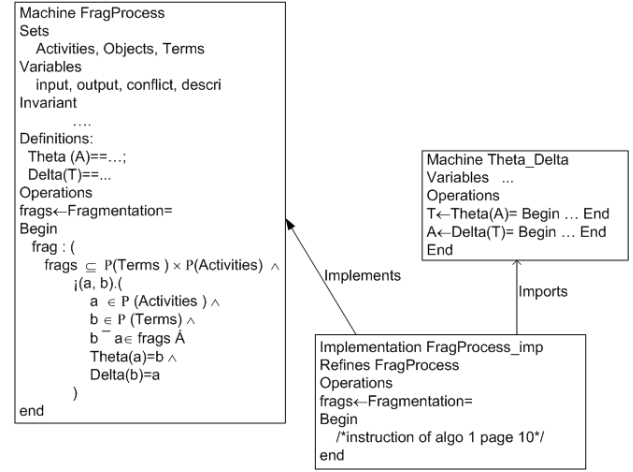


Figure 2: B Architecture for Algorithm 1 and its Correctness.

Finally, a B operation is specified in order to perform the fragmentation of the process. This operation is expressed in an abstract manner by giving the properties that the returned fragments should verify. Functions, Θ and Δ are declared as definitions

- According to the B refinement technique, proving that algorithm 1 is correct comes down to establish that it is a possible implementation of the previous B machine. To do that, we create a B implementation component of the previous machine in which operation *Fragmentation*, verifying the privacy awareness and maximality properties, is implemented by the instructions of algorithm 1. This implementation component imports a B machine in which function Θ and Δ are also described as operations in order to be called from the implementation component.

To validate these B components, we have generated and proved a set of proof obligations. Not surprising, components *FragProcess* and *Theta.Delta* do not generate any proof obligation because their operations do not modify their variable. Consequently, the invariant remains true. Component *FragProcess.Imp* generates 61 proof obligations : 53 have been discharged automatically where the others require the human intervention in order to help the prover find the right deduction rules to establish them. Recall that the proof obligations of this component aims at proving that the implementation of operation *Fragmentation* is correct with respect to its abstract specification. In this way, we prove the correctness of algorithm 1.

4.3 Fragments' Structure Building Algorithm

Although we ensured the relevance and privacy-awareness of the derived clusters, the clustering phase lefts the structural concerns away. Indeed, some clusters depict disconnected structures, when reconstructed into fragments. This is not conform to property 2. For instance, the cluster $\{e_1, a_1, a_5, a_6\}$ is disconnected and would not lead to a correct fragment. Moreover, the result may further be improved by attempting to merge fragments, i.e., that do not reveal sensitive information, into coarser-grained ones. This permits to involve more complex functionalities. For instance, the cluster $\{e_1, a_1, a_5, a_6\}$, the cluster

Algorithm 2 Fragment Building

```

1: function BUILDFRAG(FormalConceptSet
   conSet, Process  $P$ , Privacy  $C_N$ ):FragmentSet
2:   declare FragmentSet  $F \leftarrow$  new FragmentSet()
3:   begin
4:     for all  $(superT, superCl) \in \{(T, A) | \exists ((T_1, A_1), (T_2, A_2)). \{(T_1, A_1), (T_2, A_2)\} \subseteq conSet \wedge (\forall (a_1, a_2). (a_1 \in A_1 \wedge a_2 \in A_2 \Rightarrow notConflicting(\{a_1, a_2\}, C_N))) \wedge (A_1 \cup A_2) \subseteq A \wedge (T_1 \cup T_2) \subseteq T)\}$  do
5:       Fragment  $f \leftarrow$  new Fragment()
6:       //Insert activities and data elements
7:        $f.A \leftarrow superCl$ 
8:        $f.D \leftarrow P.D_{fin}[superCl] \cup P.D_{fout}[superCl]$ 
9:        $f.D_{fin} \leftarrow superCl \triangleleft P.D_{fin}$ 
10:       $f.D_{fout} \leftarrow superCl \triangleleft P.D_{fout}$ 
11:      //Insert gateways and the corresponding input data elements
12:       $f.G \leftarrow ran((superCl \triangleleft C_f) \triangleright G) \cup dom((G \triangleleft C_f) \triangleright superCl)^6$ 
13:       $f.D \leftarrow f.D \cup P.D_{fin}[f.G]$ 
14:       $f.D_{fin} \leftarrow f.D_{fin} \cup f.G \triangleleft P.D_{fin}$ 
15:       $f.O \leftarrow f.A \cup f.G$ 
16:       $f.D_f \leftarrow f.D_{fin} \cup f.D_{fout}$ 
17:       $f.C_f \leftarrow f.O \triangleleft (P.C_f \triangleright f.O)$  //Insert complete control flows
18:       $f.C_f \leftarrow (f.C_f^{-1}[\emptyset] * \perp) \cup (\perp * f.C_f[\emptyset])$  //Insert dangling control flows
19:       $f.fn \leftarrow superT$  //Insert fragment's functionalities
20:       $F \leftarrow F \cup splitIntoConnected(f)$ 
21:   end for
22:   return  $F$ 
23: end function
    
```

$\{a_1, a_7, a_8\}$, and the cluster $\{a_3, a_4, a_7, a_8\}$, that were generated from the clustering phase, can be melted together in a single cluster to involve both surgery preparation and patient's record manipulation.

Algorithm 2, *buildFrag*, takes as parameters the input process, P , the formal concepts, $conSet$, that are generated during phase 1, and the privacy-constraints C_N . The algorithm returns a set of connected and privacy-aware fragments.

The algorithm applies on each super-formal concept that is made of the union of formal concepts whose cluster activities are not conflicting (line 4). For example, given two formal concepts (T_1, A_1) and (T_2, A_2) , where activities from A_1 are not conflicting with activities from A_2 , their super-formal concept is $(T_1 \cup T_2, A_1 \cup A_2)$. The merge aims at assembling clusters involving different functionalities in order to provide coarser-grained fragments making easier their integration into new processes. A super-cluster corresponds to a connected fragment when all its activities can be connected. It corresponds to multiple connected fragments otherwise. For instance, $\{e_1, a_1, a_3, a_4, a_5, a_6, a_7, a_8\}$ is the super-cluster made from clusters $\{e_1, a_1, a_5, a_6\}$, $\{a_1, a_7, a_8\}$, and $\{a_3, a_4, a_7, a_8\}$.

Given a super-cluster, the algorithm builds a new fragment consisting of the super-cluster activities. The latter are garnished with the corresponding data elements, i.e., namely data items (either inputs or outputs), and data flows (lines 5-9).

⁶Given the relation $R \subseteq X \times X'$, $dom(R) = \{x | x \in X \wedge \exists x'. (x' \in$

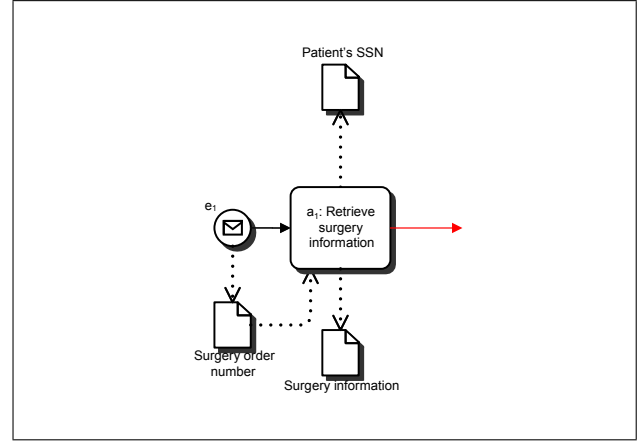


Figure 3: Example 1 of a Connected Fragment Generated from the Super-Cluster $\{e_1, a_1, a_3, a_4, a_5, a_6, a_7, a_8\}$.

Next, the algorithm inserts the gateways that are linked to the selected activities, i.e., in the initial process. Gateways' data elements are also picked and reported in the new fragment (lines 10-12). After that, the algorithm draws the control flows. Indeed, complete control flows are imported from the original process P , and this, when both source objects and sink objects belong to the fragment's objects, $f.O$ (line 15). Control flows, that were broken during the clustering phase, are replaced with dangling ones (line 16). Dangling control flows represent gluing points regarding the new process.

The algorithm, then, tags each fragment with the terms, $superT$, describing the fragment involved functionalities (line 17).

Finally, the algorithm uses function *splitIntoConnected*, according to property 2, in order to split the fragment f into multiple connected fragments when activities cannot be connected to each other. It keeps the fragment's structure as it is otherwise. Function *splitIntoConnected* is not detailed within this paper. Indeed, it is straightforward to check whether a cluster is connected or not respectively to property 2, e.g., using existing tree navigation algorithms (Nuutila & Soisalon-Soininen (1994)).

Note that the generated fragments also fulfill the property 1. Indeed, given that processes are assumed respecting such property and fragments are portions of those processes then fragments respect property 1, too.

Figure 3 and figure 4 illustrate generated fragment examples from the super-cluster $\{e_1, a_1, a_3, a_4, a_5, a_6, a_7, a_8\}$. According to algorithm 2, the fragments deal with both surgery preparation and patient's record manipulation. Note that the fragment in figure 3 involves, in fact, only surgery preparation. Assigning the specific functionalities to the fragments will be handled in future works. The privacy-preserving is well preserved, w.r.t property 3 as clusters whose output activities may form sensitive associations are dissociated. The activities of both fragment examples are semantically relevant enough for parts of the functional requirements Q . Note that the *reimburs* term, i.e., part of the requirements Q , are left away as there are no activities involving such functionality. Moreover, the fragment structure is correct respectively to definition 2 and dangling

$X' \wedge x \mapsto x' \in R\}$ and $ran(R) = \{x' | x' \in X' \wedge \exists x. (x \in X \wedge x \mapsto x' \in R)\}$

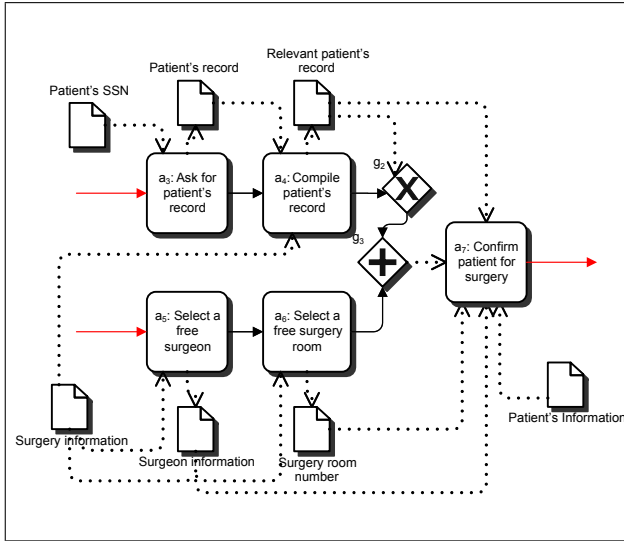


Figure 4: Example 2 of a Connected Fragment Generated from the Super-Cluster $\{e_1, a_1, a_3, a_4, a_5, a_6, a_7, a_8\}$.

control flows play gluing points in the new process. Therefore, the generated fragments can be reused in building new business processes.

5 Implementation and Results

We have implemented a tool that corresponds to the proposed algorithms. Our approach plays on existing FCA mechanisms adaptability. Indeed, a Colibri-java library⁷, for classical Formal Concept Analysis, has been extended with a prototypical coding of the semantics and privacy extensions. The tool receives as input an XML document containing the activity descriptions of each selected process, an XML document containing the process organization (as presented in definition 1), and the set of privacy constraints. It produces a set of privacy-aware fragments as presented in definition 2. We have implemented activity description preparation classes based on Fox (1992) and Porter (1997), respectively for removing stopwords and stemming terms, as well as a weighting class implementing TF/IDF Salton & Buckley (1988) weighting mechanism.

To evaluate our decomposition algorithm, we have directed the tests to check (i) the granularity of the resulted fragments and (ii) the scalability of the algorithm. To evaluate the granularity of the resulted fragments, we ran the algorithm over a set of 5 processes from SAP library (Industry Specific Business) while varying the value of the similarity threshold α and counting the average number of activities within the resulted fragments. We took a set of 3 terms, having the same weight, for the functional requirements \vec{Q} . The decomposition of each process leads to 2 fragments, on average. The statistical results are depicted in table 1.

Note that more the similarity threshold increases more the granularity decreases. This means that some activities were removed as their similarity were below α . The similarity may be further improved when activities have rich description. The granularity is not high compared to the size of the processes which reflects the weak activities interdependency. This may

	size	0	0.1	0.2	0.3	0.4	0.5
P_1	10	3.5	2.5	1.5	1	1	0
P_2	20	3.25	3.25	3	2.66	2	1.5
P_3	9	3.5	3.5	2.25	1.75	1.33	1
P_4	5	1.5	1.5	1.5	1.5	1	1
P_5	7	2.33	2.33	1.75	2	1	0

	size	0.6	0.7	0.8	0.9	1
P_1	10	0	0	0	0	0
P_2	20	1.5	1	0	0	0
P_3	9	0	0	0	0	0
P_4	5	1	1	0	0	0
P_5	7	0	0	0	0	0

Table 1: Fragments Granularity.

be improved by considering some terms similar, e.g., drug, medicine.

We finally evaluated the scalability of our decomposition algorithm. The tests were conducted on a laptop with Core Intel i5 processor, (2.27GHz*2), 4 GB memory, running Microsoft Seven. We have tested the execution on 3 processes: the first one containing 270 activities performs in 1.605 seconds, the second process containing 90 activities performs in 0.875 seconds, and the third process containing 3 activities performs in 0.531 seconds. Thus, the algorithm can be performed with big processes with a fair execution time.

6 Related Work

Several approaches have been proposed in the area of reusing business process fragments. Nevertheless, it is not well explored when it is about automatically retrieving reusable fragments and even less when it comes to ensure the privacy of sensitive information that may be inferred intentionally or not. In the following, we present some existing approaches related to business process decomposition and privacy-preserving in business process reuse.

Huang et al. (2010) propose a workflow decomposition mechanism for reuse purposes. Their technique aims to provide reusable fragments in a bottom-up fashion. Following Huang et al. (2010), processes are organized into a hierarchy of reusable fragments. The approach then computes the interdependence between each fragment activities. This approach, however, lacks semantics where it manipulates the co-occurrence of activity pairs enclosed within each fragment.

In the same thinking, Rosa et al. (2010) have proposed an approach for merging a set of processes in a pairwise fashion. The merge consists in capturing the common connected activity regions of the initial processes and adds independent parts. This provides a unique version of multiple processes that share some elements. Common regions may be retrieved and reused as part of new processes. While regions structurally fit the fragment's definition, they may however enclose irrelevant elements for the designer. Business process patterns are addressed by La Rosa et al. (2011) to reduce the complexity of business process structures. Four out of twelve proposed patterns foster the reusability of the reduced portions. This approach is also essentially structure-centric and does not provide semantics grouping concerns.

Smirnov et al. (2011) provides a semantic approach to abstract business process models into high level views. This consists in generating coarse-grained activities, a.k.a. clusters, sharing the same property

⁷<http://code.google.com/p/colibri-java/source/browse/#svn%2Ftrunk%2Fcolibri>

values over several property types. The proposed approach is based on a binary VSM handling property values, i.e., only property types are weighted. Our approach permits such manipulations. Moreover, our approach assigns a weight for each property value making the similarity computation more significant for activities.

Furthermore, Huang et al. (2010), Rosa et al. (2010), La Rosa et al. (2011), Smirnov et al. (2011) do not address any privacy-preserving mechanism. Indeed, sensitive information may freely be inferred by malicious activities that fragments are linked to.

Khalaf & Leymann (2012) and Khalaf & Leymann (2006) handle business process partitioning in order to assign some given functionalities to outside partners. The partitioning is made in such a way to maintain the behavior of the original process. Nevertheless, the partitioning is mainly performed manually. Indeed, the partitioning task follows fixing the corresponding activities for each partner.

The work presented by Ivanovic et al. (2010), proposes a fragment identification approach for outsourcing portions of a business process, while dealing with predefined privacy policies. The latter are used to restrict access to certain information to third parties. The decomposition, while it ensures the privacy of sensitive information, focuses only on the information routed between activities. The generated fragments do not involve semantics features to enable reusing them later.

7 Conclusion

We have presented a novel approach in order to provide useful, privacy-aware and reusable fragments. This is ensured by the proposed process decomposition mechanism that (i) performs according to the designer needs (requirement-driven), and (ii) takes into account privacy constraints avoiding sensitive information inferences. Furthermore, fragments are reusable and may easily be integrated into new processes as they depict a connected structure. The integration is enabled as fragments have gluing points consisting of dangling control flows.

Further improvements can be added on similarity computation. This can be achieved using ontologies. Our future work is directed towards this axis.

References

- Abrial, J. R. (1996), *The B-Book: Assigning Programs to Meanings*, Cambridge University Press.
- Eberle, H., Unger, T. & Leymann, F. (2009), Process fragments, in 'On the Move to Meaningful Internet Systems: OTM 2009', Vol. 5870 of *Lecture Notes in Computer Science*, Springer, pp. 398–405.
- Fox, C. (1992), 'Lexical analysis and stoplists', *Information Retrieval: Data Structures & Algorithms* pp. 102–130.
- Ganter, B. & Wille, R. (1999), *Formal concept analysis - mathematical foundations*, Springer.
- Huang, Z., Huai, J., Liu, X. & Zhu, J. (2010), Business process decomposition based on service relevance mining, in 'Web Intelligence', pp. 573–580.
- Ivanovic, D., Carro, M. & Hermenegildo, M. V. (2010), Automatic fragment identification in workflows based on sharing analysis, in 'ICSOC', Vol. 6470 of *Lecture Notes in Computer Science*, pp. 350–364.
- Khalaf, R. & Leymann, F. (2006), Role-based decomposition of business processes using bpel, in 'ICWS', pp. 770–780.
- Khalaf, R. & Leymann, F. (2012), 'Coordination for fragmented loops and scopes in a distributed business process', *Information Systems* pp. 593–610.
- Khalaf, R. Y. (2008), Supporting business process fragmentation while maintaining operational semantics: a BPEL perspective, PhD thesis, Institute of Architecture of Application Systems, University of Stuttgart.
- La Rosa, M., Wohed, P., Mendling, J., ter Hofstede, A. H., Reijers, H. A. & van der Aalst, W. M. (2011), 'Managing process model complexity via abstract syntax modifications', *Industrial Informatics, IEEE Transactions on* 7(4), 614–629.
- Mancioppi, M., Danylyevych, O., Karastoyanova, D. & Leymann, F. (2011), Towards classification criteria for process fragmentation techniques, in 'Business Process Management Workshops', pp. 1–12.
- Markovic, I. & Pereira, A. (2008), Towards a formal framework for reuse in business process modeling, in 'Business Process Management Workshops', Springer, pp. 484–495.
- Nuutila, E. & Soisalon-Soininen, E. (1994), 'On finding the strongly connected components in a directed graph', *Information Processing Letters* 49(1), 9–14.
- Ouyang, C., Dumas, M., Ter Hofstede, A. & Van Der Aalst, W. (2007), 'Pattern-based translation of bpmn process models to bpel web services', *International Journal of Web Services Research (JWSR)* 5(1), 42–62.
- Porter, M. F. (1997), An algorithm for suffix stripping, in K. Sparck Jones & P. Willett, eds, 'Readings in information retrieval', Morgan Kaufmann Publishers Inc., pp. 313–316.
- Rosa, M. L., Dumas, M., Uba, R. & Dijkman, R. M. (2010), Merging business process models., in 'OTM Conferences (1)', pp. 96–113.
- Salton, G. & Buckley, C. (1988), 'Term-weighting approaches in automatic text retrieval', *Information processing & management* 24(5), 513–523.
- Salton, G., Wong, A. & Yang, C. (1975), 'A vector space model for automatic indexing', *Communications of the ACM* 18(11), 613–620.
- Schumm, D., Karastoyanova, D., Kopp, O., Leymann, F., Sonntag, M. & Strauch, S. (2011), 'Process fragment libraries for easier and faster development of process-based applications', *Journal of Systems Integration* 2(1), 39–55.
- Schumm, D., Leymann, F., Ma, Z., Scheibler, T. & Strauch, S. (2010), 'Integrating compliance into business processes: Process fragments as reusable compliance controls', *Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI10)*.
- Smirnov, S., Reijers, H. A. & Weske, M. (2011), A semantic approach for business process model abstraction, in 'CAiSE', pp. 497–511.
- Sweeney, L. (2002), 'k-anonymity: A model for protecting privacy', *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10(5), 557–570.

- Ter Hofstede, A., van der Aalst, W. & Weske, M. (2003), 'Business process management: A survey', *Business Process Management* **2678**, 1019–1019.
- Zemni, M. A., Hadj-Alouane, N. B. & Yeddes, M. (2012*a*), An approach for producing privacy-aware reusable business process fragments, *in* 'ICWS', pp. 659–661.
- Zemni, M. A., Hadj-Alouane, N. B. & Yeddes, M. (2012*b*), A semantics-based privacy-aware approach for fragmenting business processes, *in* 'MVDA'.

Poisson Blended Exemplar-based Texture Completion

Hoang M. Nguyen

Burkhard C. Wünsche
Christof Lutteroth

Patrice Delmas

Computer Science Department
The University of Auckland,
Email: justin.nguyen@auckland.ac.nz
{burkhard, p.delmas, lutteroth}@cs.auckland.ac.nz

Abstract

Image inpainting is the process of correcting undesirable changes to an image in an unobtrusive way. The existing literature in this research field describes predominantly techniques designed to correct narrow missing regions, which thus often produce undesirable results when the damaged region is large.

This paper presents a novel exemplar-based image inpainting technique for automatic filling-in missing region of an image. Our solution offers two major improvements compared to existing techniques. Patches for filling in missing regions are identified using an appearance space vector, which not only encodes pixel colours, but also colour gradients, feature distances and other measures for computing image similarity. In order to speed up the search for a matching patch we use a Principal Component Analysis to reduce the size of a feature vector used for patch comparison.

The second major improvement is the technique used combine patches filling in a missing region. In order to avoid visible seams we use a Poisson-guided interpolation to blend patches.

Our evaluation and comparison with existing techniques demonstrates significantly improved performance for inpainting missing image regions.

Keywords: texture inpainting, texture reconstruction, image information recovery

1 Introduction

The problem of modifying an image to revert deterioration in a non-detectable way has long been an intensive research field in computer graphics. Generally, two classes of image inpainting techniques have been explored and studied: *pixel-based* and *exemplar-based* texture inpainting. Pixel-based inpainting methods attempt to reconstruct missing or damaged regions one pixel at a time. This class of methods is often fast and produces good results for small regions. *Pixel-based* approaches work by propagating pixel values along contours of equal luminance and computing the value of a “missing” pixel based on its surrounding “good” pixels. The method tends to produce blurred outputs for larger regions.

Exemplar-based inpainting attempts to construct the damaged regions by searching for the best-fitting patches and copying them over to the missing region.

These methods are often not as efficient as pixel-based techniques due to the time consuming process of finding best-fitting patches. However, they are most suitable for dealing with larger missing regions. Efficiently and accurately determining the patch best fitting a missing region is one of the key problems of this class of methods. Furthermore, in most applications it is not possible to fill a missing region with a single patch. Hence multiple patches must be copied, which results in partial overlaps and consequently visible seams along patch boundaries.

In this paper, we present a new Poisson-exemplar-based method for inpainting a missing region in an image. In order to improve the efficiency and accuracy of the best-fit patch finding stage, we forgo the conventional *Sum of Squared Differences (SSD)* score technique and employ so-called appearance space attributes to help with this task. For each pixel, the appearance space attribute contains not only the RGB color value, but also its signed feature distance, gradient in both directions and *HSB* values. This provides far more accurate information about each pixel and its neighbourhood, and hence makes it possible to find better matching image regions. Selected patches are fused and blended together using a Poisson interpolation technique, significantly reducing visible seams.

The remainder of this paper is organised as follows: After a brief discussion on the state-of-the-art of image inpainting in section 2, we describe our inpainting algorithm in section 3. Section 4 presents some of our results. Section 5 concludes our paper.

2 Related Work

An analysis of the literature reveals two key classes of algorithms for image in-painting. The first group of methods approaches the problem of texture inpainting from a pixel-based perspective. These methods reconstruct a missing region by computing color values for each pixel one at a time starting from the missing region’s boundary and processing inward until the entire region is filled.

Exemplar-based techniques fill missing regions by searching for patches matching the region boundary and inserting these texture patches such that discontinuities with the valid image region are minimised. Both approaches are able to produce high-quality results.

The arguably best known and successful algorithm amongst pixel-based inpainting methods was proposed by Bertalmio *et al.* (2000). The authors attempt to replicate the manual inpainting by propagating the known color values into a missing region along so called isophotes, representing the smallest spatial change of color values and structures.

Drori *et al.* (2003) present a similar approach using

adaptive circular fragments to operate on different scales to capture both global and local structures and approximating the missing region.

Ignecio *et al.* (2004) present an inpainting technique based on a fast marching method for level set applications. The method is simple and considerably more efficient than other pixel-base methods.

As pixel-based methods synthesize texture information of a pixel by examining only its neighborhood information, these methods only yield good results for small and narrow missing regions. For larger holes, the reconstructed image regions tend to be blurry and visually obtrusive.

Exemplar based methods are hence becoming increasingly popular for generating large missing texture patches.

Criminisi *et al.* (2003) propose a method that reconstructs missing texture regions by iteratively selecting a “best-fit” rectangular patch and copy it over to the target region. The order in which boundary pixels of the missing region are processed is based on the amount of information available for that pixel and whether it has any prominent features.

In this paper we present a new algorithm, which uses a similar patch search and insertion concept, but offers two key improvements. First, instead of using only pixel colours for the patch finding process, we employ an appearance space which encapsulates much more information. Second, Criminisi’s method does not handle seams along patches. Inevitably, their results often look unrealistic. In contrast, our method smoothly fuses patches together to remove all visible seams.

Cheng *et al.* (2005) updated the priority equation of Criminisi *et al.* (2003) and made it adjustable to the structural and textural information specific to an image. Ignecio *et al.* (2007) extended the concept of Criminisi’s method and applied it in the wavelet domain. Their method computes the fill-priority by first transforming the image and the provided binary mask and then use wavelet coefficients and a similarly defined priority to define the fill-order.

3 Algorithm

3.1 Fundamentals

In order to facilitate understanding of our technique and comparison with alternative techniques we adopt the notation used in the image inpainting literature. Let Ω indicate the *target region* to be inpainted. Note that there is no restriction imposed on the topology of Ω . Let $\delta\Omega$ denote the boundary of the target region. The boundary is sometimes referred to as “*fill front*” since this contour evolves inward as the algorithm progresses. The *source region* is denoted as Φ , which in our algorithm remains unchanged throughout the processes. Let Ψ_p be a window centered at the point p .

The main principle behind exemplar-based methods is simple. As with all exemplar-based texture inpainting methods (e.g. Efros *et al.* (1999), Criminisi *et al.* (2003)), the size of the template patch (window) must be specified in advance. In our algorithm, the default patch size is empirically set to 11×11 (refer to (Nguyen *et al.* 2013) for a more detailed analysis of the algorithm parameters). To synthesize the missing region, the following procedure is repeated until all pixels are filled.

For a given pixel p on $\delta\Omega$, find a patch Ψ_q where $q \in \Phi$ such that Ψ_q is most similar to those parts that are already filled in Ψ_p . The missing texture in-

formation is then transferred from Ψ_q to Ψ_p . Figure 1 illustrates this process.

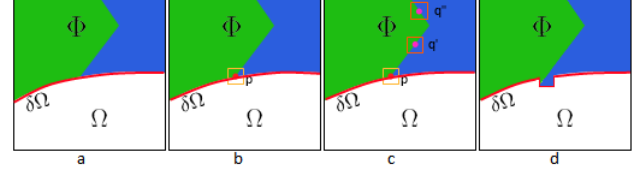


Figure 1: Exemplar-based texture inpainting. a) The original input image with the source region Φ , the target region Ω and the boundary $\delta\Omega$. b) Attempting to reconstruct an area around pixel p . c) Several likely candidate matches are found in the source region. d) The content of the best patch is copied over, resulting in a partial filling of Ω (adapted from Criminisi *et al.* (2003)).

The filling order is critical for inpainting techniques in general, and even more so for non-parametric texture synthesis. Traditionally, the most well-known method has been “*onion peel*”, where the inpainted region is synthesised in concentric layers inwardly (Criminisi *et al.* 2003). Therefore, in our method we iteratively shrink the gap of the inpainted region by continuously transferring colours from source regions to patches centered at boundary pixels.

3.2 Determining the Filling-Order

Given a set of boundary pixels, the objective is to determine the order or priority of the pixels to be processed. This task is accomplished as followed. For each boundary pixel p , let Ψ_p be a patch centered around p . The priority of p is defined as by Criminisi *et al.* (2003):

$$Priority(p) = Confidence(p) * Data(p) \quad (1)$$

The *confidence term*, which quantifies the amount of reliable information in the pixel’s neighborhood, is defined as:

$$Confidence(p) = \frac{\sum_{q \in \Psi_p \cap \Omega} Confidence(q)}{|\Psi_p|} \quad (2)$$

where $|\Psi_p|$ is the area of the patch Ψ_p and Ω denotes the target region to be inpainted. The function $Confidence(q)$ returns 1 if q is already filled and 0 otherwise. The confidence term aims to boost the priorities of patches that have more already-filled pixels, allowing them to be synthesized first.

The *data term*, which defines the strength of the isophotes arriving at the boundary, is defined as:

$$Data(p) = \frac{|\nabla I_p^\perp \cdot n_p|}{\alpha} \quad (3)$$

where ∇I_p^\perp represents a vector that is orthogonal to the gradient vector at p , n_p is the normal at p , and α denotes a normalisation factor ($\alpha = 255$ for RGB-colour images). The purpose of this data term is to find matching patches preserving linear texture features, such as straight lines or curves, and therewith extending the linear features gradually inwards.

The *confidence values* for all boundary pixels are computed and the pixel with the highest confidence value is processed first.

3.3 Candidate Patch Identification

The next task is to search for a patch in the image that retains the highest resemblance to the processed patch. This is achieved by iteratively traversing through each pixel of the image outside the missing region and computing the similarity of the patch centered around that pixel and the original patch. Instead of using the standard *SSD* to measure the similarity of two given patches, we employ *appearance space attributes*, which provide much more information and thus improve the search result.

When searching for a matching patch we consider for each pixel an 11×11 pixel neighbourhood. For each pixel of this neighborhood we consider *RGB* colours, the *gradient vector* as well as the signed Euclidean distance to the closest dominant feature to the original texture. The entire information is encapsulated into an $11 \times 11 \times (3 + 2 + 1) = 726$ -dimensional vector.

Determining the similarity of two given patches by comparing two 726-dimensional vectors is not efficient. In order to make the appearance space more practicable, the 726-dimensional vectors are projected into low-dimensional vectors using *principal component analysis (PCA)* (Lefebvre *et al.* 2006, Manke *et al.* 2009). In our method, the dimensionality is reduced to 12, which from our experiments on different types of images produced the best results.

The clear advantage of attribute space over the conventional SSD is that the attribute space approach permits any meaningful information about the pixels and their surrounding to be embedded for matching purposes. By reducing the dimensionality, the computation time can be kept manageable.

3.4 Patch Fusion

The final step is to replicate the content of the candidate patch and smoothly blend it with the target region. We employ a *Poisson-guided interpolation* approach proposed by Perez *et al.* (2003) for this task. The principle behind this is fairly straightforward.

Suppose Ψ_B is the candidate patch to be copied and fused over the target patch Ψ_A , and let ∂_A and ∂_B be the boundaries of the target and candidate patches respectively. The goal is to adjust the colour information of Ψ_B , while preserving the *relative information* (image gradient) as much as possible, so that the transition between the newly modified patch Ψ_C and the rest of the image is gracefully blended. This is accomplished as follows:

First, the values of the boundary pixels of Ψ_C are initialised to be equal to the corresponding values of the boundary pixels of Ψ_A . This is to ensure that the isophotes arriving at the boundary are properly maintained.

$$\Psi_{C(x,y)} = \Psi_{A(x,y)} \quad \forall (x,y) \in \partial_B \quad (4)$$

Next, each colour channel's value of the remaining interior pixels within Ψ_C are independently adjusted to be consistent with the boundary pixels while constraining the image gradient to be identical to that of Ψ_B .

$$\nabla C(x,y) = \nabla B(x,y) \quad \forall (x,y) \in \Psi_C \setminus \partial_C \quad (5)$$

where $\nabla(x,y)$ denotes the image gradient at the pixel (x,y) , and $\nabla C(x,y)$ and $\nabla B(x,y)$ are defined as

$$S_1 = \sum_{(x+\delta x, y+\delta y) \in \Psi_A} C(x+\delta x, y+\delta y) \quad (6)$$

$$S_2 = \sum_{(x+\delta x, y+\delta y) \in \partial_A} A(x+\delta x, y+\delta y) \quad (7)$$

$$\nabla C(x,y) = |N| \quad C(x,y) - S_1 - S_2 \quad (8)$$

where N is the number of valid pixels. A pixel is considered valid if it is inside the processed patch.

$$\nabla B(x,y) = \sum_{(x+\delta x, y+\delta y) \in \Psi_A \cup \partial_A} B(x,y) - B(x+\delta x, y+\delta y) \quad (9)$$

δx and δy designate a set of 4-connected neighbours around x and y . The equation 5 can then be expressed in the form of a system of linear equations with i variables (i is the number of pixels in $\Psi_{C(x,y)}$), and can be solved using an *iterative matrix solver* such as the *Jacobi Method*.

4 Evaluation

In this section, we investigate the effect of different algorithm parameters and compare its performance in comparison with popular existing algorithms.

4.1 Evaluation of Appearance Space Attributes



Figure 2: a) The image contains three black square regions where image information was removed in order to fill the regions using image inpainting techniques. b) The original input image. c) Inpainted image using SSD. d) Reconstructed image using appearance space attributes.

We have tested different appearance space attributes and found that the best matching patches are found by using a combination of *gradient values*,

signed feature distance and RGB colours. Adding additional information such as HSB channels and neighbourhood variance increases the cost and produces no visible improvement. The difference between using standard SSD and using *Appearance Space Attribute* is demonstrated in figure 2.

The reconstructed image using SSD has a poor quality. All three missing regions have been filled using patches which do not properly match the boundary of the hole. Reconstructing the image using *appearance space attributes* demonstrates clear improvements. Although there are still some artefacts around the eyebrow region, these will be mended during the blending process.

4.2 Blending versus Non-Blending

We present several examples to demonstrate the effectiveness of our algorithm compared with existing techniques.

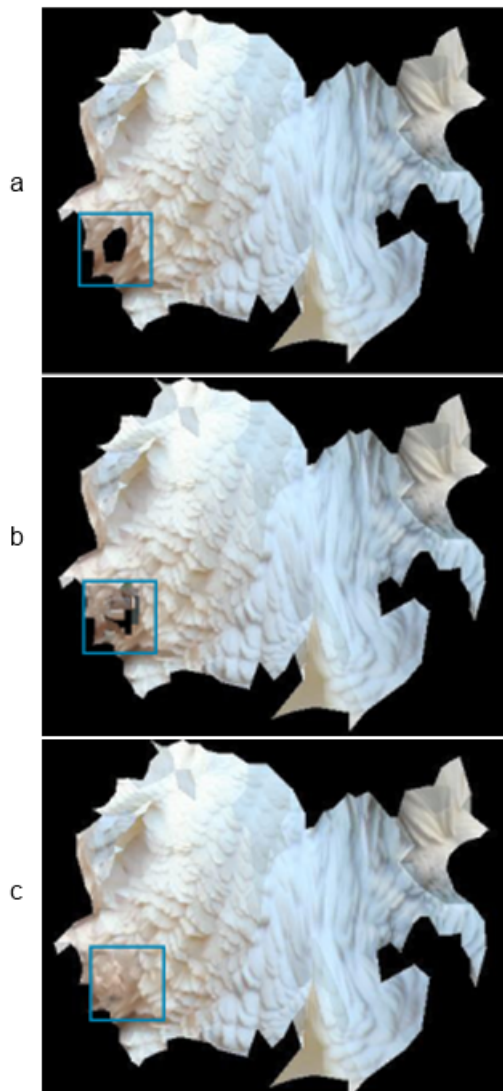


Figure 3: Reconstructed texture with and without blending. a) Input texture. b) Result using the method by Criminisi *et al.* (2003). c) Our result where patches are blended together using Poisson-guided interpolation.

Figure 3 illustrates inpainting results on a part of a texture atlas. Note that without blending the overlapping inserted patches have visible seams. Furthermore, not the entire missing region was filled. In

contrast, using Poisson-guided interpolation produces a very realistic and complete result.



Figure 4: Inpainted texture without blending (left), and with blending (right).

Figure 4 demonstrates another example of how Poisson-guided interpolation improves the overall inpainting results. Notice how the eyebrow now appears more natural.

4.3 Evaluation against Other Inpainting Methods

In this section, we evaluate our method against some of the best known texture inpainting methods described in the literature.

Figure 5 shows an example in which spots of a cheetah are removed using several well-known inpainting methods.

Ignecio *et al.* (2004)'s method is the most efficient. It took approximately 45 seconds to accomplish the task. However, the result is unsatisfactory. Most remnants of the cheetah's spot are still evident. Additionally, as some of the spots are relatively large, the inpainted image regions appear blurry.

Bertalmio *et al.* (2000)'s method takes a little more time to process but generate a better result (57 seconds). However, as with Alexandru's method, it fails to remove some of the spots completely. For some large gaps, blurry textures can be seen. Additionally, in some cases (for some spots) colours are not propagated correctly resulting in patches with colours not consistent with the regions neighborhood.

The bottom image in figure 5 shows that our method successfully removes all spots with texture information consistent with the surrounding image region. However, there are some small regions at the tail where the algorithm was unable to reconstruct the fur without spots correctly (part of the tail's texture extrudes to the neighbouring area). This is probably due to the fact that the selected window size was relatively large in this case. Overall, our algorithm works well and produces good results compared to other inpainting methods.

Figure 6 presents another example comparing different inpainting methods. Bertalmio *et al.*'s (2003) method was unable to fill the missing region. This is probably due to the fact that intensity values from the source region are not properly propagated inwardly. Traditional exemplar-based techniques such as Criminisi *et al.* (2003) produce fairly good result, although there is a large artefact in the reconstructed region. Ignecio *et al.*'s (2004) method produced a reasonably good result, although the inpainted region appears very blurry. Some parts of the window in the image are not reconstructed properly. Our algorithm performs well in this test case. Although the inpainted region still exhibits slight blurriness, the overall structure of different scene components has been correctly reconstructed.

5 Conclusion and Future Work

In this paper we have presented a novel image inpainting algorithm for reconstructing large missing texture regions from digital photographs. The results of this inpainting process is a new image in which the deterioration has been “inpainting” and reverted in such a way that few visible traces of it remain.

The basic idea of our approach is to replicate missing textures by searching for “best-fit” texture patches in the source regions and smoothly insert these patches into the missing region to form the final result. The filling-order is determined using pixels’ confidence value, which is defined by the amount of information available for that pixel and the image isophotes. This allows our algorithm to propagate both linear and round texture features into the target region.

Our solution offers two major improvements compared to existing techniques. Patches for filling in missing regions are identified using an appearance space vector, which not only encodes colour differences between regions, but also colour gradients, feature distances and other measures for image similarity. In order to speed up the search for a matching patch we use a Principal Component Analysis to reduce the size of a feature vector used for patch comparison. The second major improvement is the use of Poisson-guided interpolation to blend patches.

We have evaluated our method’s performance against some of the best known inpainting methods described in the literature and found that our results are superior.

References

- Marcelo Bertalmio, Guillermo Sapiro, Vicent Caselles and Coloma Ballester (2000), Image Inpainting, *in* ‘Proceeding SIGGRAPH ’00 Proceedings of the 27th annual conference on Computer graphics and interactive techniques’, pp. 417–424.
- Iddo Drori, Daniel Cohen-Or and Hezy Yeshurun (2003), Fragment-Based Image Completion, *in* ‘ACM Transactions on Graphics’, Vol. 22, pp. 417–424.
- A. Criminisi, P. Prez and K. Toyama (2003), Object Removal by Exemplar-based Inpainting, *in* ‘ACM Transactions on Graphics’, pp. 721–728.
- Wen-Huang Cheng, Chun-Wei Hsieh, Sheng-Kai Lin, Chia-Wei Wang and Ja-Ling Wu (2005), Robust algorithm for exemplar-based image inpainting, *in* ‘In Proceedings of CGIV’, pp. 64–69.
- Ubirate Ignecio and Cleudio R Jung (2007), Block-based image inpainting in the wavelet domain, *in* ‘Visual Computing’, pp. 733–741.
- Alexandru Telea (2004), Journal of Graphics Tool, *in* ‘An image inpainting technique based on the fast marching method’, Vol. 9, pp. 23–34.
- Patrick Perez, Michel Gangnet and Andrew Blake (2003), ACM TRANSACTIONS ON GRAPHICS, *in* ‘Poisson image editing’, Vol. 9, pp. 23–34.
- Alexei Efros and Thomas Leung (1999), In Proceeding of ICCV, *in* ‘Texture synthesis by non-parametric sampling’, pp. 1033–1038.
- Felix Manke and Burkhard Wunsche (2009), Image and Vision Computing New Zealand 2009. IVCNZ 09, *in* ‘Analysis of appearance space attributes for texture synthesis and morphing’, pp. 85–90.

Sylvain Lefebvre and Hugues Hoppe (2006), ACM SIGGRAPH 2006 Papers, *in* ‘Appearance-space texture synthesis’, pp. 541–548.

Hoang Minh Nguyen, Burkhard Wunsche, Patrice Delmas and Christof Lutteroth (2013), Image and Vision Computing New Zealand 2013. IVCNZ 13, *in* ‘Enhanced Patch-Based Image Inpainting with Poisson Interpolation’, 2013 [Accepted for publication].

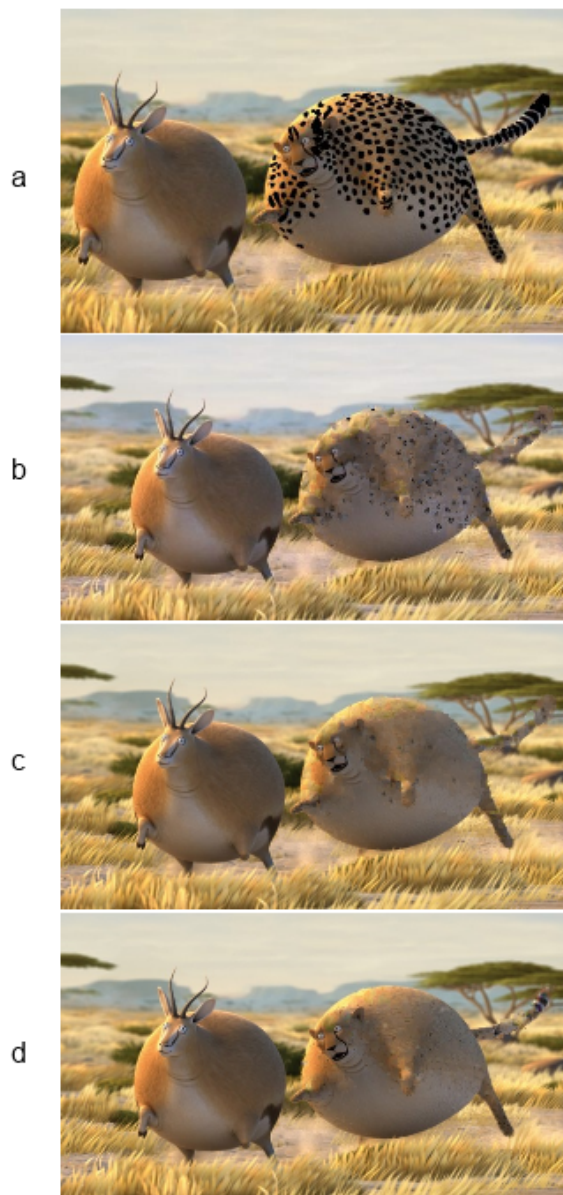


Figure 5: A scene from the animation series - "Rolling Safari". a) The original input image. All black spots are considered missing regions and we employ image inpainting techniques in order to fill the black spots with colour information consistent with the remaining fur colour of the cheetah. b) The inpainted result using Alexandru's method Ignecio *et al.* (2004). c) Result obtained with Bertalmio *et al.* (2000) method. d) Result obtained using our image inpainting technique.

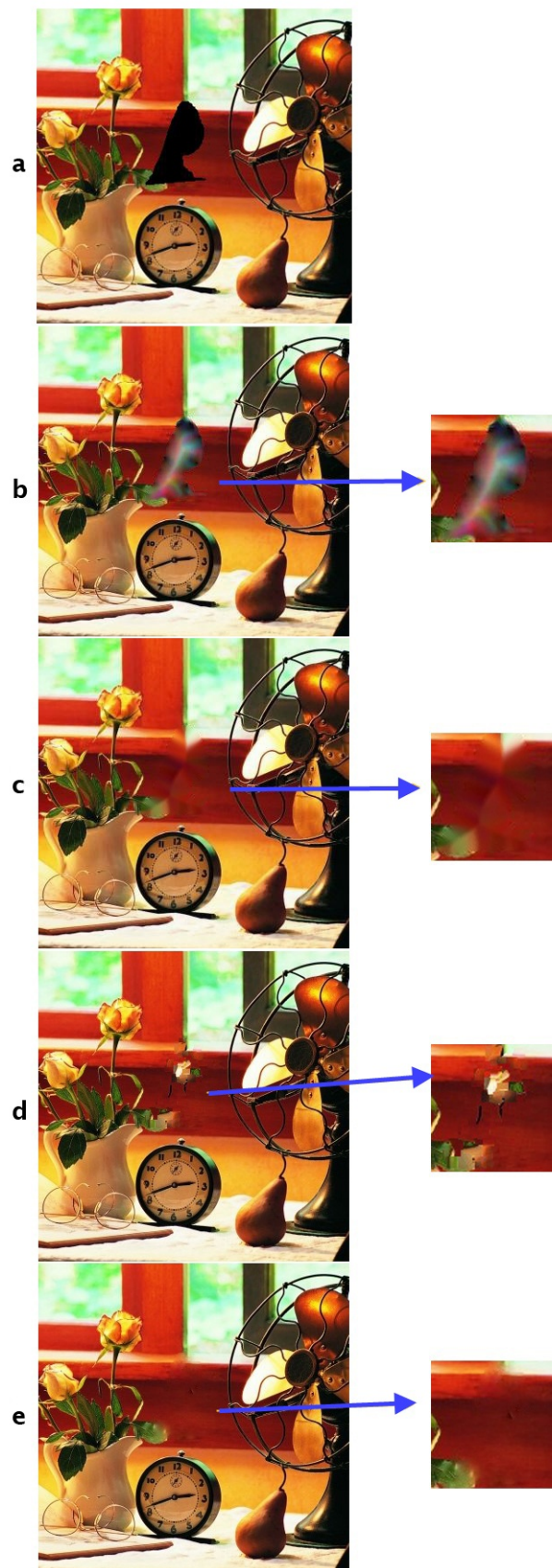


Figure 6: a) The input image. Image inpainting results obtained using the algorithms from: (b) (Bertalmio *et al.* 2000), (c) (Telea *et al.* 2004), (d) (Criminisi *et al.* 2003) and (e) our method.

A Replication and Reproduction of Code Clone Detection Studies

Xiliang Chen¹

Alice Yuchen Wang¹

Ewan Tempero²

¹ Electrical and Computer Engineering
The University of Auckland
Auckland, New Zealand

xche185@aucklanduni.ac.nz, ywan478@aucklanduni.ac.nz

² Computer Science
The University of Auckland
Auckland, New Zealand
e.tempero@auckland.ac.nz

Abstract

Code clones, fragments of code that are similar in some way, are regarded as costly. In order to understand the level of threat and opportunity of clones, we need to be able to efficiently detect clones in existing code. Recently, a new clone detection technique, CMCD, has been proposed. Our goal is to evaluate it and, if possible, improve on the original. We replicated the original study to evaluate the effectiveness of basic CMCD technique, improved it based on our experience with the replication, and applied it to a 43 open-source Java code from the Qualitas Corpus. We confirmed the effectiveness of the original technique but found some weaknesses. We improved the technique, and applied our improved technique. We found that that 1 in 2 systems had at least 10% cloned code, not counting the original, indicating that cloned code is quite common.

1 Introduction

Code clones, fragments of code that are similar in some way, are regarded as costly (Juergens et al. 2009, Li & Ernst 2012). There is some evidence that the number of clones in any given system can be non-trivial (Baker 1995, Baxter et al. 1998, Falke et al. 2008, Juergens et al. 2009, Schwarz et al. 2012). This means, that if code clones do represent an unnecessary cost, then their existence is both a threat to the software quality and an opportunity for improvement. In order to understand the level of threat and opportunity, we need to be able to detect clones in existing code. If there are few code clones, then their cost is not so important. In order to understand the cost associated with clones, we need to be able to identify clones and determine the cost due to their existence.

To detect clones, we need a means that is both efficient and effective. If the techniques used to detect clones have poor accuracy, then effort will be wasted identifying false positives, and the results will be uncertain due to the unknown false negatives. If the clone detection techniques are inefficient, it will be difficult to gather enough data to be useful. Many existing clone detection techniques are quite expensive, particularly in time, or trade off time for accuracy.

Recently, Yuan & Guo (2011) demonstrated a new technique, CMCD (for Count Matrix Clone Detection), for detecting clones that is very efficient and appears quite effective. The basic idea is language-independent and relatively straight-forward to implement, so that it may be possible to produce good clone detectors for many languages quite quickly. In fact it is very simple, especially compared to other clone detection systems, raising questions as to whether it can be generally effective. If it is as good as it appears, CMCD has very good potential for significantly increasing the number and size of empirical studies, thus improving our understanding of the impact of code clones. While the paper explained CMCD well, the evidence it provided for how good CMCD is was quite weak, so there is still a question as to its effectiveness. In this paper, we answer these questions. Specifically, we confirm the original claims, present an improved version of the technique, demonstrate that it is effective, and present the results from using the new technique on a large corpus of Java code. These results indicate that about half the systems we studied had more than 10% cloned code.

In scientific study, it is not enough to observe something once to be convinced of the validity of some theory, the observations must be *repeated*. As Popper said, “We do not take even our own observations quite seriously, or accept them as scientific observations, until we have repeated and tested them.” (Popper 1968) While there have been a number of empirical studies reporting the degree to which code clones exist, there are various issues that exist with those studies. Their goal is usually not to determine to what degree clones exist, but to demonstrate how effective a given clone detector is. They rarely examine the same systems as each other, so it is not obvious how to compare the results with each other. They also are generally quite small. We are aware of only 2 large studies, and they are both of systems written in C (Uchida et al. 2005, Kamiya et al. 2002). Many more studies are needed, and there is a need to perform large studies of other languages. Our work is another step in this process. In this paper we present one of the largest code clone detection studies to be undertaken in Java.

What constitutes a useful repetition is a matter of some debate (Drummond 2009), however in this paper we consider what Cartwright refers to as *replicability* — doing the same experiment again — and *reproducibility* — doing a new experiment (Cartwright 1991). In this paper we replicate (as much as possible) the study by Yuan and Guo to demonstrate the validity of CMCD and we attempt to reproduce the results of various empirical studies undertaken to de-

Copyright ©2014, Australian Computer Society, Inc. This paper appeared at the Thirty-Seventh Australasian Computer Science Conference (ACSC2014), Auckland, New Zealand, January 2014. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 147, Bruce H. Thomas and David Parry, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.

termine the degree to which code clones exist in Java code.

The rest of the paper is organised as follows. In the next section we present some of the literature on empirical studies of code clones to determine what has been established so far. In section 3, we summarise the original CMCD technique. In section 4, we describe the modifications we have made to CMCD, and how we carried out the replication and reproduction studies. We present our results in section 5 and discuss them in section 6. Finally we present our conclusions and discussion future work.

2 Background and Related Work

In this section we introduce the concepts generally associated with clone detection and give an overview of clone detection related research. The literature in clone detection research is quite extensive, and so we necessarily can only give a sample here. Roy et al. (2009) provide a good survey and we use their terminology below. We detail only the work that is directly relevant to ours.

A *code fragment* is any sequence of code lines that can be any granularity, such as a complete method definition or sequence of statements inside an if-statement (Bellon et al. 2007). A *clone pair* is defined by having two code fragments that are similar by some given definition of similarity. When more than two fragments are similar, they form a *clone cluster* or clone group. There are two main types of similarity between code fragments: textual similarity and semantic similarity. Two fragments have textual similarity when the text they contain matches to a large degree. This might be the consequence of copying and pasting one fragment to the other, perhaps with minor modifications. Two fragments are semantically similar if they have similar functionality, but may have completely different text. Clone types can be categorised into four types based on both textual and semantic similarities (Roy et al. 2009):

Type-1: Identical code fragments except for variations in whitespace, layout and comments.

Type-2: Syntactically identical fragments except for variations in identifiers, literals, types, whitespace, layout and comments.

Type-3: Copied fragments with further modifications such as changed, added or removed statements, in addition to variations in identifiers, literals, types, whitespace, layout and comments.

Type-4: Two or more code fragments that perform the same computation but are implemented by different syntactic variants.

Most of the clone detection techniques can be summarised into four main categories: textual, lexical, syntactic and semantic (Roy et al. 2009).

Textual approaches compare the source code with little or no transformation. In most cases raw source code is used directly in the clone detection process. Such approaches must cope with variation of all kinds, including in whitespace. An early such approach was described by Johnson (1994).

Lexical approaches transform the source code into a sequence of lexical tokens using compiler-style lexical analysis. Comparison is then done between sequences of tokens to identify common subsequences.

These approaches easily deal with variation in whitespace, layout, and comments. Also, variations in identifiers or literals can also be dealt with. An example of this approach was described by Baker (2007).

Syntactic approaches use a parser to convert the source code into parse trees or abstract syntax trees (ASTs). Comparison is then done between the trees for commonality, possibly using structural metrics. The work by Baxter et al. (1998) is perhaps the best known of these approaches.

Semantic approaches use static program analysis, which does a more sophisticated comparison than just at the syntax level. One technique that is used is to represent the code as a program dependency graph (PDG) and then analyse that. An example of this approach uses backward slicing (Komondoor & Horwitz 2001).

Roy et al. (2009) describe 4 *scenarios* giving examples of each of the categories described above, with further sub-scenarios for each category, a total of 16 examples. They then evaluated more than 40 techniques with respect to these 16 examples, providing both clear criteria (at least as a starting point) for what might constitute a clone, and a comprehensive summary of that the state-of-the-art at that time could handle.

As mentioned in the introduction, our work is based on the CMCD technique developed by Yuan & Guo (2011), which we will detail in the next section. This technique is a syntactic approach, specifically it falls in to the category Roy et al. call metrics-based approaches. Generally, these approaches make measurements of the code fragments using one or more metrics. The value of the measurements indicate how similar the code fragments are. One example of this is by Mayrand et al. (1996) who gather measurements from a number of metrics, such as different characteristics of the layout of the code, number and type of statement, and characteristics of control flow such as number of decisions, average nesting level, and number of exits. Another example is by Kontogiannis (1997), who uses more common metrics such as Fan-in, Fan-out, and Cyclomatic Complexity Number.

Once measurements are produced, the existence of clones is determined by the similarity of the measurements. The comparison of measurements may be done in different ways, depending on who the measurements look like. For example, Kontogiannis groups the measurements into 4 categories, and then compares each category separately. The results of the 4 comparisons are then used to produce an ordinal scale measurement capturing how different (or similar) two code fragments are. Another example is by Davey et al. (1995), who use a self organising neural net to do the comparisons.

With all the variations in techniques, the question that naturally arises is which is the best? Unfortunately there is no clear answer, not only because the answer depends on the reason for detecting clones, but also because there has been insufficient comparison between the techniques.

For an example on how context might affect which quality attributes of the clone detection technique we desire, consider plagiarism detection in student assignments. In this context, we would probably do this off-line (as a batch process), we might expect the size of the inputs to be relatively small (hundreds, or perhaps a few thousand, lines of code), and we would likely be willing to accept missing some cases (false negatives) in order to ensure not making any false accusations (false positives). On the other hand, to support a code review function in an IDE, we would want real-time information of possibly a large code base,

but would accept a reasonable degree of false positives and false negatives. These two examples represent a trade-off of preferences in performance versus accuracy. Other trade-offs include what constitutes a clone, for example only requiring detection of Type 1 and Type 2 clones, and what granularity of code fragments are to be considered, such as comparing only subroutines or comparing blocks.

As an example of other uses of clone detectors, Li and Ernst examined the degree to which code clones also contained duplicated buggy code (Li & Ernst 2012). Their detector used a semantic approach based on PDGs. They examined 3 systems (Git, Linux kernel, and PostgreSQL). Using the systems' bug reports, they identified the code fragments where the bugs occurred, and then tried to find clones of those fragments. They compared their system against 4 other clone detectors. Their system performed as well or better than the others in both accuracy and performance.

There have been some comparisons of different techniques. Bellon et al. (2007) compared 6 clone detectors that used different approaches over 4 C and 4 Java systems with respect to their accuracy and performance. While there was no clear winner, they noted that the AST-based approaches tended to have higher precision (fewer false positives) but were also longer execution time, whereas token-based approaches had higher recall (fewer false negatives) but were faster. They commented that "if idea from the token-based techniques could be made to work on ASTs, we would be able to find syntactic clones with less effort."

Falke et al. (2008) did a follow up study using the same infrastructure as by Bellon et al. to examine the quality of clone detectors based on suffix trees. They found that using suffix trees was faster than the standard AST matching, but with varying recall and precision.

Two important questions relating to clone detection research are: Is the belief that clones are a problem correct, and; Are there enough clones in real code to matter? Juergens et al. (2009) addressed the first question by developing a new clone detector and applying it to 5 projects, 4 from 2 companies (3 in C#, 1 in Cobol), and one open source (Java). They were particularly interested in what they called *inconsistent* clones, code fragments that differ by more than just simple changes that apply to the whole fragment, such as renaming variables (effectively Type-2 clones). They presented identified inconsistent clones to the developers of the systems, and from the developers determined whether the inconsistency was intentional or not, and whether the clones were faulty or not. From this they concluded that inconsistent clones are a major source of faults. The results by Li & Ernst (2012) also suggest that clones are a problem, by finding clones that contain the same bugs.

There does not appear to have been any systematic large-scale studies to determine the degree to which code clones exist. However, most presentations of new clone detectors provide data from their application that indicates that clones are relatively common.

For example, in an early study, Baker found on the order of 19% of the X Window System (Baker 1995) are either Type-1 or Type-2 clones. Baxter et al. looked at 19 subsystems of a process-control system with 400 KSLOC of C code (Baxter et al. 1998). Their results indicated that on average 12.7% of code was cloned, with at least two subsystems having over 28%.

Juergens et al. found 300 or more clone groups (2 or more code fragments that are clones of each

other) in 4 of the 5 systems. They do not indicate what proportion of the overall code base these groups represent, nevertheless it must be non-trivial.

While Bellon et al.'s goal was to compare detectors, they include information about candidates reported by the tools, the size of code fragments identified as candidates, and information on accuracy in their results. While it is difficult to determine the proportion of cloned code, again it is clear that it must be non-trivial.

Schwarz et al. (2012) examine the repositories of Squeaksource, a repository in the Smalltalk ecosystem. They found 14.5% of all methods strings (560K different methods in 74K classes) were present in at least two distinct repositories.

We are aware of two large empirical studies, both by the same research group. The earlier one studies 125 packages of open source software written in C (Uchida et al. 2005). The size of the systems ranged between 478 and 2,678,939 "LOC". They do not report how they measured LOC. They found there is much variation in how much cloned code there is, but on average they found 11.3%. This group did a later study of the "Packages and Ports Collection" of FreeBSD. The primary goal of this study was to demonstrate their distributed version of CCFinder (Kamiya et al. 2002). They analysed more than 400 million LOC over nearly 6700 projects, which appeared to include many of the systems in their previous study. They found on average 4% of code clones, but there were several cases where the degree of code clones was much higher. However they did not report results for individual projects.

In summary, various studies consistently report that code clones exist to a non-trivial degree, with many measurements of more than 10% being reported. However, most studies are only of a small number of systems, and many of those systems are quite small. What large studies there are examine only systems written in C. Our interest is whether we would see different results in a different language (Java in our case).

3 Original CMCD Technique

In order to make our contribution clear a good understanding of the original CMCD technique is needed, which we provide below. More details are available in the original publication (Yuan & Guo 2011). The modifications we made are described in the next section.

The CMCD technique determines the similarity between two code fragments by modelling each with a *count matrix* and comparing the count matrices. A count matrix is made up of a set of *count vectors*. In the original CMCD, there is one count vector for each variable that appears in the code fragment. The values in a count vector come from a set of *counting conditions* that are applied to the variable that vector represents. The counting conditions represent how a variable is "used". The intuition is, if two code fragments are indeed clones, then a variable in one fragment will have a counterpart in the other fragment that is used in very similar ways, so the count vectors will be very similar. Also, most variables in one fragment will have counterparts in the other fragment, so the count matrices will be very similar. If, on the other hand, the fragments are very different, then there is a high probability that many variables in one fragment will have no obvious counterpart in the other, so the count matrices will look different.

Table 1: The original Counting Conditions (Yuan & Guo 2011)

- 1 Used
- 2 Added or subtracted
- 3 Multiplied or divided
- 4 Invoked as parameter
- 5 In an if-statement
- 6 As an array subscript
- 7 Defined
- 8 Defined by add or subtract operation
- 9 Defined by multiply or divide operation
- 10 Defined by an expression which has constants in it
- 11 In a third-level loop (or deeper)
- 12 In a second-level loop
- 13 In a first-level loop

As the Yuan and Guo noted in the original publication, exactly what constitutes a “use” is maybe not as important as applying the counting conditions consistently. Nevertheless the counting conditions do need to indicate some reasonable notion of “use”. The original counting conditions are shown in Table 1. These counting conditions all are uses of variables that are familiar to any programmer. Clearly other counting conditions are possible as the authors acknowledge, but it is not obvious whether the extra cost of adding more will significantly change the outcome. We return to this point in the next section.

Two count vectors are compared by computing the normalised distance between them. The original technique uses euclidean distance and normalises (roughly) by dividing by the vector lengths (see paper for full details). The resulting distance is in the range [0..1], where 1 means identical.

After computing the count vectors for each variable for each code fragment, the resulting count matrices need to be compared to determine similarity. An issue arises in that, while each variable in one fragment may have a very similar counterpart in the other fragment, this may not be obvious if the order of the count vectors is different in the count matrices, that is, it is not enough to just compare the first row of one matrix with the first row of the other, and so on. CMCD resolves this issue using maximum weighted bipartite matching as follows.

Each row in the two matrices being compared is treated as a vertex in a graph, and each vertex from one matrix has an edge to every vertex in the other matrix. Each edge is weighted by the distance between the two respective count vectors. This results in a weighted bipartite graph. The maximum weighted bipartite matching of this graph is then a pairing of count vector from one matrix with a count vector in the other matrix that maximises the sum of the count vector distances. This sum is then the measure of similarity between the code fragments.

The similarity value may also be normalised, to account for comparing code fragments of different sizes, or have a different number of variables. Also, in case it is possible for two quite different fragments to get a high similarity measurement, a false positive elimination step is applied using heuristics. The authors do not give any details as to what heuristics they use.

The same idea can be used to compare two sets of code fragments — a weighted bipartite graph can be constructed where a vertex is the count matrix for a code fragment, and edges are between vertices from one set to the other weighted by the similarity score between the corresponding code fragments. Again, maximum weighted bipartite matching can be used to determine how similar the two sets are. In this

way two classes can be compared for similarity by treating each method as a code fragment and applying the technique as described above.

Yuan and Guo evaluated their CMCD technique by using it in three different ways. First, they applied it to the 16 scenarios described by Roy et al., demonstrating that it detected all 16 cases. They then applied it to 29 student medium-sized project submissions (7 KLOC – 38 KLOC, 585 KLOC in total). The processing took 123 minutes on relatively standard hardware and they found 2 clone clusters. Manual examination concluded that would have been difficult to identify the clusters through manual inspection. Despite the fact that all projects implemented the same functionality, they did not find any false positives.

The third evaluation method was to analyse JDK 1.6.0_18 (about 2 MLOC). They compared every pair of methods in this code base, ignoring very small methods such as getters and setters. The processing took 163 minutes and found 786 similar methods over 174 clusters. One of the clusters included at least one instance that appeared to contain a fault. They provide no information of how the quality of these results was determined.

The evaluation provided by Yuan and Guo indicates that CMCD has some value, but only one large system was analysed, and it is difficult to judge the quality of its results.

4 methodology

The research questions we would like to answer are:

RQ1 Is the CMCD technique as effective as its authors claim and can it be improved?

RQ2 How much code is created through cloning?

The basic steps we follow are:

1. Implement the CMCD technique as close as practical to the original.
2. Perform two of the three evaluations described in the original paper (see section 2).
3. Based on the results of, and experience gained by, performing the previous step, refine our implementation.
4. Evaluate the effectiveness of the refinement, returning to step 3 if the results indicate the need for, or possibility of, improvement.
5. Apply the refined implementation to a large body of code, returning to step 3 if the results indicate the need for, or possibility of, improvement.

Some of these steps are elaborate further below.

There are two details we need to clarify: what definition of clone we are using and what level of granularity of clone we will detect.

As others have noted, in particular Roy et al. (2009), there is no agreed upon evaluation criteria as to when two code fragments are clones. We use the same intuition as others, namely that two code fragments are clones if one could “reasonably” have resulted by copying and pasting the other and making “minor” changes. While this introduces a degree of subjectivity, we follow Yuan and Guo and use the scenarios proposed by Roy et al., which provides some means of comparison with other work. We discuss this further in Section 6.3.

We also follow the original CMCD technique, which compares code fragments that are methods,

that is, it does not detect clones that are smaller than methods. We choose to do so as one of our goals is to replicate Yuan and Guo's study. How this technique might be applied to sub-method clones is a topic for future work.

4.1 CMCD implementation

As Yuan and Guo used Java as their target language, we choose to do the same. Their implementation determined the count matrices based on the Jimple representation of the Java source, which is a 3-address code representation produced using the SOOT framework (Vallée-Rai et al. 1999). We had a concern about this decision.

The Jimple representation is necessarily different from the original source code, and furthermore is a transformation of the compiled code (bytecode) rather than the original source code. Yuan and Guo argue that these transformations have little effect on the results. Our concern is that the two transformations may mean that slight differences in the source code may result in more significant differences in the Jimple representation. For example, information could be lost during compilation which may affect the level of accuracy, especially if optimisation techniques are used. Also, the transformation to Jimple involves introduction of temporary variables, and slight differences in the source code may result in different temporaries, potentially resulting in a more significant change at the Jimple representation than exists in the original source.

If we are right, then we would get better results dealing with the source code directly. Furthermore, if Yuan and Guo are right, it should not matter if our implementation uses a different technique to determine the count matrices.

Consequently we decided to base our implementation on a parser for Java source code. We used ANTLR (antlr.org) to create the parser. This produces an Abstract Syntax Tree (AST), which is then traversed, applying the counting conditions as appropriate to each vertex. The count matrices are created and compared as in the original.

Unlike the original technique, rather than measure similarity between methods (smaller values means less similar), we measured *differences* (smaller values means more similar).

As noted in Section 3, the meaning of the measurements can depend on the method size. The measurement for two large methods might be the same as for two small methods, which would mean the large methods are much more similar than the two small methods, but the absolute values suggest they are equally similar. Also, the values of the counts can impact the measurement. The difference between two large counts (e.g. 100 versus 90) for a given counting condition can be the same as for two small counts (11 versus 1), again indicating that the former is more similar than the latter, but just by the measurements they appear equally similar.

So some form of normalisation is needed. Unfortunately, the original paper does not describe how normalisation was performed, so we had to develop our own. We carried out a large number of trials of different forms of normalisation to find a form that have the best characteristics regarding false positives and false negatives (see below). We concluded the best normalisation was achieved by summing the values for the smaller count matrix, and dividing the raw difference measurement by that sum.

The false positive clone detection method mentioned in the original paper was also not described

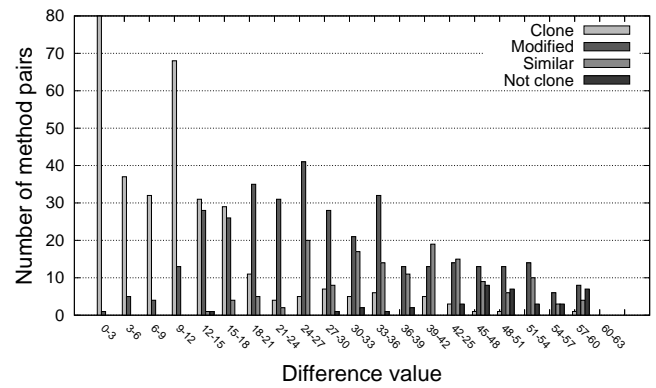


Figure 1: Showing the trade-off in candidate pair classification according to difference choices of threshold value. The 0–3 value has been truncated (from 179) for presentation purposes.

and thus we have come up with our own false positive detection method. As cloned fragments of code are similar or the same as the original fragment of code, we used a textual approach to discard clone pairs detected that had over 50% differences in text. This difference was computed after spaces and comments had been removed.

To improve performance, we classified method pairs by comparing the normalised difference between the two matrices to a predetermined threshold value. The threshold value was determined by analysing the distribution of difference values of clone pairs. This process consisted of:

1. Detecting all possible clone pairs for a selected software system and calculating the difference values.
2. Manually reviewing each method pair found and classifying it into one of the categories: clone, modified clone, similar but not clone, and not clone. (Also see below.)
3. Plotting a chart showing the distribution of different type of method pairs.
4. Determine the threshold value based on distribution.

Freecol version 0.8.0 was used for the analysis. The chart is shown in figure 1. From the chart, the default threshold value was chosen to be 45 to provide a balance between false positive and false negatives. A manual evaluation process like this was also used to evaluate different normalisation forms and choosing the level of text difference threshold.

Very small methods, such as getters and setters, are likely to look very similar. Also, even if they are created through cloning, identifying them as such is not very useful because they are so small. The original technique chose to ignore small methods for this reason, but did not specify how they identified such methods. In our implementation, the number of nodes in AST is used as the size of the method. A method is considered small if its size is less than a certain value. The number of lines of code was not used as the size of the method because it did not reflect the complexity of the code fragments and it may vary significantly depending on the coding style. By

looking at small methods such as getters and setters, we determined that an AST with 50 or fewer nodes could be reasonably classified as small.

Constructors are also ignored, as constructor clones are not very interesting. In addition, it is easy to get two constructors with the same variable count in a large system, and therefore they will introduce false positives.

4.2 Replication

We applied our implementation of the CMCD technique to the same 16 scenarios used by Yuan and Guo, and also to JDK 1.6.0_18. We did not have access to the student submissions and so did not replicate that part of their study.

4.3 Evaluation and Refinement

The original paper hinted that other counting conditions might be useful, so we planned from the beginning to support adding new conditions. That meant we also needed some way to select different conditions, and some way to show the results. We also needed to be able to vary various parameters, such as the choice of thresholds (see below). To support evaluation, we quickly learnt that it was important to not just see the list of candidate pairs, but to also show the contents of the pair, highlighting the differences between the two fragments. By examining candidate pairs in this way, we could then efficiently determine the accuracy of the choice of parameters by determining by manual inspection whether the candidate pair was indeed a clone pair. Finally, we needed the means to record the result of the manual inspection.

To this end, we developed a tool that can apply the foundation CMCD technique to any code base and that supports choosing different sets of counting conditions, different parameter values, reporting candidate clone pairs, highlighting the differences between a selected candidate pair, and recording the result of the manual inspection.

Candidate pairs are classified as “clone”, “modified clone”, “similar but not clone” or “not clone”. “Clone” is where the method pair is clearly identical with minor changes such as differences variable types or variable names. “Modified Clone” is the same as “clone” but allowing a few addition or deletion of statements. “Similar but not clone” is used for classifying code clones where at a glance, they have lots of similarities in terms of structure and sub fragments of code, but is modified enough to not be considered clones. “Not clones” is where the method pair is clearly not a clone. This classification data can be saved for analysis and future clone detections, so that there is no need to reclassify clones when the detection process is rerun with different input parameters.

Clone pairs can be sorted based on any of the characteristics of the pairs (such as the value of the pre-normalised difference between a pair). This aids the identification of clone patterns in our results by ordering the results to allow for easy access to groups of data, and visualisation of correlation between data types. For example, clone pairs can be sorted by clone classification and then by the difference value of the method pair to determine if there is a correlation.

The tool we developed allows us to identify false positives (candidate pairs that are not in fact clones). Identifying false negatives (clone pairs that are never offered as candidates) is more challenging, however our tool also supports this because it allows us to

```
ant-1.8.0 argouml-0.34‡ c_jdbc-2.0.2 cayenne-
3.0.1 cobertura-1.9.4.1 compiere-330 drawswf-
1.2.9 freecol-0.10.3‡ freemind-0.9.0‡ ganttproject-
2.0.9 gt2-2.7-M3 heritrix-1.14.4 hibernate-4.0.1‡
hsqldb-2.0.0 jFin_DateMath-R1.0.1 jag-6.1
javacc-5.0 jgraph-5.13.0.0‡ jgraphpad-5.10.0.2
jgrapht-0.8.1 jhotdraw-6.0.1‡ joggplayer-1.1.4s
jrat-0.6 jrefactory-2.9.19 jruby-1.5.2 jtpen-7.1
marauroa-3.8.1 maven-3.0 nakedobjects-4.0.0
nekohtml-1.9.14 poi-3.6 pooka-3.0-080505 roller-
4.0.1 sablecc-3.2 struts-2.2.1 sunflow-0.07.2
trove-2.1.0 velocity-1.6.4 wct-1.5.2 weka-3.6.6‡
xalan-2.7.1 xerces-2.10.0 xmojo-5.0.0
```

Figure 2: Systems used from Qualitas Corpus release 20120401. Systems for which multiple versions were analysed are indicated by ‡.

easily change various parameters to the technique, in particular the thresholds. The choice of thresholds affects the level of false positives and false negatives — the higher the threshold the more false positives but the fewer false negatives. If we want to determine the degree of false negatives for a given threshold t , we can set the threshold to a value l much larger than t , and examine those candidate pairs that are clones reported at level l that are not reported at level t . These pairs are then false negatives at level t .

Identifying false negatives, as well as allowing us to provide error bounds on our results, also provides support for refining the technique. By examining false negatives, we can identify new counting conditions that may have the potential to detect such cases.

4.4 Empirical Study

Our empirical study was carried out on 43 open source Java systems from the Qualitas Corpus, release 20120401 (Tempero et al. 2010). We did both a breadth (different systems) and a longitudinal (multiple versions of the same system) study. The systems we used are listed in Figure 2, with those used for the longitudinal study marked by ‡. See the Qualitas Corpus website (qualitascorpus.com) for details of the systems studied, such as which files are analysed.

5 Results

In this section, we present the results of the different parts of our study. Their interpretation and consequences will be discussed in the next section.

5.1 Replication Study

As with the original CMCD implementation, our implementation was also successful at detecting clones for all 16 of Roy et al.’s scenarios. We also ran our implementation on the JDK 1.6 update 18 and found 11,391 similar methods in 2523 clone clusters. The process used 51 minutes using a 2.7GHz Intel Core i5 CPU.

5.2 Refinement

From the results of our replication study, we identified limitations in the original CMCD implementation. About 15% of the candidate pairs identified in our results were false positives. These false positives were recognised as clones mainly due to the choice of

$$\text{normVCM} = \frac{dVCM(\text{total}(vCM1) + \text{total}(vCM2))}{\text{total}(vCM1) + \text{total}(vCM2) + \text{total}(mCM1) + \text{total}(mCM2)}$$

Figure 3: Normalising difference between variable and method count matrices for two code fragments.

```
private Element remove(Connection connection,
                        Element element) {
    String address = connection.getSocket().
        getInetAddress().getHostAddress();
    int port = Integer.parseInt(
        element.getAttribute("port"));
    metaRegister.removeServer(address, port);
    return null;
}
```

 Figure 4: Example of a method with no uses of variables according to the original counting conditions, and so has an empty count matrix (from the Freecol class `net.sf.freecol.metaserver.MetaServer`)

Table 2: The new Counting Conditions

- 11 Variable used in first level while loop
- 12 Variable used in second level while loop
- 13 Variable used in third level while loop (or deeper)
- 14 Variable used in first level for loop
- 15 Variable used in second level for loop
- 16 Variable used in third level for loop (or deeper)
- 17 Variable used in switch-case statement
- 18 Method invoked
- 19 Method used in if-statement
- 20 Variable invoked on

counting conditions. In Yuan and Guo’s paper, the 13 counting conditions described were not sufficient to handle all the cases. For example, switch-case statements were ignored because the counts of variables did not reflect the existence of a switch-case statement.

Another issue was that code fragments that contained only method invocations had empty count matrices, despite potentially having non-trivial code. Figure 4 shows a small method with this property, but we saw a number of larger examples of this.

This lead us to change 3 existing conditions (11, 12, and 13 in Table 1) to represent the use of variables in loops at a more fined-grained manner, and added other conditions, including for method invocation. The new conditions are listed in Table 2.

As well as new counting conditions for variables, we also apply the same counting conditions to methods in a separate method count matrix. This matrix is normalised in the same manner as described for the existing (variable) count matrix as described in Section 4.1. With the two count matrices, the comparison of code fragments is done by determining the difference between the variable count matrices for each fragment and the method count matrices for each fragment. This again raises the issue that the respective sizes of the matrices could confound the result. For example, if the two variable count matrices are the same, but the method count matrices are different, then the size of the method count matrices might affect the result. Each pair of matrices is normalised and then the two normalised values are added together.

The normalisation function is shown in Figure 3. In that figure, *vCM1* and *vCM2* are the variable count matrices for code fragments 1 and 2 respectively, and *mCM1* and *mCM2* are the method count matrices; *total*(*) returns the sum of all values in the count matrix parameter; *dVCM* is the difference for the variable count matrices using the procedure described in Section 4.1; *normVCM* is the difference between the variable count matrices normalised with respect to the method count matrices.

5.3 RQ2: Empirical Study

We used our implementation on the systems listed in Figure 2 and the multiple versions of those systems indicated. In all, there were 310 different versions, involving 210,392 files and 26,702,561 non-comment non-blank lines of code (NCLOC). The total time taken was approximately 26 hours.

The results of the empirical study are summarised in Figure 5. The systems are ordered according to the number of methods we compared in each system (that is, ignoring “small” methods and constructors) in order to see if there are any trends due to some notion of system size. In fact were the systems ordered according to lines of code, the order would not be very different.

The figure shows three values: the *total* cloned code, that is, the percentage of the code (determined by non-comment non-blank lines of code — NCLOC) that appears in a clone cluster. The light grey shows the proportion of code that is *cloned* not counting the “original” that was cloned and the dark grey is the size of the *original* code that was cloned.

We show the total because we believe that that is what other studies report (although this is generally not stated) and we want to compare with them. However, we also believe that it is worthwhile seeing the size of the code that is cloned. If two systems have (for example) 10% total cloned code, but in one the original is 1% and the other it is 5%, then this difference is worth noting. Note that we do not really know which method was the original and which was cloned, but as they are all similar we can pick one as a representative (hence the use of quotes above).

The ranges of the values are: total 6.5% (*nekohtml*) – 59.5% (*cobertura*) with an average of 17% (*sunflow*), original 2.3% (*jrat*) – 11.8% (*cobertura*) with an average of 5.3% (*poi*), and cloned 3.8% (*nekohtml*) – 47.8% (*cobertura*) with an average of 11.7% (*pooka*). The medians are: total — 14.6%, original — 5.3%, and cloned — 10.0%, all by the system *poi*.

While there seems to be a slight trend of increasing cloned code with system size, the largest system (*gt2*) has 446,863 NCLOC and 13,174 methods, which is much bigger than the second largest, *jruby*, with 160,360 NCLOC and 7646 methods, and yet the amount of cloned code is less than many other systems (but see Section 6.2).

We examined the outliers, and found that a large proportion of generated code were included in these systems. Due to the nature of generated code, they could be similar or identical and therefore recognised

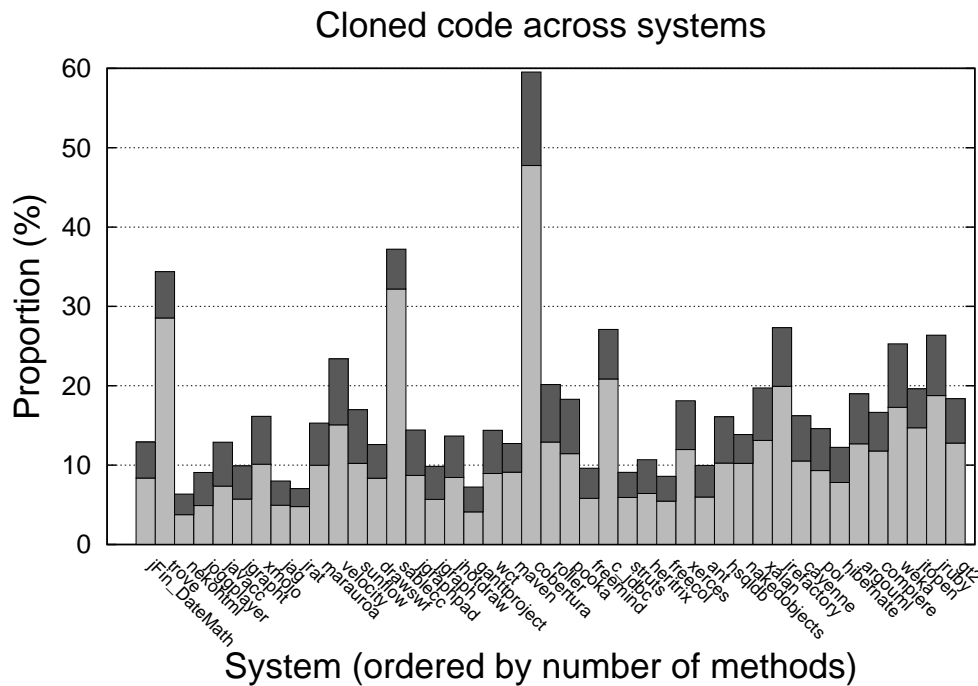


Figure 5: Proportion (%) of cloned code across the 43 systems in the study. The combined height of a bar is the proportion of code that appears in a clone cluster. The height of the dark grey bars shows the size of the “original” code that has been cloned.

as code clones by the clone detector. With the 3 top outliers (**trove**, **sablecc**, and **cobertura**) removed, the largest values are: total cloned 27.3%, original 8.3%, cloned 20.9%.

The process we used to determine the threshold value (the data is shown in Figure 1) also provides us with the means to estimate our false positive and false negative rates. For this study, we used a threshold value of 45. Those candidate clones with a difference value below the threshold (and thus reported by our tool as clones) that we manually classified as “similar” or “not clone” were classified as false positives, and those with a difference value above the threshold (that is, reported as not clones) but classified as “clone” or “modified clone” were classified as false negatives. Based on this, we had a false positive rate of 14.8% and false negative rate of 6.7%. We do note that all of the false positive method pairs found contained structurally similar code.

In addition to detecting clones in the latest version of the software systems in the Qualitas Corpus, different versions of software systems were also analysed. The results are shown in Figure 6.

6 Discussion

6.1 RQ1: Replication

The CPU time used between original implementation and our implementation was of the same order (our hardware is somewhat faster). However, the number of clones found was significantly different. Some of the clone pairs we detected were manually reviewed to assess the correctness of the result. A large proportion the clone pairs we found were the result of generated code. These generated code fragments were very similar to each other and therefore detected as code clones. We suspect that these methods were not considered in the original paper.

Yuan and Guo indicated that their implementation had a very low false positive rate, but did not provide any information on the false negative rate. Often there is a trade-off between false positives and false negatives, and so it is possible that their false negative rate was quite high. Since we had to develop our own normalisation and false positive elimination steps, it is possible that our false negative rate is not as high as the original. This might also explain why we found so many more candidate clone pairs.

Another possible source of variation was that it was not clear exactly which classes were examined in the original study, since Yuan and Guo analysed bytecode and we analysed source code.

While we did not get exactly the results reported by Yuan and Guo, they are close enough for us to conclude that the CMCD technique is as good as they claim. Furthermore, by manually reviewing detected clone pairs, there are clearly opportunities for improvement.

6.2 RQ2: Empirical Study

The smallest amount of cloned code we saw was 3.8% in **nekohtml** (6.5% if the original is included), which is the second smallest system (6,625 NCLOC and 185 methods) we analysed, meaning that the absolute amount of cloned code was also fairly small (421 NCLOC cloned code). Given that half of the systems we analysed (all larger than **nekohtml**) have 10% or more (14.6% if the original is included) points to non-trivial amounts of cloned code in open source Java systems. This is consistent with the findings of other studies.

Over the life-time of a system, according to Figure 6 there is again possibly a slight increasing trend over time for the systems that we have 20 or more versions for, however, as systems also grow over time, this might be further evidence of a relationship be-

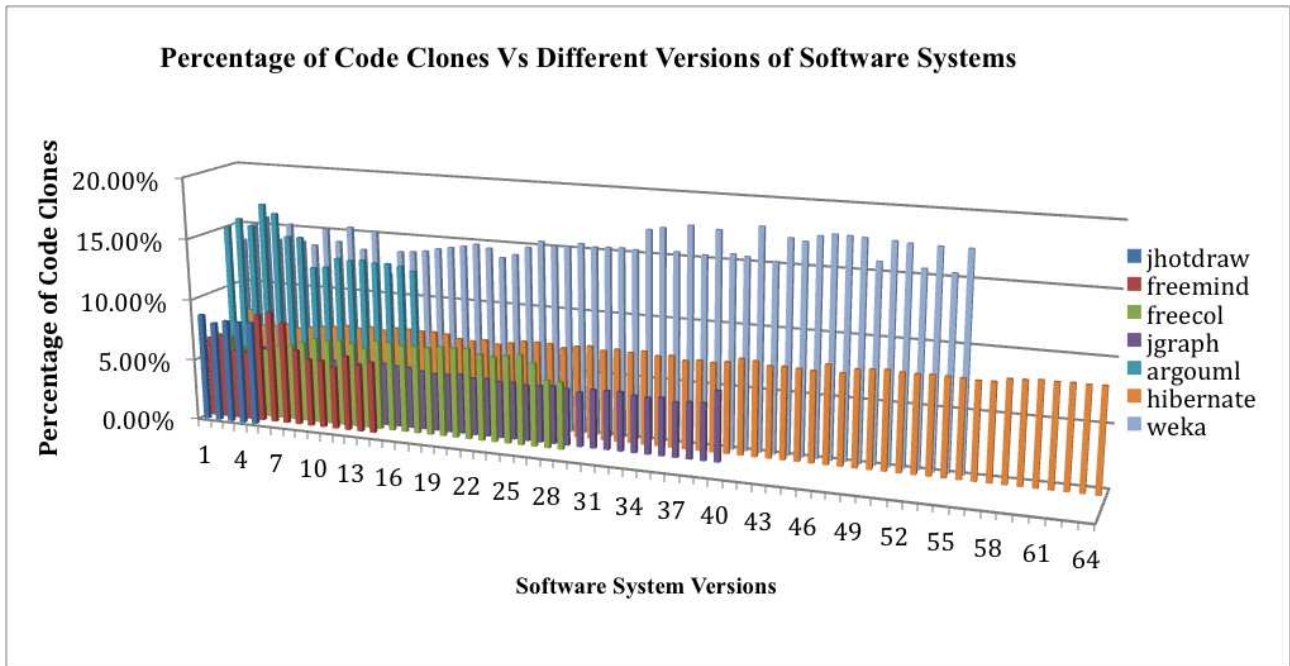


Figure 6: Study results showing the percentage of code clones across different versions of software systems

tween system size and amount of cloning. As we gather more data, we may be able to confirm this relationship.

It is worth noting that our implementation, like the original, has very good performance. We were able to analyse nearly 27 million NCLOC in about 26 hours on commodity hardware.

6.3 Threats to Validity

As with other clone detection research, a possible threat to the validity of our results is what we consider to be a clone. We have mitigated this threat by requiring that two people (the first two authors) agree on the designation (as described in Section 4) of each candidate pair.

Another possible threat is the correctness of our implementation. In particular, there is the possibility that some peculiar combination of circumstances will be mis-reported by our implementation. We have mitigated this through approximately 50 per-hours of manual review of candidate clone pairs.

One issue with comparing our results with others is that fact that we detect clones at the method level of granularity. This means that if the code in one method is completely copied to another method, but that other method also has at least as much code again added, we will not detect the use of cloning. We do not doubt that this happens, but our manual reviews found few such examples, leading us to conclude the impact on our results is small.

We have provided false positive and false negative rates for our results. These are based on our manual reviews, as supported by our tool, and so are necessarily a small subset of the total code base we analysed. While we cannot rule out missing incorrectly classified clone pairs, the nature of the CMCD technique is such that we believe our results are likely to apply generally.

Finally, we note that our results are generally in agreement with other studies, which gives us good confidence in them.

7 Conclusions

We have examined a technique for clone detection proposed by Yuan & Guo (2011) and found that generally their claims for its performance and accuracy are warranted. We have improved the original technique, in particular by adding more counting conditions and a separate method count matrix. Our improvements significantly reduce the false positives of the original. We confirmed the performance characteristics of the original study, being able to analyse nearly 27 million NCLOC in about 26 hours on commodity hardware.

We evaluate our improved CMCD through extensive manual validation supported by a visualisation tool. We replicated some of the original study and performed a large-scale empirical study of Java code. The study examined 43 systems in which we found that 1 in 2 systems had at least 10% cloned code, not counting the original. These results are broadly in agreement with other empirical studies. In particular, they are very close to a previous large study of systems written in C (Uchida et al. 2005). This suggests that the degree to which clones exist is not due to a particular language or style (procedural versus object-oriented).

In future work, we would like to improve the error bounds on the accuracy of our implementation and adapt it to work on sub-method granularity. While our empirical study is one of the largest performed for Java, it was not done on the whole of the Qualitas Corpus due to project constraints. We hope carry out an even larger study.

There is still much to be discovered about code clones. Based on our findings reported here, we believe the CMCD technique provides a very promising means to support such discovery.

References

- Baker, B. (1995), On finding duplication and near-duplication in large software systems, *in* '...', 1995., Proceedings of 2nd Working Conference on', p. 86.

- Baker, B. S. (2007), 'Finding Clones with Dup: Analysis of an Experiment', *IEEE Transactions on Software Engineering* **33**(9), 608–621.
- Baxter, I., Yahin, A., Moura, L., Sant'Anna, M. & Bier, L. (1998), Clone detection using abstract syntax trees, in 'Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272)', IEEE Comput. Soc, pp. 368–377.
- Bellon, S., Koschke, R., Antoniol, G., Krinke, J. & Merlo, E. (2007), 'Comparison and Evaluation of Clone Detection Tools', *IEEE Transactions on Software Engineering* **33**(9), 577–591.
- Cartwright, N. (1991), 'Replicability, reproducibility, and robustness: Comments on Harry Collins', *History of Political Economy*.
- Davey, N., Barson, P., Field, S. & Frank, R. (1995), 'The development of a software clone detector', *International Journal of Applied Software Technology* **3/4**(1), 219–236.
- Drummond, C. (2009), Replicability is not reproducibility: nor is it good science, in 'Evaluation Methods for Machine Learning ICML Workshop', pp. 2005–2008.
- Falke, R., Frenzel, P. & Koschke, R. (2008), 'Empirical evaluation of clone detection using syntax suffix trees', *Empirical Software Engineering* **13**(6), 601–643.
- Johnson, J. (1994), Substring matching for clone detection and change tracking, in 'Proceedings International Conference on Software Maintenance ICSM-94', IEEE Comput. Soc. Press, pp. 120–126.
- Juergens, E., Deissenboeck, F., Hummel, B. & Wagner, S. (2009), Do code clones matter?, in '31st International Conference on Software Engineering', IEEE, pp. 485–495.
- Kamiya, T., Kusumoto, S. & Inoue, K. (2002), 'CCFinder: a multilinguistic token-based code clone detection system for large scale source code', *IEEE Transactions on Software Engineering* **28**(7), 654–670.
- Komondoor, R. & Horwitz, S. (2001), Using slicing to identify duplication in source code, in '8th International Symposium on Static Analysis (SAS)', pp. 40–56.
- Kontogiannis, K. (1997), Evaluation experiments on the detection of programming patterns using software metrics, in 'Proceedings of the Fourth Working Conference on Reverse Engineering', IEEE Comput. Soc, pp. 44–54.
- Li, J. & Ernst, M. D. (2012), CBCD: Cloned buggy code detector, in '2012 34th International Conference on Software Engineering (ICSE)', IEEE, pp. 310–320.
- Mayrand, J., Leblanc, C. & Merlo, E. (1996), Experiment on the automatic detection of function clones in a software system using metrics, in 'Proceedings of International Conference on Software Maintenance ICSM-96', IEEE, pp. 244–253.
- Popper, K. (1968), *THE LOGIC or SCIENTIFIC DISCOVERY*, Routledge.
- Roy, C. K., Cordy, J. R. & Koschke, R. (2009), 'Comparison and evaluation of code clone detection techniques and tools: A qualitative approach', *Science of Computer Programming* **74**(7), 470–495.
- Schwarz, N., Lungu, M. & Robbes, R. (2012), On how often code is cloned across repositories, in '2012 34th International Conference on Software Engineering New Ideas and Emerging Results Track (ICSE)', Ieee, pp. 1289–1292.
- Tempero, E., Anslow, C., Dietrich, J., Han, T., Li, J., Lumpe, M., Melton, H. & Noble, J. (2010), 'The Qualitas Corpus: A Curated Collection of Java Code for Empirical Studies', *2010 Asia Pacific Software Engineering Conference* pp. 336–345.
- Uchida, S., Monden, A., Ohsugi, N., Kamiya, T., Matsumoto, K.-I. & Kudo, H. (2005), 'Software analysis by code clones in open source software', *Journal of Computer Information Systems* **45**(3), 1–11.
- Vallée-Rai, R., Co, P. & Gagnon, E. (1999), Soot-a Java bytecode optimization framework, in 'Conference on the Centre for Advanced Studies on Collaborative Research (CASCON)', p. 13.
- Yuan, Y. & Guo, Y. (2011), CMCD: Count Matrix Based Code Clone Detection, in '2011 18th Asia-Pacific Software Engineering Conference', IEEE, pp. 250–257.

Mining Indonesian Cyber Bullying Patterns in Social Networks

Hendro Margono, Xun Yi, Gitesh K. Raikundalia

College of Engineering and Science

Victoria University

PO Box 14428, Melbourne 8001, Victoria, Australia

hendro.margono@live.vu.edu.au; Xun.yi@vu.edu.au; Gitesh.Raikundalia@vu.edu.au

Abstract

Bullying in social media such as Twitter and Facebook has been recognised as a serious issue in Indonesia. Bullying in social media is a type of human rights violation that involves other people following an initial perpetrator in sending bullying messages repeatedly and intentionally in order to cause distress and risk to the victims. Moreover, some people use Twitter for different, more innocuous, but still unpleasant, purposes such as embarrassing someone.

Our research analyses Indonesian bullying words on Twitter so as to discover Indonesian bullying patterns. It also discusses how to mine Indonesian bullying words on Twitter by using text mining techniques. Analysing Indonesian bullying words is one of the challenges in this work.

Our research has successfully identified that “bangsat” and “anjing” terms are the trend of Indonesian bullying patterns on Twitter.

This work also compares Indonesian bullying patterns in Jakarta and Surabaya. The results are quite similar. The “bangsat” and “anjing” terms usually occur on Twitter located in both cities. Finally, our research discusses how text mining could provide a solution towards analysing Indonesian bullying words patterns in Twitter messages.

Keywords: Data Mining, Text Mining, Cyber bullying, social computing.

1 Introduction

With the rapid growth of the Information and Communication Technology, the number of people interacting using modern technologies such as the Internet has been increased significantly in Indonesia. Over the last decade, the Internet in Indonesia has developed rapidly to become a vital medium of communication in both personal and professional lives (Hui 2010). This is a new online concept and associated technologies will change our perspective on using Internet media to interact with each other in communities.

Telkom is one of the telecommunication companies in Indonesia providing internet services, Telkom states that

more than 40 million Indonesian people accessed the internet in 2011 and this will continue to increase rapidly to more than 70% in 2014 (Telkom 2012). Telkom also indicates that more than 63.1 million people used the internet for interaction in 2012, especially using social media such as Facebook and Twitter. Around 43.6 million Indonesian people used Facebook and 19.5 million used Twitter as media to communicate. This indicates that Indonesia has a potential growth of social media usage as media communication to share information..

Considering the fact that the internet and the number of social media users will grow rapidly in the next decade in Indonesia, social media is considered as a potential medium of cyber bullying. This is because social media, such as Twitter and Facebook provide some options for people to build their network, allowing them to chat or interact freely. At this point, everybody can express their ideas spontaneously to fulfil their need for existence, actualization, and socialization transferred in words, pictures and videos. Unfortunately, over time, the comfort of existence, actualization and socialization to build information and communication has been misused by people. They have used the social media to bully someone by sending offensive words, pictures or videos, which is called cyber bullying.

Cyber bullying is a kind of human rights violation to hurt or embarrass someone through ICT such as the Internet, mobile phone or other technology (Commission 2013) Furthermore, Hosking (2013) reported young Australians have been bullied at least fortnightly. The purpose of cyber bullying usually involves intimidating someone through sending text messages, emails, phone calls, chatting, and videos or pictures that are often seen as anonymous.

Bullying face-to-face is common in life. Bullying through the internet has increased in recent years and has become widespread in the world. When we look at the data from some surveys on the Internet, more than 77% students were bullied in 2012 (Graphs.net 2012). According to the Graphs.net’s survey, bullying on the Internet will grow significantly if the parents are not proactive in guiding their child when their child accesses social media on the Internet.

The Ipsos is an independent market research company, which conducted a survey to rank bullying in some countries (Gottfried 2012). They examined bullying among approximately 200,000 school-age children in 40 countries in the world in 2005-2006 and reported that Indonesia is one of the countries that has a high percentage of cyber bullying (Gottfried 2012). Ipsos reported that more than 91% of Indonesian citizens are

aware that their children were bullied on social media (Gottfried 2012). In Australia, it is about 87%, Poland is around 83%, Sweden is about 85%, the United States is about 82%, and Germany is around 81% (Gottfried 2012). Moreover, the majority of citizens in each of the 24 countries agreed that cyber bullying needs special attention from parents, particularly Japan (91%) and Indonesia (89%) (Gottfried 2012).

Kaman (2007) has conducted a survey about cyber bullying across 40 countries including Indonesia in 2005-2006. The result is that Indonesia took third place after Japan and South Korea. This indicates that cyber bullying in Indonesia is a major problem needing urgent attention.

Given the rapid growth of cyber bullying in Indonesia, identifying bullying patterns in social media is an important research task to understand what kind of bullying patterns occur. Hence, detecting Indonesian bullying patterns in Twitter should be analysed more deeply to know the strong relationship between Indonesian bullying words. This study will contribute to knowledge about human rights violations in social media in Indonesia.

Analysing Indonesian bullying words in Twitter is an interesting issue to explore. The Indonesian bullying words are unique compared to other countries, because they are always related to animals, psychology, disability, and attitude. For example, “*kamu gila, perilaku kamu seperti anjing, bangsat*” (“You are crazy, you act like a dog, you rascal”). Other than that, there are several bullying words combinations – “*bangsat, anjing, gila*” (rascal, dog, crazy). So, the perpetrator technically uses two or more words to bully.

Even though some previous research has been conducted on bullying studies, such as to explore the variety of characteristics of bullying in social media, e.g., (Campbell 2005), identifying the relationship between Indonesian bullying words using text mining technique has not been investigated as far as we know. Hence, this research has proposed to analyse Indonesian bullying words in social media. This research uses text mining techniques as a tool to extract the words and to mine data in databases.

Bullying in social media may influence people to commit harassment and violence, whether physical or psychological (Kowalski, Limber & Agatston 2008): for example, when the perpetrator arranges to meet with the victims and they possibly accomplish violence acts towards the victims. There have been a few cyber bullying cases in Indonesia that have influenced physical violence. These cases are located in Yogyakarta in which mostly young high school females are the victims. Bantul, Gunung Kidul, Kulonprogo, Sleman, Klaten, Magelang, and Purworejo in Central Java also have a similar cyber bullying phenomenon (Octaviany & Waskita 2012). The advantage of learning Indonesian bullying words is to recognise and to identify the words when they appear in social media. Then, we may be able to seek earlier help and prevent any violence from taking place.

In Indonesia, cyber bullying occurs continuously, which includes harassment, denigration, impersonation, invasion

of privacy, threats and exclusion - being excluded from the society. Besides, law enforcement in Indonesia is still weak. So, the victims rarely report such incidents to the authorities, such as the police or their parents.

This work uses Rapid Miner¹ software in order to analyse the Indonesian bullying words from Twitter. There are several processes before analysing the Indonesian bullying words: first, importing data from the repository in Rapid Miner; second, filtering words to clean up unstructure sentences; third, using FP-Growth and association rules techniques.

The data is collected using Twitter adder². Moreover, in the process of filtering words, we created a stem Indonesian bullying dictionary to detect some bullying words in Indonesian Twitter posts. Jakarta and Surabaya are two big cities in Indonesia which have become objects in this research because some people who sent bullying messages in Twitter come from Jakarta and Surabaya. Our research has identified Indonesian bullying patterns in both cities. We found the trend of Indonesian bullying patterns in Jakarta and Surabaya to be quite similar.

Additionally, the results of this research are an important contribution to the Indonesian government, Non-Government Organizations and Indonesian society about human rights violations through the Internet.

This paper is organized as follows. Section 2 describes the related work of this paper. In section 3, we describe FP-Growth and Association Rule mining which will be used to analyse the research problem. In section 4, implementation of text mining, FP-Growth, and Association Rule to solve the research problems will be detailed. Analysing the relationship among words will discover new bullying patterns. The last section is the Conclusion section.

2 Related Work

Some previous research has discussed cyber bullying in social media. Chen et al. (2012) have conducted research to detect offensive language in social media. The technique that has been used to identify offensive language is the Lexical Syntactic Feature (LSF) approach. The LSF framework has been successful in detecting some offensive content in social media, which has achieved precision of 98.24%, and recall of 94.34% and also succeeds in detecting users who sent offensive messages, achieving precession of 77.9%, and recall of 77.8% (Chen et al. 2012). A similar technique to detect offensive content in social media is a lexicon-based approach which can block and filter the offensive words and sentences in social media (Popescu & Etzioni 2007; Taboada et al. 2011). The lexicon-based technique has been successful in detecting sentiment terms in social media.

Identifying context words in Twitter using the machine learning technique has been shown in previous research. Pak and Paroubek (2010) constructed a simple binary

¹ Rapid Miner software is available at <http://rapid-i.com/content/view/26/201/>

² Twitter adder is available at <http://www.tweetadder.com/download>

classifier using n-gram and POS (Part-of-Speech) features to classify tweets on the basis of positive, negative and neutral sentiment. Their approach has similar techniques to the unigram, bigrams and POS tags approach introduced by Go, Bhayani and Huang (2009). Go, Bhayani and Huang have analysed the distribution of certain POS tags between positive and negative posts.

Recently, Maynard, Bontcheva and Rout (2012) conducted research to detect negative opinions in social media using the rule-based approach. Their research has identified negative opinions, which contain sentiment sentences, with 86% precision and 71% recall, and also identified sentences where the accuracy of the polarity (positive or negative) was 66%.

Another approach to detecting cyber bullying is a language-based method introduced by Reynolds, Kontostathis and Edwards (2011). Their research has successfully identified 78.5% of message posts from “Form spring” which contain cyber bullying by recording the percentage of curse and insult word posts.

3 Collecting Indonesian Bullying Words from Indonesian Twitter Posts

The collected data are the important part in conducting research. Much of the work required to collect data from Twitter will be spent obtaining and preparing the data. Choosing the proper approach will depend on the ultimate purpose of the analysis. There is Twitter Adder software development which can capture text in social media. The reader should be aware that these examples of Indonesian bullying words may contain offensive and often abusive language.

3.1 Data Collection from Indonesian Twitter

The first step in collecting data is download tweets text in Twitter using Twitter Adder. This step generally needs a key word to capture posting tweet text from Twitter automatically. The classification of Indonesian bullying words which were posted on Twitter will be represented in Table 1.

In our research, to collect data we used the Indonesian language because we would like to mine Indonesian bullying words which occur in Twitter. Furthermore, bullying on the Internet is also a part of human rights violations. To understand what Indonesian bullying words mean, we try to translate some Indonesian bullying words into English in Table 1.

We accessed Twitter’s public timeline to search tweets containing Indonesian bullying words to suit our interest. We removed all facts that did not express Indonesian bullying messages like news and objective phrases from collected data.

Bullying words	Indonesia	English
Bullying words related to animals	- Bangsat - Anjing - Babi - Monyet - Konyuk	- Rascal - Dog - Pig - Monkey - Monkey or stupid
Bullying words related to stupidity and Psychology	- Goblok - Idiot - Geblek - Gila - Tolol - Sarap - Udik - Kampungan	- Stupid - Idiot - Fool - Mental disorder - Stupid - Crazy - Rube - Hick
Bullying words related to disabled persons	- Buta - Budek - Jelek	- Blind - Deaf - Ugly
General Bullying words	- Setan - Iblis - Keparat - Gembel - Brengsek - Sompret - Bajingan	- Satan - Devil - Assh..* - Poor - Bastard - Jeepers - Scoundrel / Bastard
Bullying words related to attitude	- Bejad	- Depraved action

Table 1: The Classification of Indonesian Bullying Words

Overall, the total number of comments downloaded are around 14000 tweets, which specifically contain Indonesian bullying words, and the whole tweet messages that had been analysed. These data were recorded for about a week. The data are saved in Microsoft Excel format and will be recorded in Table 2.

After recording data in Table 2, the next step is preparation of data to be analysed using data mining techniques. This work will analyse posting messages on Twitter. From Table 2, the object to be analysed is the last tweet attribute.

Tweet ID	Location	Follower	Friend	Last Tweet	Last tweet date
36409 4716	Bandung	302	222	Avanya andaiy, avanya mohamad DFDM: "sarap (crazy)!! addnan_ch: Oh anjing (dog) goblog (Stupid) fakyyu siah monyet (monkey), setan (Satan), babi (pig) alas bangsat (rascal) shit damn aaaaahhhh an	4/06/20 13 14:35
50039 8381	Bekasi	148	179	baju lu beresin gak anjing lu, bangsat, monyet, ta i, sialan, gua bilangin mama. gini nihh lagi emosi, semua gua lempar ke ade	4/06/20 13 14:17

Table 2: Example Data in Excel Format

The data from Table 2 will be transformed into data repositories in Rapid Miner software. After transforming the data, we change the type of attribute of the last tweet into text because all attributes have been changed to be polynomial automatically in Rapid Miner.

3.2 Processing Data Using Rapid Miner

The next step is the preparation of the processing document. First, we design a processing document from the retrieved data to be numeric data. The purpose of designing the process document is to navigate processing data through connecting one process to other processes. The design process involves process document operator generating data from repositories using Rapid Miner Software. Rapid Miner has services for data manipulation, calculation and graphical display. Moreover, Rapid Miner also offers a broad range of statistical methods. Rapid Miner allows users to transform the text into a structured representation. Rapid Miner also has functions for managing text documents, manipulating documents and heterogeneous text format. Figure 1 shows the design of processing data from the retrieved data to be words matrix data.

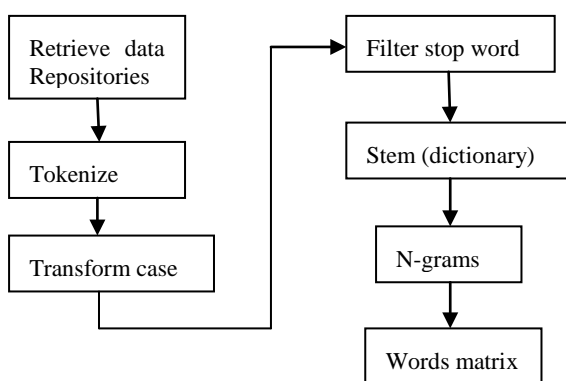


Figure 1. Processing data in Rapid Miner

We set the process document operator by creating word vectors and selecting attributes that will be analysed. We design some programs to filter the text. In this case, we create retrieving data, process document and numerical-to-binominal operators. Therefore, we need to convert all the numerical values to binominal values in order to be scanned at the next step. Every operator connects to each other.

Table 3 shows the example of a processing document using some filters such as tokenize, transform case, and stop word stem dictionary and n-grams. The function of n-grams is to detect two or more bullying words within a message. If a message consists of more than two bullying words, it is detected to be a bullying message. By selecting specific words within a message, the n-gram searches whether or not the event involves both users in applying offensive phrases in their message.

As we can see in table 3, the process cleans up some terms which do not have meaning.

The processing document operator involves some process that is as follows:

Operator	Before	After
Tokenize	Avanya andaiy, avanyaâ™RT mohamadDFDM: "sarap!! addnan_ch: Oh anjing goblog fakyu siah monyet setan babi alas bangsat shit damn aaaaahhhh an. (What are you ¤RT mohaddDFDM crazy. Oh, He is dog, stupid, fuc.*, monkey, satan, pig as rascal, shit.*)	Avanya andaiy RT mohamad DFDM sarap addnan ch Oh anjing, goblog fakyu siah monyet, setan, babi alas bangsat shit damn (What are you RT mohaddDFDM addnan ch crazy. Oh, He is dog, stupid, fuc.*, monkey, satan, pig as rascal, shit.*)
Transforming Case	Avanya andaiy RT mohamad DFDM sarap addnan ch Oh anjing goblog fakyu siah monyet setan babi alas bangsat shit damn. (What are you RT mohaddDFDM addnan ch crazy. Oh, He is dog, stupid, fuc.*, monkey, satan, pig as rascal, shit.*)	Avanya andaiy RT mohamad DFDM sarap addnan ch Oh anjing goblog fakyu siah monyet setan babi alas bangsat shit damn (What are you rt mohadddfdm addnan ch crazy. oh, he is a dog, stupid, fuc. *, monkey, satan, pig as rascal, shit.*)
Filter Stop Word	avanya andaiy rt mohadddfdm sarap addnan ch oh anjing goblog fakyu siah monyet setan babi alas bangsat shit damn (What are you rt mohadddfdm addnan ch crazy. Oh, he is dog, stupid, fuc.*, monkey, satan, pig as rascal, shit.*)	avanya andaiy mohadd sarap addnan anjing goblog fakyu siah monyet setan babi alas bangsat shit damn (What are you mohadd addnan crazy. he is dog, stupid, fuc.*, monkey, satan, pig as rascal, shit.*)
Stem Dictionary	avanya andaiy mohadd sarap addnan anjing goblog fakyu siah monyet setan babi alas bangsat shit damn (What are you mohadd addnan crazy. he is dog, stupid, fuc.*, monkey, satan, pig as rascal, shit.*)	apa kamu apa mohadddfdm sarap ada anjing goblok jancuk siah monyet setan babi ala bangsat shit damn (What are you mohadd crazy. he is dog, stupid, fuc.*, monkey, satan, pig as rascal, shit.*)
n-Grams (terms)	apa kamu apa mohadddfdm sarap ada anjing goblok fakyu siah monyet setan babi ala bangsat shit damn (What are you mohadd crazy. he is dog, stupid, fuc.*, monkey, satan, pig as rascal, shit.*)	apa apa_kamu kamu kamu_apa apa apa_mohadddfdm mohadddfdm mohadddfdm_sarap sarap_sarap_ada ada ada_anjing anjing anjing_goblok goblok goblok_jancuk jancuk jancuk_siah siah siah_monyet monyet monyet_setan setan setan_babi babi babi_ala ala ala_bangsat bangsat bangsat_shit shit shit_damn damn (What are you mohadd crazy. he is dog, stupid, fuc.*, monkey, satan, pig as rascal, shit.*)

Table 3: Example Filtering Document

Table 3 illustrates the example of before and after cleaning up tweets and the result is that some unstructured Indonesian bullying words have been cleaned. The example of the clean message can be seen in table 3 in the column after the stem dictionary.

3.3 Stem Indonesian Bullying Dictionary

Figure 2 describes the stem dictionary that we developed to clean some Indonesian bullying terms when people sent tweets which may have the same meaning as other terms. For example: the bngt has the same terms and meaning as bangsat.

The stem Indonesian bullying dictionary has the purpose to filter some words which may have the same meaning as Indonesian bullying words. The stem Indonesian bullying dictionary process will transform some unstructured Indonesian bullying words into more structured words because people sent tweets by typing incomplete words. For example, *bngt* and *anj* will be replaced to become *bangsat* and *anjing*.

To replace some unstructured words in tweets, we create an Indonesian bullying dictionary by typing some Indonesian bullying words in Microsoft Word and save it in text format. For example: anjing:anj.*; bangsat:bngt.*. Typing anjing:anj.* means that all words by typing anj or anji will be replaced to be anjing.

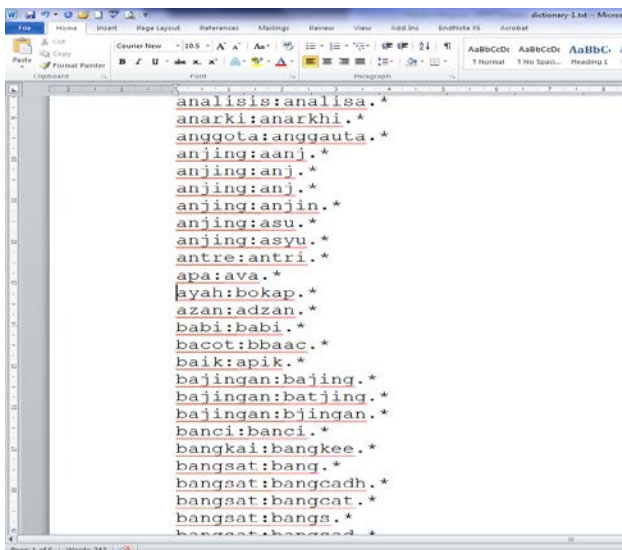


Figure 2: Creating Stem Indonesian Bullying Dictionary

Figure 2 shows the stem Indonesian bullying dictionary in text format. The stem dictionary has the function of replacing unstructured Indonesian bullying words with structured words. The resulting stems Indonesian bullying dictionary is shown in Table 3.

The result of the processing document operator is data matrix which is shown in Appendix 1. The data show some Indonesian bullying words which occur in some tweets. For example, “bangsat” (rascal) term occurs in tweets 1,2,3,..., etc.. The 0 means the word not appearing in the tweets, and 1,2,3 mean how many times the word appears in the tweets. Appendix 1 shows the data matrix after generating some operators in the processing document.

The next operator in the processing document is numerical-to-binominal. If the value of an attribute is between the specified minimal and maximal values, it is false, otherwise true. The result of generating this operator is shown in appendix 2.

Data from the matrix will be analysed using data FP-Growth and Association Rule. The data will change into letters representing the real data. The reason to change the real data to letters is that the computation of data mining will read data in letters or numbers. Table 3 describes representing the real data as letters or numbers.

3.4 Indonesian Bullying Words in Data Set

In association rule mining, the data which come from tweets will be represented as alphabet in the data set. The purpose of representing Indonesian bullying words in the alphabet is to calculate how often the words occur in the database and to compute how strong the relationship is between words. Representing data in a data set of association rules will be described below.

In association rule *TID* is a set of transactions. In this process, *TID* represents a set of all tweets posts in a database of Indonesian cyber bullying words. $I = (a,b,d,c,d,e,...,z)$ is a set of items which contain Indonesian bullying words. Let A be a set of items of Indonesian bullying words. When $A \subseteq T$, an implication from $A \Rightarrow B$ can be called an association rule, where $A \subseteq I$, $B \subseteq I$, and $A \cap B \neq \emptyset$. In order to obtain a strong relationship between items in an association rule, all itemsets should satisfy the minimum support threshold and minimum confidence threshold as prerequisites in association rule mining. For calculating minimum support and minimum confidence, this work uses the formula given by Han et al. (2012).

Support $(A \Rightarrow B) = P(A \cup B)$

Confidence

$$(A \Rightarrow B) = P(B/A) = \frac{(\text{support}(A \cup B))}{(\text{support}(A))} = \frac{(\text{support_count}(A \cup B))}{(\text{support_count}(A))}$$

Table 4 shows an example of representing bullying words by letter.

Item	Represent
a	Bangsat (rascal)
b	Anjing (dog)
c	Babi (pig)
d	Monyet (monkey)
e	Kunyuk (monkey)
f	Baiingan (scoundrel/bastrad)

Table 4: Illustrate Set of Item in Data Set

Table 4 shows the illustrated Indonesian bullying words represented in an alphabet. For example: the letter a represents “bangsat”, b represents “anjing”, etc. The concept association rule is market basket analysis; hence all Indonesian words should be represented in the alphabet. The purpose of representing Indonesian bullying words with the alphabet is to get itemsets patterns in the Indonesian Bullying database.

Tweet ID	Last Tweet
1	a, b, c, d, g, n
2	a, b, d
3	a, b, k
4	a, b, c, d
5	a, b, m, t

Table 5: Example Data in Data Set

Table 5 shows some example data recorded in the database. This means the itemsets that occur in transactional data are tweets which contain bangsat (a), anjing (b), babi (c), monyet (d), goblok (g), and sarap (n).

4 Mining Indonesian Bullying Patterns on Indonesian Twitter Post

Data preparation, collection data from Twitter are the main work in this research. The purpose of analysing message from Twitter is to identify Indonesian bullying words patterns and trends. To achieve this aim, we will make an effort to identify pattern words using text mining technique.

The process of mining Indonesian bullying patterns in Twitter can be seen in figure 3.

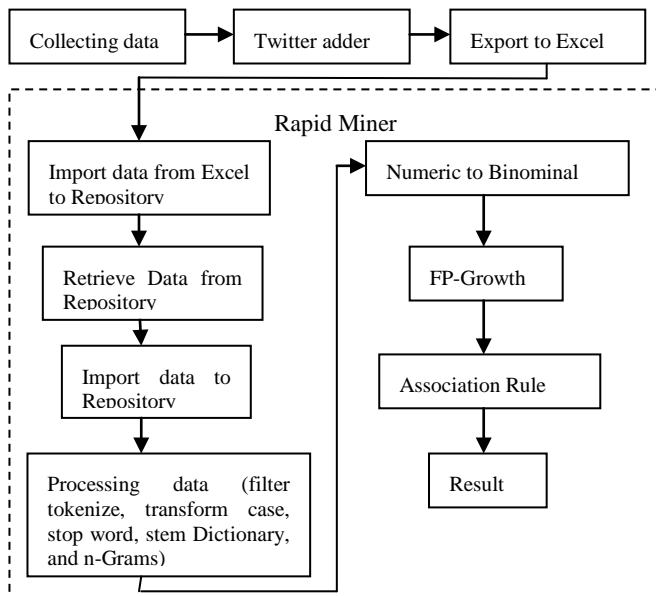


Figure 3. The process analysing Indonesian bullying words from Indonesian Twitter

This research will use two steps to analyse bullying which occurs on Twitter.

1. First step, we will conduct analysis tweets using FP-growth, in terms of finding frequent patterns itemsets.
2. Second step, we use association rules mining to analyse the strong relationship between a pair of words.

In order to obtain a strong relationship between items, the minimum support threshold is 0.05 and the minimum confidence threshold is 0.95.

4.1 Mining Indonesian Bullying Pattern Using FP-Growth in Rapid Miner

In this section, this work uses FP-Growth to analyse frequent patterns itemsets without generating candidate itemsets (Han, Kamber & Pei 2012). We believe this method to be more efficient compared to Apriori algorithm. The FP - Growth algorithm is an alternative algorithm in data mining used to find frequent itemsets. This approach uses FP-Tree algorithm which encodes the data set into a tree and then extracts the frequent itemsets from this tree.

The construction FP-Tree is divided into several steps.

1. To get support count for each item, scan the data set firstly to find frequent itemsets. Then, abandon the items which are not frequent and then sort the frequent items in decreasing order.
2. To create FP-Tree, scan every transaction from the data set. Every transaction will be read as follows:
 - a. Create a new path if the transaction is a unique transaction form and set the counter for each node to be 1
 - b. Add the common itemsets node counters if the transaction shares a common prefix itemsets then and create new nodes if needed.
3. Continue from first to second steps until each transaction has been mapped onto the tree.

Using the Rapid miner software, the data set will be generated and calculated automatically based on FP-Growth method. After being generated automatically, the frequent itemsets have been found, shown in table 6.

Support Count	Item1	Item 2	Item 3
0.330	bangsat (rascal)		
0.323	anjing (dog)		
0.070	anjing_bangsatsat		
0.267	bangsat	anjing	
0.054	bangsat	babi (pig)	
0.070	bangsat	anjing_bangsatsat	
0.081	anjing	babi	
0.070	anjing	anjing_bangsatsat	
0.052	bangsat	anjing	babi
0.070	bangsat	anjing	anjing_bangsatsat

Table 6: The Result after generating FP-Growth in Rapid Miner

Table 6 describes generated support count in FP-growth. There are frequent itemsets (Indonesian bullying words) which occurred in FP-growth: “bangsat” (rascal) and “anjing” (dog) were dominant in frequent itemsets. This means most Indonesian tweeters use both “bangsat” and “anjing” words to bully someone on Twitter. Moreover, another word such as “bangsat and babi” (pig) also has support count at around 0.054. It means that some Indonesian tweeters use “bangsat” and “babi” words to push other tweeters.

4.2 Mining Indonesian Bullying Pattern Using Association Rule in Rapid Miner

The first method that will be applied in this research is association Rule which is based on market basket analysis. This method is usually applied in supermarkets, grocery stores, or book stores. The purpose of association Rule is to identify the trend items which tend to be purchased together in supermarkets, grocery stores, or book stores. The benefit when the shops have applied this method is receiving information on the kind of items that tend to be purchased together. Moreover, the shops can optimise the layout of items in the store which can potentially increase sales by cross-selling items. When

the association rule is applied to mine text, the text field would become the transaction and the words themselves would become the items.

Using the Rapid miner software, the transaction of itemsets is generated and calculated automatically and the result is shown in table 7. Table 7 shows that the frequent itemsets from transaction in database have been found. “Bangsat” and “anjing” are dominant in frequent itemsets. Meanwhile, “iblis” (devil) and “setan” (satan) have taken the second place after both “bangsat” and “anjing” words. “Bangsat” and “babi” have support count 0.052 and confidence 0.970 with imply “anjing” words. That means that “bangsat” and “babi” words have strong relation with “anjing” words. Furthermore, “iblis” word has support count 0.056 and confidence 0.986 to “setan” word. This means that “iblis” also a strong relation with “setan”.

Premises	Conclusion	Support Count	Confidence
bangsat (rascal), babi (pig)	Anjing (dog)	0.05	0.97
iblis (devil)	setan (satan)	0.05	0.98
anjing_bangsatsat	bangsat	0.07	1.0
anjing_bangsatsat	anjing	0.07	1.0
anjing_bangsatsat	bangsat, anjing	0.07	1.0
bangsat, anjing_bangsatsat	anjing	0.07	1.0
anjing, anjing_bangsatsat	bangsat	0.07	1.0

Table 7: The Result after Generating Association Rule in Rapid Miner

When we are looking at the result in table 7, it can take the point of view that most Indonesian tweeter use “bangsat (rascal)”, “anjing (dog)”, “iblis (devil)”, and “setan (satan)” words to bully someone on social media especially on Twitter. The bullying occurs when people send tweets to others by sending two Indonesian bullying words such as “bangsat” and “anjing” or “iblis” and “setan”.

The reasons perpetrators were sending tweets using Indonesian bullying words on Twitter are follows:

1. They have expressed their emotion because the victims may have a bad attitude which may influence another person.
2. They have a purpose to harass or abuse someone by sending the bullying message.
3. They were following other tweeters in bullying someone.

Appendix 3 shows the relationship between Indonesian bullying words in a graph. “Bangsat and “anjing” words have strong relation with satisfying *support_count* and *confidence*. Meanwhile, “iblis (devil)” and “setan (satan)” words also have satisfied the support count and confidence. Nevertheless, “bangsat” and “anjing” do not have relation with “iblis” and “setan” words.

4.3 Comparison Indonesian Bullying Pattern between in Jakarta and Surabaya cities

Indonesia has many islands which spread from Sabang to West Papua Island, about 13.000 Islands (Maruli 2010).

Hence, this work will try a comparison between two regions in Indonesia which make a high percentage contribution to bullying in Indonesian Twitter. This work will analyse the cities of Jakarta and Surabaya because both cities are among the biggest cities of Indonesia. Many Indonesian people who come from rural areas move to Jakarta and Surabaya. Therefore, Jakarta and Surabaya have a lot of communities which tend to interact with each other.

To examine the Indonesian bullying pattern in Jakarta and Surabaya regions, both FP-Growth and Association Rule will be used. The purpose of this is to compare two locations which contribute Indonesian bullying words on Twitter. The comparison between Jakarta and Surabaya will be described clearly in this work.

The first technique to identify the Indonesian bullying pattern in Jakarta and Surabaya is FP-Growth. Using the rapid miner software, the result after generating FP-Growth algorithm is as follows:

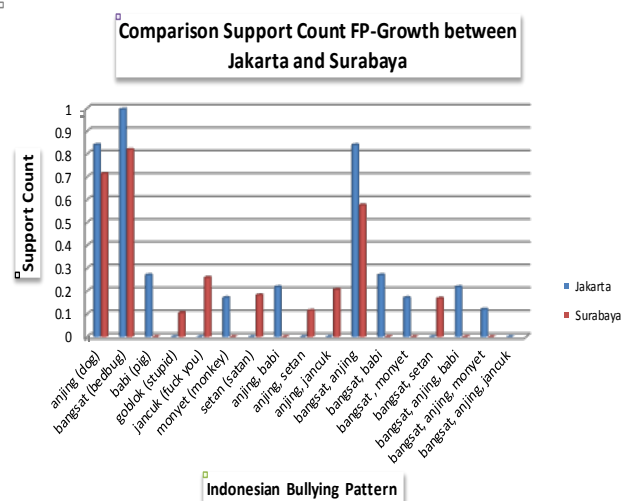


Figure 4: Comparison Support Count in FP-Growth between Jakarta and Surabaya

Figure 4 shows the comparison support_count in FP-Growth between Jakarta and Surabaya. We find the trend in Indonesian bullying patterns in Jakarta after generating it in FP-Growth. The “bangsat”, and “anjing” terms are the most popular bullying patterns which occur in Jakarta. This means most tweeters from Jakarta used “anjing” and “bangsat” terms to bully someone on Twitter. Moreover, “bangsat” has a big support count at around 0.9 and “anjing” has support count at about 0.8. When we look at “bangsat” and “anjing”, the support count is more than 0.8. This means that the frequency patterns in Jakarta are “bangsat” and “anjing”.

When we compare this to Surabaya in figure 4, we also find the trend of Indonesian bullying patterns in Surabaya after generating in FP-Growth. The “bangsat” and “anjing” are popular bullying terms in Surabaya. The frequent patterns of Indonesian bullying patterns in Surabaya are similar to Jakarta. The “bangsat” and “anjing” terms have a big support count. This means the tweeters in Surabaya use “bangsat” and “anjing” to bully someone on Twitter.

The differences in bullying patterns between tweeter in Jakarta and Surabaya are as follows:

1. There are other Indonesian bullying patterns in Jakarta such as “babi”, which has support count at around 0.25 and “monyet”, which has support count at about 0.15.
2. Others Indonesian bullying patterns in Surabaya are “goblok”, “jancuk”, and “setan”. “Goblok” has support count at about 0.1, “jancuk” has support count at around 0.25, and “setan” has support count at about 0.18.

Another technique that will be used is Association Rule. The Association rule technique will identify how strong the relationship is between Indonesian bullying words.

Premises	Conclusion	Jakarta		Surabaya	
		Support Count	Confidence	Support Count	Confidence
anjing (dog)	bangsat (rascal)	-	-	0.576	0.807
bajingan (scoundrel)	bangsat	-	-	0.102	0.922
bangsat	anjing	0.246	0.803	-	-
babi (pig)	anjing	0.218	0.808	-	-
setan (satan)	anjing	0.077	0.852	-	-
monyet	bangsat	0.171	1	-	-
setan	bangsat	0.090	1	0.168	0.926
babi	bangsat, anjing	0.218	0.808	-	-
bangsat, babi	anjing	0.218	0.808	-	-
setan	bangsat, anjing	0.077	0.852	-	-
bangsat, setan,	anjing	0.077	0.852	-	-
babi, monyet (monkey)	anjing	0.073	0.885	-	-
babi, monyet,	bangsat, anjing	0.073	0.885	-	-
anjing, babi,	bangsat	0.218	1	-	-
anjing, monyet,	bangsat	0.119	1	0.051	0.805
anjing, setan,	bangsat	-	-	0.113	0.991

Table 8: Comparison between Jakarta and Surabaya after Generating Association Rule

The results of the comparison after generating association rule between Jakarta and Surabaya are shown in table 8. In Jakarta, the combination two and three Indonesian bullying words that have great support count and confidence calculation are “monyet (monkey)” and “bangsat (rascal)”; “setan (satan)” and “bangsat”; “anjing (dog)”, “babi (pig)”, and “bangsat”; “anjing”, “monyet” and “bangsat”. This means that the strong rule of the Indonesian bullying words in Jakarta are “monyet” and “bangsat”; “setan” and “bangsat”; “anjing”, “babi”, and “bangsat”; “anjing”, “monyet” and “bangsat”.

When we compare this to Surabaya, the combination two and three Indonesian bullying words which have great support count and confidence calculation are “bajingan” and “bangsat”; “setan” and “bangsat”; “anjing”, “setan”

and “bangsat”. This means that the strong rule of the Indonesian bullying words in Surabaya is “bajingan” and “bangsat”; “setan” and “bangsat”; “anjing”, “setan” and “bangsat”.

Words with similar calculation support count and confidence for Indonesian bullying words between Jakarta and Surabaya after generating association rule are the “setan” “bangsat”; “anjing”, “monyet” and “bangsat” words. This means that the words have strong rule.

When we look at the result, the trend of Indonesian bullying words is “bangsat” and “anjing”. This trend illustrates that bullying through technology media has become a new phenomenon in modern life. People can send messages easily with anonymity to embarrass someone they do not like.

Moreover, all the trends of Indonesian bullying patterns both in Jakarta and Surabaya have important information for people who are interested in cyber bullying, especially in Indonesia. The trend of Indonesian bullying patterns also supports information for the Indonesian government to develop software for detecting offensive messages which contain Indonesian bullying words on Twitter. This result shows the real data Indonesian cyber bullying on Twitter, which can support the Indonesian government in enforcing Indonesian communication law.

Additionally, the trend of Indonesian bullying patterns on Twitter can be associated with social anxiety. Perpetrators send offensive messages on Twitter which can affect the victims whereby they feel unsafe, become unwell and anxious. In this situation, the victims are experiencing a social anxiety. The social anxiety is a potential cause of the victims feeling shyness, anxiety disorders, other emotional and temperamental factors. The victims feel discomfort and fear when they interact with other people in social media involving the perpetrators judging and evaluating them.

The social anxiety can be related to the self-esteem of the victims. The perpetrator will evaluate the victims based on the qualities they possess. The victims with low self-esteem tend to focus on their own weaknesses rather than focusing on their strengths. This will have a huge impact on their psychological well-being and their actions, thus leading to disorders like social anxiety.

The trend of Indonesian bullying patterns visualises that the perpetrators have a social attitude problem in which they always feel superior to the victims. They also invite their friends and other followers to be involved in the bullying.

5 Conclusion

In this study, we investigate the Indonesian bullying pattern which exists in Indonesian Twitter posts. Specifically, we propose the stem Indonesian dictionary to identify Indonesian bullying words in Twitter Posts, and further, to analyse the relationship between Indonesian bullying words. Our research has several contributions. First, we practically conceptualize the Indonesian lexicon bullying words and further detect Indonesian bullying word. Second, we analysed the Indonesian bullying words using Association Rule and FP-Growth to find trends and patterns of Indonesian

bullying words in Indonesian Twitter. Experimental results show that the stem Indonesian dictionary in Rapid Miner satisfies the identification of some Indonesian bullying words in Indonesian Twitter posts. Moreover, this research used Association Rule and FP-Growth techniques in Rapid Miner Software to find frequent Indonesian bullying patterns which were transformed into itemsets. The results after generating the data from repositories have shown that the trend in Indonesian bullying patterns which occurred in Indonesian Twitter posts after being generated were “bangsat”, “anjing”, “iblis”, and “setan”. Furthermore, the result of finding Indonesian bullying patterns in Jakarta and Surabaya is quite similar. Both “bangsat” and “anjing” terms are the most popular used by tweeters who live in Jakarta and Surabaya.

In addition, after generating data from the repository, both techniques, FP-Growth and Association Rule, has similar results. It means both techniques actually have powerful calculation able to find frequent itemsets which appeared in the database.

In conclusion, this research will mine deeply the Indonesian bullying words using clustering techniques to cluster and Naïve Bayes to classify some Indonesian bullying words. In the future, this research will be developed to detect and to predict what kind of Indonesian bullying patterns will occur on Twitter.

6 References

- Campbell, MA. (2005): 'Cyber bullying: An old problem in a new guise?', *Australian Journal of Guidance and Counselling*, vol. 15, no. 1, pp. 68-76.
- Chen, Y., Zhou, Y., Zhu, S., & Xu, H. (2012) : 'Detecting Offensive Language in Social Media to Protect Adolescent Online Safety', in *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Conference on Social Computing (SocialCom)*, pp. 71-80.
- Commission, AHR (2013): *Cyberbullying, Human rights and bystanders*, Australian Human Rights Commission viewed 29/4/2013 2013, <https://bullying.humanrights.gov.au/cyberbullying-human-rights-and-bystanders-0>.
- Go, A., Bhayani, R., & Huang, L. (2009): 'Twitter sentiment classification using distant supervision', *CS224N Project Report, Stanford*, pp. 1-12.
- Gottfried, K. (2012): *One in Ten (12%) Parents Online, Around the World Say Their Child Has Been Cyberbullied, 24% Say They Know of a Child Who Has Experienced Same in Their Community*, Ipsos, viewed 1/4/2013 2013, <http://www.ipsoshk.com/wp-content/uploads/2012/04/Cyberbullying-factum-AP.pdf>.
- Graphs.net. (2012): *Bullying Statistics 2012*, graphs.net, viewed 1/5/2013 2013, <http://www.graphs.net/201209/bullying-statistics-2012.html>.
- Han, J., Kamber, M., & Pei, J. (2012): *Data mining: concepts and techniques*, 3rd edn, Elsevier, Amsterdam.
- Hosking, W. (2013): 'Plug in judges: cyber bullying call to arms', *Herald Sun*, 13 July, News 17.
- Hui, JY. (2010): 'The Internet in Indonesia: development and impact of radical websites', *Studies in Conflict & Terrorism*, vol. 33, no. 2, pp. 171-91.
- Kaman, C. (2007): *What country has the most bullies?*, Latitude News, viewed 29/4/2013 2013, <http://www.latitudenews.com/story/what-country-has-the-most-bullies/>.
- Kowalski, R, Limber, S & Agatston, P. (2008): 'Cyber Bully: Bullying in the Digital Age', *Australia: Blackwell*.
- Li, Min., Sun, Xiaoxun., Wang, Hua., Zhang, Yanchun., and Zhang, Ji. (2011): *Privacy-aware access control with trust management in web service*. World Wide Web 14, 4 (July 2011), 407-430. DOI=10.1007/s11280-011-0114-8 <http://dx.doi.org/10.1007/s11280-011-0114-8>
- Maruli, Ae. (2010): 'Hasil Survey Terbaru Jumlah Pulau Indonesia', *Antara News*.
- Maynard, D, Bontcheva, K & Rout, D 2012, 'Challenges in developing opinion mining tools for social media', *Proceedings of@ NLP can u tag# user_generated_content*.
- Octaviany, K & Waskita, D. (2012): 'Fenomena Cyberbullying Facebook Pelajar Yogya: Tindakan Cyberbullying ternyata Seperti Fenomena Gunung Es', *Viva News*, 27/06/2012, Technology.
- Pak, A & Paroubek, P. (2010): 'Twitter as a Corpus for Sentiment Analysis and Opinion Mining', in *LREC*.
- Popescu, A-M & Etzioni, O. (2007): 'Extracting product features and opinions from reviews', in *Natural language processing and text mining*, Springer, pp. 9-28.
- Reynolds, K., Kontostathis, A. & Edwards, L. (2011): 'Using machine learning to detect cyberbullying', in *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, vol. 2, pp. 241-4.
- Sun, Xiaoxun., Wang, Hua., Li, Jiuyong., and Zhang, Yanchun. (2012): *Satisfying Privacy Requirements Before Data Anonymization*. Comput. J. 55, 4 (April 2012), 422-437. DOI=10.1093/comjnl/bxr028 <http://dx.doi.org/10.1093/comjnl/bxr028>
- Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011): 'Lexicon-based methods for sentiment analysis', *Computational linguistics*, vol. 37, no. 2, pp. 267-307.
- Telkom. (2012) : *Indonesia Menuju 100 Juta Pengguna Social Media*. Telkom, viewed 29/4/2013 2013, <http://www.telkomsolution.com/news/it-solution/indonesia-menuju-100-juta-pengguna-social-media>.
- Wang, H., Zhang, Y., Cao, J. (2009) : Effective collaboration with information sharing in virtual universities, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 21, No. 6, pages: 840-853, June.

Appendix 1

Data Matrix

Result Overview WordList (Process Documents from Data) ExampleSet (Process Documents from Data)

Meta Data View **Data View** Plot View Advanced Charts Annotations

ExampleSet (14877 examples, 1 special attribute, 45049 regular attributes)

Row No.	text	a	a_bangsats	a_di	a_didagoan	a_dovei	a_ga	a_ka	a_kalah	a_ko
1	lasttweet	0	0	0	0	0	0	0	0	0
2	apa apa_kar	0	0	0	0	0	0	0	0	0
3	gila gila_gila	0	0	0	0	0	0	0	0	0
4	sarap sarap_	0	0	0	0	0	0	0	0	0
5	cageur cage	0	0	0	0	0	0	0	0	0
6	bangsat ban	0	0	0	0	0	0	0	0	0
7	baju baju_k	0	0	0	0	0	0	0	0	0
8	garyavg garj	0	0	0	0	0	0	0	0	0
9	sok sok_iy i	0	0	0	0	0	0	0	0	0
10	rt rt_alvinfat	0	0	0	0	0	0	0	0	0

Appendix 2

Numeric to Binominal

Result Overview ExampleSet (Numerical to Binominal)

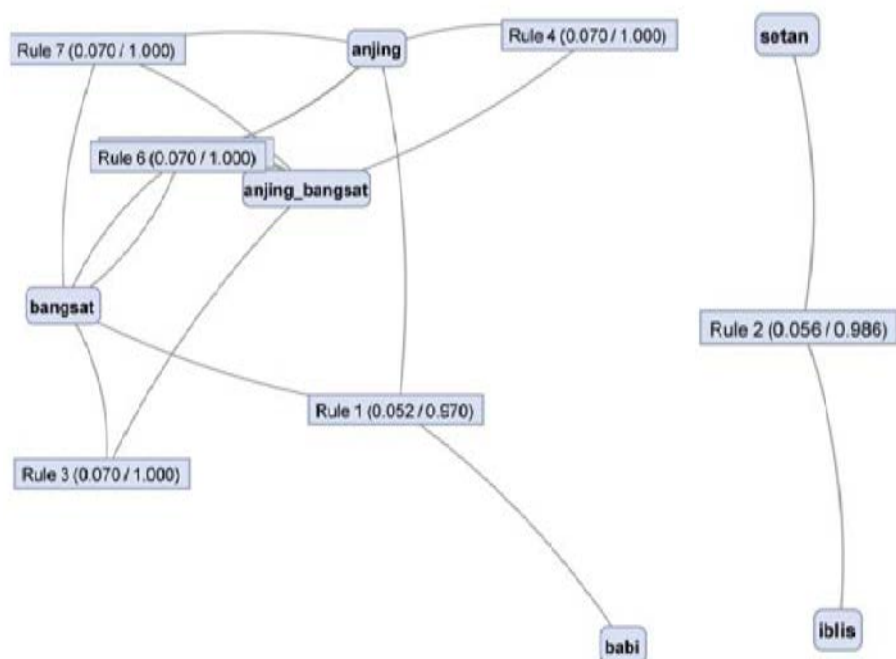
Meta Data View **Data View** Plot View Advanced Charts Annotations

ExampleSet (14877 examples, 1 special attribute, 45049 regular attributes)

Row No.	text	a	a_bangsats	a_di	a_didagoan	a_dovei	a_ga	a_ka	a_kalah	a_ko
1	lasttweet	false	false	false	false	false	false	false	false	false
2	apa apa_kar	false	false	false	false	false	false	false	false	false
3	gila gila_gila	false	false	false	false	false	false	false	false	false
4	sarap sarap_	false	false	false	false	false	false	false	false	false
5	cageur cage	false	false	false	false	false	false	false	false	false
6	bangsat ban	false	false	false	false	false	false	false	false	false
7	baju baju_k	false	false	false	false	false	false	false	false	false
8	garyavg garj	false	false	false	false	false	false	false	false	false
9	sok sok_iy i	false	false	false	false	false	false	false	false	false
10	rt rt_alvinfat	false	false	false	false	false	false	false	false	false

Appendix 3

Association Rule in Graph



Current Educational Technology Use for Digital Information Acquisition by Young New Zealand Children

Nicholas Vanderschantz, Annika Hinze, Sally Jo Cunningham

Department of Computer Science, Faculty of Computing & Mathematical Sciences
University of Waikato, Hamilton, New Zealand

{vtwoz, hinze, sallyjo}@cs.waikato.ac.nz

Abstract

Improving children's information acquisition using digital documents is an under-studied field. We performed a survey with both teachers and parents to highlight the current use of technology and digital information by young New Zealand children, both in schools and at home. We found that children have access to a range of technologies and information sources both at home and at school. They use a mix of computers, print books and eBooks to access documents. This paper analyses the results of our survey and discusses its implications for further studies and interventions on educational practices for children's information acquisition.

Keywords: ICT. Children's Technology. Mobile Information Search.

1 Introduction

Children are encountering on-screen reading and learning in both formal educational settings and in their daily recreational activities. (Cooper, 2005) describes this as the *omnipresence of technology*. We note that classroom technology includes interactive learning tools, on-line standardized testing material, digital books on CD-ROMs and eBooks, and digital reference books such as encyclopedia and dictionary. Children in New Zealand classrooms have been observed to use this full range of technologies during their typical educational pursuits. For example Timpany & Vanderschantz (2011) observed a range of technologies including digital whiteboards, laptop computers and mobile tablet and personal touch-screen interactive devices such as the iPod and iPad in a single New Zealand school in 2011.

Whilst these technologies are being introduced into today's classrooms, it is not clear from the literature how effective these technologies are in facilitating information acquisition. Interaction with these technologies, both at pre-school and primary school, has been shown to be associated with cognition development (Li and Atkins, 2004) as evidenced through results in school readiness tests (Boehm-3 Preschool) and cognition development

tests (WPPSI-R). Yet, there is still much to learn about children's information needs and the impact of technology interventions in the classroom.

Even though some research has addressed reading and learning with interactive software (Li and Atkins, 2004; Timpany and Vanderschantz, 2011), facilitating children's information acquisition in *digital documents* remains largely unexamined. Our goal with this ongoing research is to develop tools that can be immediately useful to children in New Zealand classrooms and homes. A future study in our ongoing work in this area will see the design and development of tool(s) that can be deployed immediately within at least one of these New Zealand schools that we survey here. We therefore require clear understanding of the technology that children have access to presently and how we may extrapolate with reasonable confidence in moving forward with future research.

This paper explores children's use of Information Communication Technology (ICT) as seen by teachers and parents when seeking information at home and at school in New Zealand today. We first motivate the need for our study by an extensive analysis of the literature related to children's digital information acquisition. This related work will show the lack of current literature reporting contemporary ICT usage statistics.

We compare and contrast the results of our survey from 10 suburban schools and one rural school in the Waikato. The results of our analysis lay the foundations for our research into methods of facilitating children's information acquisition with digital documents and will serve to benefit researchers in related areas with provision of concrete evidence of the use of specific ICT and information acquisition sources.

The remainder of the paper is structured as follows. We highlight in Section 2 the paucity of literature that reports how tools (hardware and software technologies) for children's information acquisition can support effective information problem solving and information literacy. Section 3 briefly describes our study methodology and reports selected results of questions asked of parents and teachers in suburban school in the Waikato. We contrast this with results from the same survey conducted in a rural school in Section 4. We discuss our conclusions from the survey in Section 5 and summarise our findings in Section 6.

2 Related Work

This section shows the relevance of our research within the field of education and computer science. In particular, we highlight gaps that require further research into

processes and tools to support children's information acquisition.

2.1 Pedagogy & Educational Theory

Mono-directional educational techniques are no longer the favoured primary educational method in schools in New Zealand and around the world. Brand-Gruwel et al (2005, p. 488) observe that students are instead "expected to construct their own knowledge, search and process information and combine it with their prior knowledge in order to tackle authentic tasks and problems". This statement describes current teaching *pedagogy*, which encompasses teaching practices as well as teaching philosophies. The fields of science, philosophy and psychology are all contributing to advancing educational practices, theories and pedagogies.

The pedagogy of the New Zealand primary curriculum is largely influenced by the educational theories of Vygotsky, Papert and Piaget. They represent a teaching philosophy that requires the students to construct their own problems and explore ways to solve those problems in an active and self-motivated manner. Victor Vygotsky is considered the father of *Activity Theory* -- a socio-cultural descriptive theory that pays attention to people and their work within their environment and culture. Seymour Papert is known for his work in *Constructionist Theory*: based on constructivist theory, constructionism espouses that students create mental models to understand the world around them, often through tangible real world objects. Papert's predecessor and mentor Jean Piaget is credited with the theories of *Constructivist Theory*, which posits that learning is an active, constructive process where students create their own learning by linking prior knowledge through construction rather than acquisition.

These theories are relevant to our research because schools following these approaches are expected to use teaching and learning that directly appeals to information problem solving skills. In the following section, we show that interaction with ICT form a significant part of the learning activities of children in New Zealand classrooms. Therefore both digital and analogue documents are likely to be assessed on a daily basis, therefore supporting the need for our ongoing investigations; the need for further investigation of digital information acquisition by children.

2.2 Technology (ICT) in the classroom

The study of Information Communication Technology (ICT) is closely related to pedagogy and has long formed debate within the education literature. In the 1960s, Papert and others suggested using the computer as a tool to enhance learning (Leaning, 2010). Initially ICT was viewed as a means for teaching science and other technology-based subjects. However, it became clear through Papert's work that the computer could be a tool for creative learning in a range of academic disciplines for children. During the 1980's, the Piagetian theory of real world experiences and the constructivist views of learning were used to argue against computers in the educational environment as they were considered to be contrary to real world and "too abstract" (Yelland, 1999, p. 5). However, it is now largely agreed that technology

alone cannot replace good educational practice, but should be part of a "blended" solution, in which technology is "integrated into a coherent educational program" (Leaning, 2010, pp. 240–241) alongside appropriate teaching interventions.

Tamim et al (2011) argue that the debate about technology's role in education has still not been fully resolved, even after numerous studies at all levels of the education system globally that date as far back as the 1960's. Sims (1998, p. 630) states, "there remains much to learn about the impact of interactivity on learning within the context of computer-based applications".

Differing from our own work, the literature often examines the use of high-level or globalised concepts and resources such as a computer, an Interactive White Board (IWB), an iPod Touch or other such hardware in the classroom. These are primarily examined as methods for distributing or delivering content or educational outcomes. In contrast, our research will seek to understand information behaviour processes and develop tools for children's information acquisition. These tools will not be designed for content creation or information dissemination, but to provide solutions for children's successful information search and information problem solving.

A child's successful use of both digital information and information technology typically is measured by the child's ability to browse, search and find information in a digital context. Successful use of information and technology for information acquisition is integral to current educational practices. Investigation into the specific problems children have with information technology for information search and use will be required in future work in our ongoing study and therefore our current survey results reported here are the first step towards developing such investigations and in future tools that immediately and directly meet the needs of New Zealand children in the classroom and home.

2.3 Focus of our work

From our study of related work we observe the dearth of literature to encourage best principles of design for children's digital material *in general*. Specific work is required to assist with developing a list of requirements for software that supports children's digital information acquisition. We hypothesise that tools are needed that are specifically aimed at encouraging good information practices for children which assist with their development as self reliant information and library users. These tools must be born out of an understanding of information behaviour models used by children. Overall, we identify the following problems and gaps in current literature:

1. Sparse research with respect to children's digital document use for information acquisition in a New Zealand context
2. Tools are required that assist with children's effective information behaviour processes
3. Investigation into digital document interface design considerations for both adults and children is required

Our ultimate goal with this ongoing research is to develop tools that support children's information acquisition with

digital documents and can be deployed to children in New Zealand schools and homes.

Our ongoing investigation therefore targets the following broad research question: What is required to improve children's information acquisition with digital documents?

As a first step to answer the question and close the gaps identified above, we executed the survey reported in this paper, which analysed ICT and information acquisition behaviour of children in New Zealand schools (addressing Problem 1). The survey was structured around four scoping questions:

- a. What technology are local schools using in the classroom?
- b. What technology do children have access to in the home?
- c. What ages are appropriate focus years for our further research developing tools to support information acquisition?
- d. Given the free choice, do children search for information on digital devices or in print?

The next section (Section 3) details the results from the survey in suburban schools in the Waikato region of New Zealand, Section 4 discusses the results from a rural school.

3 Suburban Schools Survey

New Zealand government-funded schools at pre high school level (ie. Years 1 through 8) are typically separated into *primary schools* (catering to new entrant Year 1 through Year 6) and *intermediate schools* (catering to Year 7 and 8). In New Zealand Year 1 students begin school at five years old. Primary schools in New Zealand are typically streamed into individual year levels with a teacher facilitating a single year level in a classroom. Intermediate schools in New Zealand are often composite classrooms with a teacher facilitating Year 7 and 8 students in a single classroom. This survey was conducted in the Waikato region of New Zealand, which is located in the central North Island of the country. Using the *Directory of Schools* (Education Counts, 2013) we estimate that the Waikato Region of New Zealand has approximately 252 schools that can be classified as either, Full Primary, Contributing, Composite, or Intermediate schools (that is schools that are pre high school level in the New Zealand School System).

3.1 Method

Principals of 27 suburban schools at pre high school level in the Waikato Region of New Zealand were approached to take part in this survey during November and December 2012. Of these, 10 schools (5 primary schools and 5 intermediate schools) agreed to take part in the survey. These 10 schools fell into the range of decile ratings between 4 and 9. The Decile Rating System (Ministry of Education, n.d.) is the measure of the socio-economic catchment zone for a school. Three intermediate schools were decile 4, one decile 5 and one decile 9, while we had one decile 4, two decile 5, one decile 6 and one decile 9 primary school. A decile 1 rating indicates a high proportion of students from low socio-economic communities, while a rating of 10

indicates a low proportion of students from low socio-economic communities.

Two different survey forms were created, one questionnaire to be answered by parents and one questionnaire to be answered by teachers. Approximately one teacher at each year level at each school was approached to take part in the survey. We asked the principal to identify a teacher at each year level to be invited to take part. Teachers were explicitly informed that they could opt to not take part in the study. Principals were asked to choose teachers who fitted a standard model of their school environment rather than a class who had higher technology access than typical in their school. These "technology classrooms" are present in a number of the schools that we surveyed and to the best of our knowledge were not included in this survey. If these teachers chose to take part, in turn they selected three female and three male students in their class whose parents were invited to participate in the survey. The criteria for the teacher to select students in the class was at the sole discretion of the teacher. We specifically encouraged a somewhat random choice by the teacher by requesting that a teacher does not bias their choice based on high or low technology use. Because we were working with so many different schools and required explicit ethical approval we did not develop a true random function for the selection processes. 154 parents of a possible 252 parents (61%) chose to respond to this survey, while 34 of a possible 39 teachers (87%) also chose to return surveys.

The parents were asked 11 questions about the parent's use of technology at home and at work as well as 11 questions about the child's use of technology at home and at school. The teachers were asked 14 questions about the teacher's knowledge of their students technology use at home and at school.

These survey questions directed to the parents and teachers can be grouped under the following classifications:

- Access to Technology
- Technology for Completion of School Work
- Information Sources
- Reading in print books & eBooks

This paper looks at the results of 6 questions from the teacher questionnaire and 4 questions from the parent questionnaire. This paper does not report the results of questions pertaining specifically to technology use for entertainment, internet use, and technology use for searching library catalogues. Nor are questions pertaining to technology use by the parents considered here.

3.1.1 Questions from teacher questionnaire:

Teachers were asked to consider the survey questions with reference to *some*, *all* or *none* of the students in their class. It can be expected that the teachers answered yes if 1 or more children in their class fitted the criteria of the question. The exception to this would be when the question specifically asked about *most* children.

- 1a. Please list what technology children in your class have access to at school that they use to complete school work.
- 2a. What does the school provide for in school use?

- 3a. Please list what technology children in your class have access to at school that they use to complete schoolwork.
- 4a. What do *most* children use to find information in your classroom?
- 5a. Have you set tasks this year where a child must find information from a source of their choosing? What sources would a child have used to complete this task at school?
- 6a. Do children read ebooks for education in your classroom?

3.1.2 Questions from parents questionnaire:

Parents were asked to consider the questions in their questionnaire with reference to the child that brought home this questionnaire, not for all children in their home.

- 1b. What technology does your child have access to at home?
- 2b. What technology does your child use to complete school work?
- 3b. What does your child use to find information a) at home b) at school?
- 4b. Does your child read ebooks for education at home?

3.2 Sample

This survey received the following sample of responses across years from parents and teachers.

Year Level	Parents	Teachers
Y1 & 2	33	7
Y3 & 4	23	6
Y5 & 6	32	7
Y7 & 8	58	12

Table 1. Sample of responders

3.3 Results

We present here preliminary analysis of the results of this survey of teachers understanding of their pupils and parents understanding of their children's technology use and information acquisition.

3.3.1 Survey of Teachers

The results of the teacher survey are presented as year level composites of the teachers' responses as it became apparent from the responses that all Year 7 and 8 teachers teach composite classrooms rather than streamed year level classrooms. For this reason creating composites of the results of Years 1 and 2, Years 3 and 4 and Years 5 and 6 gave the clearest comparable data sets.

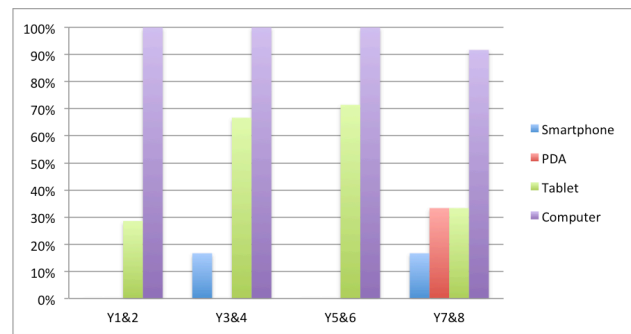


Figure 1. Technology that Children Access at School

Teachers reported a variety of technologies that children had available for use at school (Figure 1). Computer and tablet appearing to be the most available technology at all year levels. PDA was also listed at years 7 & 8 as being readily available compared to other year levels.

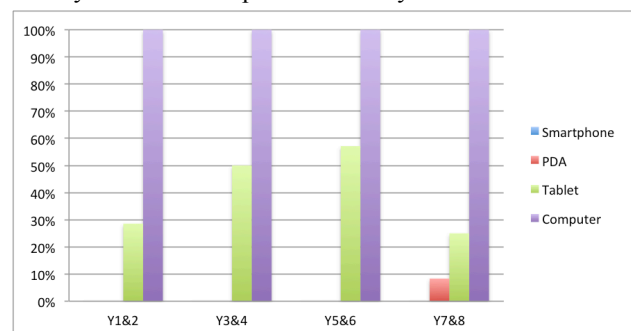


Figure 2. Technology Supplied by the School

Figure 2 reports technology the school was listed as supplying for students to use at school. In addition to the technologies listed here, a number of analogue technologies were also named by the teachers, such as audio, video and photographic technologies. All schools were reported as supplying computers at all year levels and tablets were also identified at all year levels surveyed. Interestingly only at Intermediate (Year 7&8) were PDA's (such as iPod Touch) listed.

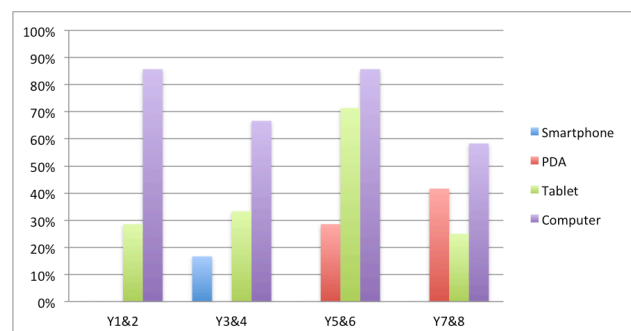


Figure 3. Technology used to Complete Schoolwork

Teachers reported that students complete schoolwork (either at home or at school) using computers as the most common tool at all levels. They also named tablets being used at most year levels for completion of schoolwork. For this question technologies were named without prompting. By contrast, the answers reported in Figure 4 were chosen by the teachers from a set of listed technologies as listed in the key of Figure 4.

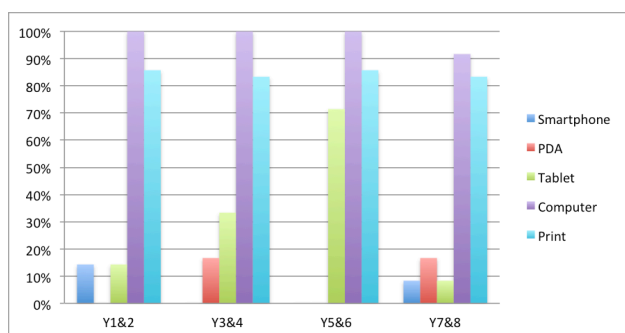


Figure 4. Technology used to Find Information

Teachers stated that most children predominantly use Print and Computer for information finding in the classroom. However, at Years 5&6 we can see substantial use of Tablets used to find information.

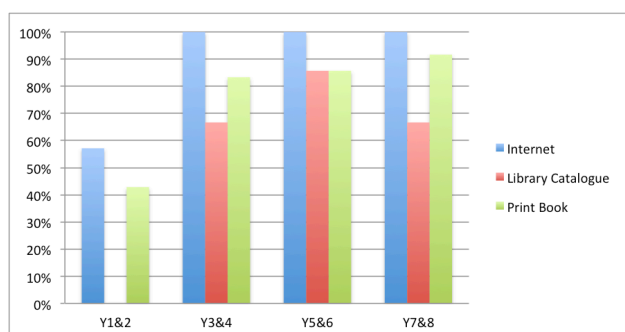


Figure 5. Technology used to Complete a Set Task

The high use of the Internet and library catalogues shown in Figure 5 supports the need for investigation into children's information acquisition in digital environments. It is interesting to note that at all 4 levels teachers assert that children are most likely to use the Internet to find information on a set task as apposed to a printed book.

Very little use of eBooks is described for finding information for a set task. Only one teacher suggested that children in their class used eBooks at school to complete a set task and this is not represented in the above figure.

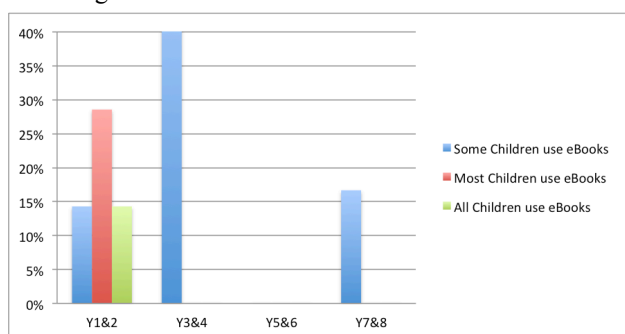


Figure 6. Reads eBooks for Education at School

In Figure 6, we see how little eBooks are used at school (note that the Y-Axis ends at 40%). Surprisingly teachers noted highest use by children at Years 1 & 2.

3.3.2 Survey of Parents

Due to the need to create composites of the teachers' data, we have used those same composites for the year level data for the parent data.

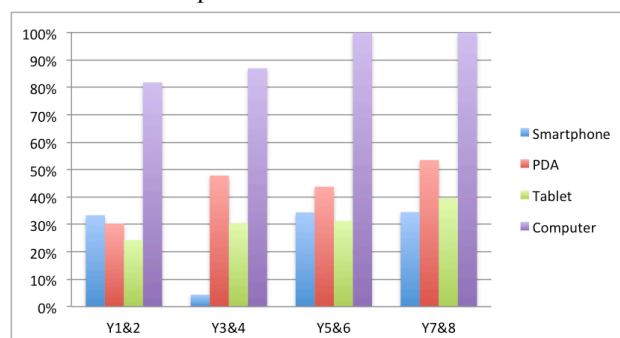


Figure 7. Technology that Children Access at Home

Parents reported a fairly even spread of access to technology by children in the home. Most interestingly, we don't see a significant spike at any year level for any of the technologies listed.

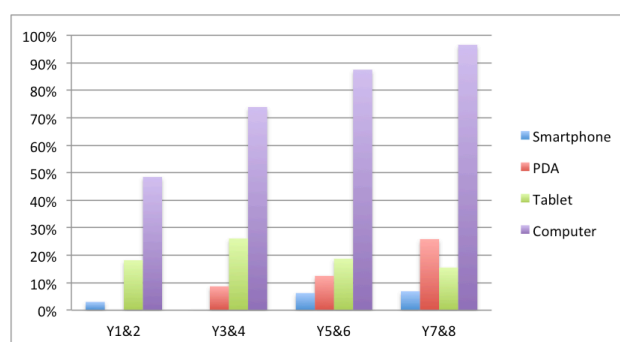


Figure 8. Technology used to Complete Homework

Parents reported that children at Years 6, 7 and 8 were more likely to use PDA or Smartphone for homework and total technology use appears to be higher for children in these 3 Year levels.

Interestingly, Figure 7 and 8 can be read to show that access to a Tablet and use of such a device for homework is likely to be fairly spread across all year levels.

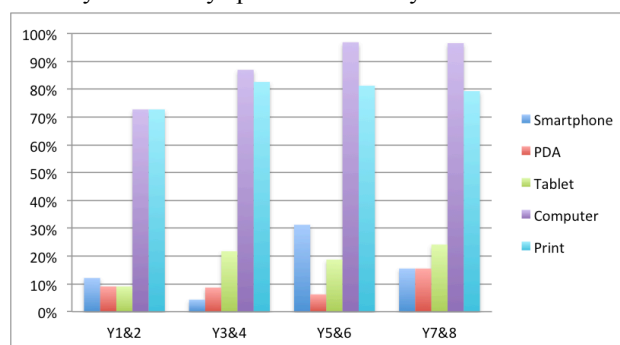


Figure 9. Technology used to Find Information

Parents responded fairly strongly that print and computers were the key tools for finding information at home or at school. Quite some way behind was the tablet with the next most recurring use at both home and school. At Year 7&8 we see reported an increase in the use of Tablets for information finding and the reverse at Years 1 thru 4.

To understand the comparative use of technology for finding information vs print for finding information we compared the two based on parents responses. This is seen in Figure 10 and Figure 11 below.

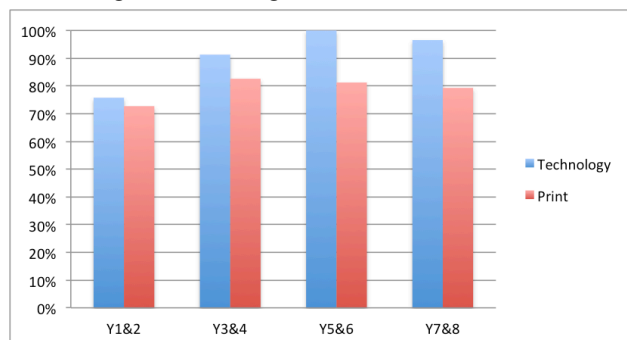


Figure 10. Technology vs Print used to Find Information at Home

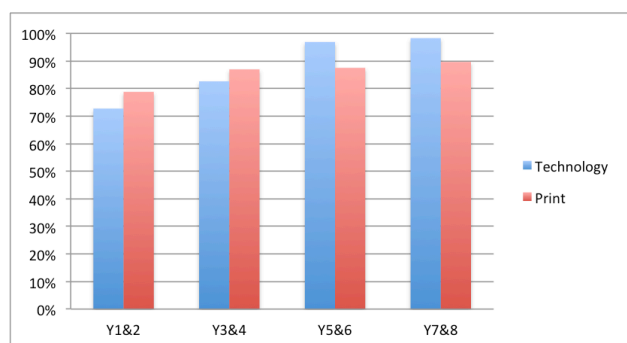


Figure 11. Technology vs Print used to Find Information at School

Most notable in the comparisons shown in Figure 10 and Figure 11 is that at home, technology is believed to be very nearly as likely to be used for finding information by children as print sources. At most year levels technology is slightly more likely to be used than print both at home and at school. Parents also believe that print is more likely to be used than technology for children at Years 1 through 4 at school. Interestingly the difference in technology use compared to print use is more pronounced for technology use at home than at school. The most marked difference in the use of technology compared to print for finding information appears to be at Years 5&6 and Years 7&8 in the home and at school. There appears to be less marked difference to technology use at school compared to at home and this is of course due to the lesser use of Smartphones and PDA's for finding information at school as was shown in Figure 9 above.

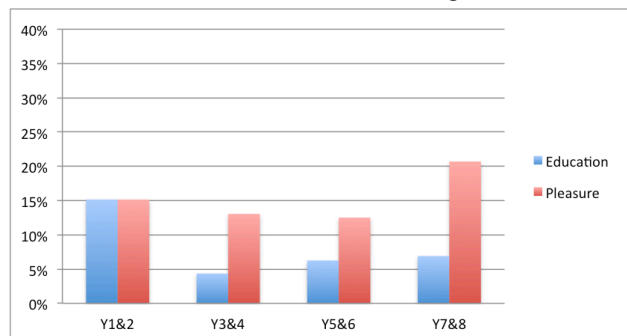


Figure 12. Reads eBooks for Education at Home

When investigating children's reading of eBooks for pleasure and for education at home it appears that children are more likely to be found reading eBooks at home for pleasure than for education. In Figure 12 (note that the Y-Axis ends at 40%). we can see that very low numbers of children's parents believe their children use eBooks for either pleasure or education at home, with seemingly slightly increased numbers of students at Year 7 & 8 more likely to read eBooks for pleasure than at any other level. Interestingly children as young as Year 1 and 2 are indicated in this study as using eBooks for pleasure and education at home.

4 A Rural School Case

To strengthen our investigation, and to ensure we have a picture of the situation in a range of publicly funded pre-high school level schools typical in New Zealand, we have also conducted our survey at a rural school in the Waikato region of New Zealand.

Differing from the Suburban Schools described in Section 3 of this paper,

Rural schools in New Zealand typically have a much smaller enrolment and are far more geographically isolated than suburban schools. Due to the smaller school roll of a rural school it is typical for the teaching to be conducted by a single teacher with all year levels in a single classroom. This is the case for the school who took part in this survey where the entire school is facilitated-as a single classroom across Years 1 through 6. These schools are known by a range of terms in New Zealand and internationally, including Country Schools, Rural Schools, Isolated Schools, and One Room Schoolhouses.

We have used the *Directory of Schools* (Education Counts, 2013) to estimate that approximately 7% (176 of 2503 schools) of the total number of New Zealand schools listed are Rural Full Primary, Contributing, Composite, or Intermediate schools with rolls less than 30 pupils. We also estimate that 9% (23 of 252 schools) are Rural Primary, Full Primary, Contributing, Composite, or Intermediate schools with rolls less than 30 pupils in the Waikato Region of New Zealand.

4.1 Method

The Principal of a single Rural School in the Waikato Region of New Zealand was approached to take part in this survey during March 2013. This school caters to students at years 1 to 6 with a current roll of 12 students and a teaching staff of two teachers. One teacher facilitates the learning for all 12 students at any one time.

The same two survey forms were answered by parents and teachers as was used in the public schools survey. Both teachers were invited to participate in the survey and the parents of all 12 children were invited to take part in the survey. Where a family had more than one child at the school the family was invited to choose to have one parent answer the survey for one child only or to have both parents independently fill in surveys. 8 parents of a possible 12 parents (66%) chose to respond to this survey, while 2 of a possible 2 teachers (100%) also chose to return surveys.

4.2 Sample

This survey received the following sample of responses as shown in Table 2.

Level	Y1	Y2	Y3	Y4	Y5	Y6	Y?	Y1-6
Parents	0	1	1	0	3	1	2	
Teachers								2

Table 2. Sample of responders

We received surveys from parents of children in Year 2, 3, 5 & 6 as well as 2 surveys where the year level of the child was not listed. We also received surveys from both of the teachers at this school.

4.3 Results

Due to the small numbers of students and teachers at this rural school we compare here composites of the results from Primary schools (Years 1 through 6) and Intermediate schools (Years 7 and 8) with the entire rural school (Years 1 through 8).

4.3.1 Survey of Teachers

We compare here the Rural School with the Suburban Schools surveyed in Section 3 of this paper.

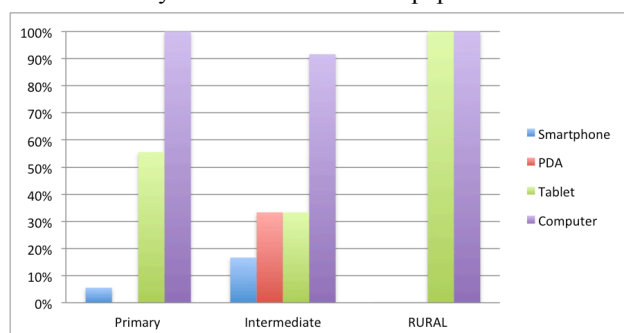


Figure 13. Technology that Children Access at School

As can be seen from Figure 13 very little Smartphone use was identified at suburban primary schools and intermediate schools, however, no Smartphone use was identified at the rural school. No PDA use was identified at either the suburban or rural primary school, though was noted at suburban intermediate schools. The strong difference is the access to Tablets at the rural school is substantially higher (100% access) compared to the suburban primary schools (55%) and intermediate schools (30%).

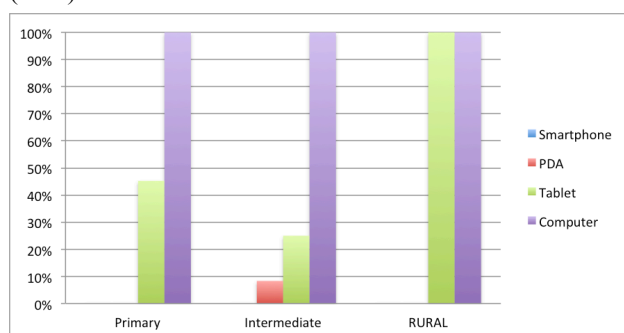


Figure 14. Technology Supplied by the School

As with technology access shown in Figure 13 we see in Figure 14 that the rural school supplies tablets to the entire student roll for in school use.

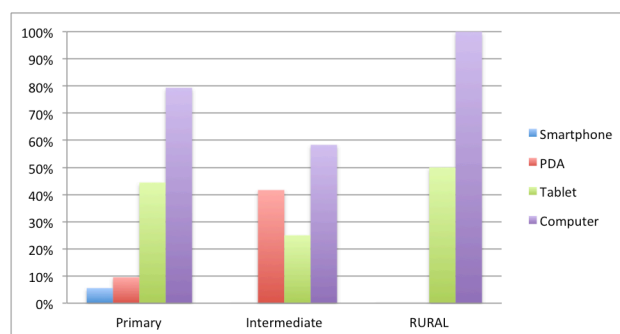


Figure 15. Technology used to Complete Schoolwork

While Figure 13 and Figure 14 seemed to show high access to Tablets in the rural school compared to the suburban schools Figure 15 suggests that teachers from both rural and suburban schools feel that students use tablets to complete schoolwork to a fairly similar degree. Computer use to complete schoolwork is suggested as being higher for the rural school. While there is some Smartphone and PDA use identified by teachers at suburban schools for completing schoolwork this is not the case for the rural school.

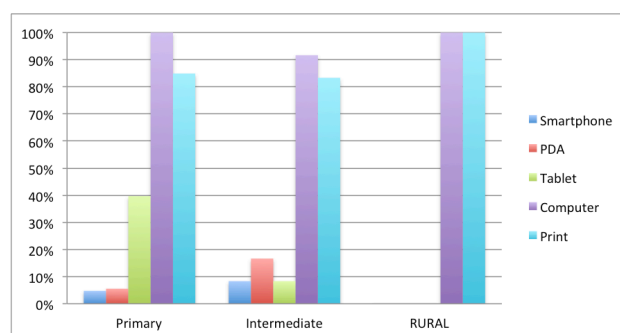


Figure 16. Technology used to Find Information

When asked about technology used to find information the use of print and computer was fairly comparable for the rural school and both the suburban primary and intermediate schools. However, both of the suburban primary and intermediate schools also listed use of Smartphone, PDA and Tablet which is not listed for the rural school.

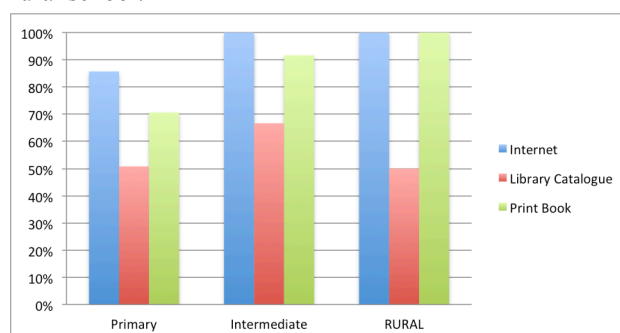


Figure 17. Technology used to Complete a Set Task

The results of what technology children used to complete set tasks was comparable across the suburban schools and the rural school.

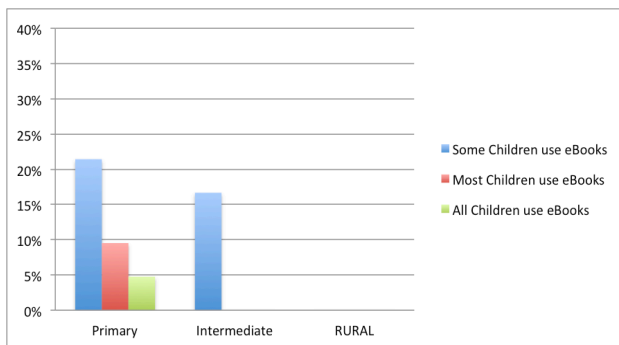


Figure 18. Reads eBooks for Education at School

We see very little use of eBooks at school in the suburban schools and no use of eBooks recorded by the teachers of the rural school (note that the Y-Axis ends at 40%).

4.3.2 Survey of Parents

We compare here the parent answers at the Rural School with the answers from Suburban Schools surveyed in Section 4.

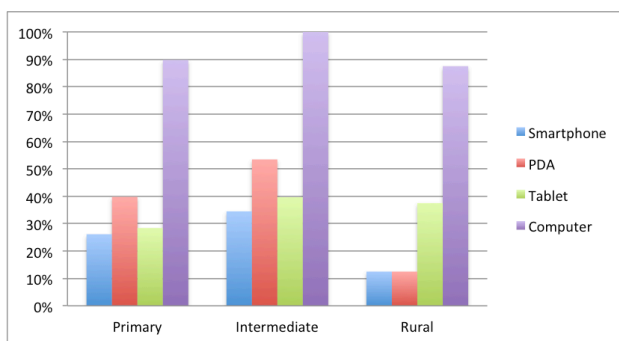


Figure 19. Technology that Children Access at Home

The major difference between technology access at home for children who attend suburban schools compared to children who attend the rural school is that parents note much higher access to Smartphones and PDA's at the home for the suburban schools children.

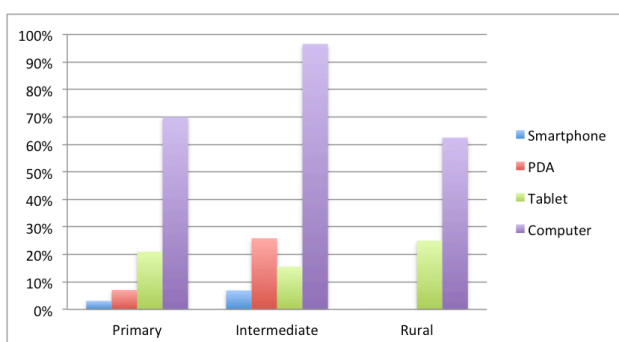


Figure 20. Technology Children Use to Complete Homework

As was seen in Figure 19 children at the rural school do not have access to PDA or Smartphone and thus it is unsurprising that Figure 20 shows no PDA for Smartphone use for homework completion and Figure 21 shows no PDA or Smartphone use at school by rural school children.

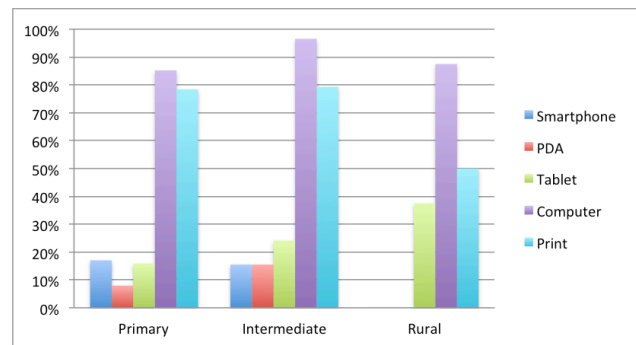


Figure 21. Technology used to Find Information

Computer use for information finding is reported as similar at both rural and suburban schools. Parents seem to consider that print use for information finding is used much less prominently than computers for information finding at the rural school. We see an increase in the reported use of Tablets for information finding compared to completion of homework for the intermediate children and the rural children.

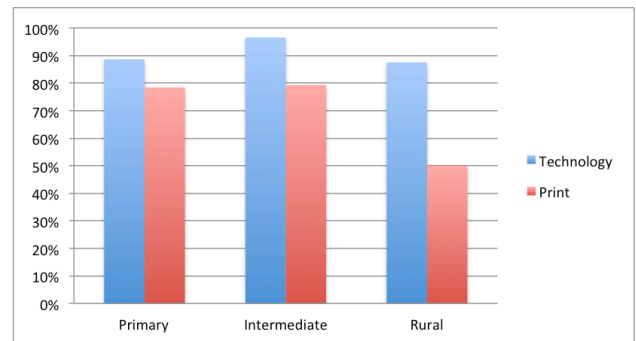


Figure 22. Technology vs Print used to Find Information at Home

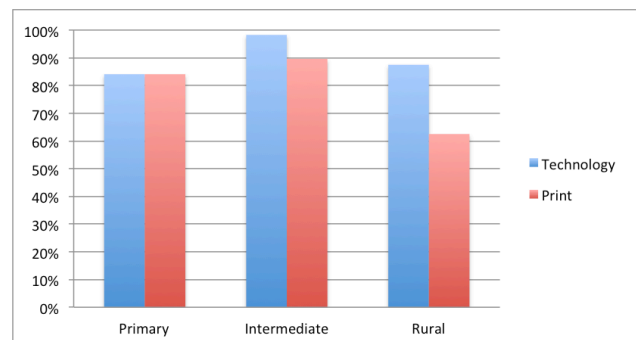


Figure 23. Technology vs Print used to Find Information at School

Most notable in the comparisons shown in Figure 22 and Figure 23 is that parents believe that at home technology is as likely to be used for finding information by children as print for suburban school children, while children of the rural school are more likely to use print for finding information.

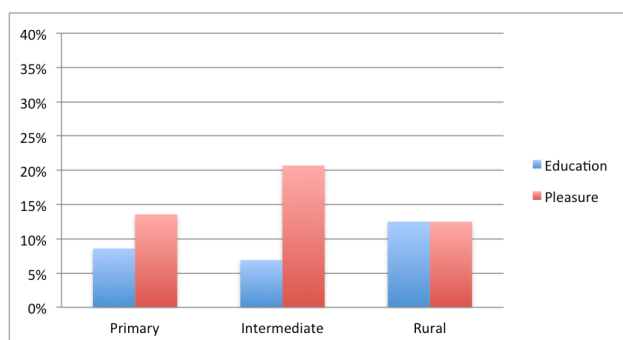


Figure 24. Reads eBooks for Education at Home

As was seen in Figure 12, in Figure 24 (note that the Y-Axis ends at 40%) we see the numbers of children reading eBooks at home for pleasure or education is very low for suburban school children and for rural school children. Parents of rural school children consider their children as likely to be reading eBooks for pleasure as for education compared to suburban school children who were more likely to be reading eBooks for pleasure.

5 Discussion

This section firstly considers answers to the scoping questions posited in Section 2 of this paper. The results of the rural case and their implications are reviewed. Finally, the use of and access to technology compared to print media for information acquisition is considered.

5.1 Answering our scoping questions

The survey was structured around four scoping questions that sought to understand the ages at which children used information technology in the home and in the classroom for information acquisition as well as the comparative use of print or technology for information acquisition.

- What technology are local schools using in the classroom?
- What technology do children have access to in the home?
- What ages are appropriate focus years for our further research developing tools to support information acquisition?
- Given the free choice, do children search for information on digital devices or in print?

We now have indicative answers to these questions that we discuss here.

(a&b) From our analysis of the results of this early study we observe that children use a range of technologies including Tablets, PDAs, Smartphones and Laptops/Desktop computers both at school and at home. Therefore information acquisition solutions should not be constrained by a specific technology. We would also argue that this range of technologies available at many age ranges justifies the further investigation of any individual one of these technologies. For finding information parents reported that Year 7&8 children were likely to use Tablets for finding information more than for completing homework, with the reverse being reported by parents of Years 1 thru 4 children.

(c) The observation that technology use seems fairly spread across all year levels does not assist with decision making for targeting of future investigations. However,

our findings may indicate that age group targeting will allow for generalizability of results of future studies. For our future studies we therefore consider age group selection based on the impact of comprehension or reading skill on the task. Given the slightly higher numbers of parents and teachers describing technology access across the range of technologies discussed in this survey at years 7 & 8 (11 and 12 years old) and 5 & 6 (9 and 10 years old) it is likely that these will be fruitful year levels to concentrate on with future studies. The higher numbers of responses of both parents and teachers at these year levels also indicates that these will be suitable year levels to target with future research due to a willingness to participate in studies of this nature. Thus we would argue that Years 5 through 8 are pertinent years for our studies.

(d) Given the relatively high use of some of the mobile information technology devices such as Tablet in the home and the classroom it is interesting to still see limited use of eBooks in the home and classroom. Reasons for this low uptake are unknown and bear further investigation.

5.2 Considering the rural case

From conversations with the principal of the rural school, the researchers are aware that the school has access to both broadband internet and wireless networking for the students, and a well resourced computer lab. As is noted in the results section, the school supplies one to one tablet devices to the students and teachers for use at school. One of the eight parents who responded to the rural survey specifically answered the survey “sorry, we do not have internet in our area”. This is indicative of the fact that access to broadband internet at home for students of rural New Zealand schools is likely to be limited. Mobile devices, and internet resourcing in schools is a factor in research that takes into consideration children living in rural or remote locations and attending rural schools.

In our study it was unique to see that 100% of the student body at this rural school had access to a tablet supplied by the school. Interestingly however, the teachers did not list the tablet as being used to find information for a set task at this school. This is in contrast to the finding that approximately 40% of teachers listed primary school children in suburban schools as finding information using a tablet device at school.

We believe that continuing to work closely with this rural school will benefit our own and others investigations. The small roll and commitment to collaborative investigation on the behalf of the school board, the schools parents and the principal create an opportunity to work on investigations in a close manner.

5.3 Print Compared to Technology for Information Acquisition

While the title of this paper uses the terms *Educational Technology Use* we must not neglect the consideration of the printed medium for information acquisition. At both primary and intermediate level for the suburban schools surveyed we report here increased use of technology for finding information at home and at school. This increase warrants further study of tools for children’s digital

information acquisition and supports the need for development of tools that are immediately deployable in New Zealand schools and homes. This noted increase in technology use for finding information was not noted for the rural case and may indeed be due to the limited numbers reported as well as the noted limited internet access at home for some of these responders.

Studies (McKay et al., 2012) have shown increased eBook sales, however from our survey we do not see this across the age ranges surveyed. This survey suggests that while students in New Zealand are accessing eBooks in very small numbers this is not yet being driven by teachers in a formal manner for information search or perhaps even for reading for pleasure or learning to read. This contrasts the findings of Digital Book World and PlayScience (Shuler, 2013) who report that that over half of all American children in their survey read eBooks. Shuler also reports that 85% of those children who read eBooks are doing so at least once per week.

6 Summary & Future Work

This ongoing research has the overarching goal to discover what intervention is required to improve children's information acquisition with digital documents. As was shown by the related work, there is a dearth of literature that describes the current situation in NZ classrooms and homes with regards to ICT access and use for information acquisition.

The initial survey reported in this paper has given direction for future work in this area. We saw that technology use compared to print for information finding was slightly higher for children in years 5&6 and 7&8 both at home and at school. Age range targeting of future investigations is a relevant concern because reading, comprehension and literacy skills vary drastically from Year 1 to Year 8. Therefore a year level or year-level-range will need to be identified for future studies based on the types of information tasks that are completed by students at a particular year.

This survey has detailed the range of information acquisition technologies that children have access to at home and at school to assist with development of appropriate future studies. Future studies will investigate how these technologies are used for specific information acquisition tasks. Our next step is to investigate the types of information tasks completed within the classroom and the ways that children approach these tasks. We hope to identify the problems encountered with tasks as well as the affordances of current tools. This next step will bring us closer to our goal to develop tools that support children's information acquisition with digital documents.

This survey has also begun to answer our question of whether children search for information in print or using digital devices, however, presently we cannot argue why they use a particular medium more than the other. We offer here a snapshot of the current situation of ICT use for information acquisition in a New Zealand classroom and home. This important first step serves as a platform for our own and others' investigations in this field.

7 References

- Brand-Gruwel, S., Wopereis, I., Vermetten, Y., 2005. Information problem solving by experts and novices: analysis of a complex cognitive skill. *Computers in Human Behavior* 21, 487–508.
- Cooper, L.Z., 2005. Developmentally appropriate digital environments for young children. *Library Trends* 54, 286(17).
- Education Counts, 2013. Ministry of Education - Education Counts <http://educationcounts.govt.nz> Accessed 15 October 2013.
- Leaning, M., 2010. The one laptop per child project and the problems of technology-led educational development. *High-Tech Tots: Childhood in a Digital World* 231–248.
- Li, X., Atkins, M.S., 2004. Early childhood computer experience and cognitive and motor development. *Pediatrics* 113, 1715–1722.
- McKay, D., Buchanan, G., Vanderschantz, N., Timpany, C., Cunningham, S.J., Hinze, A., 2012. Judging a book by its cover: interface elements that affect reader selection of ebooks, in: *Proceedings of the 24th Australian Computer-Human Interaction Conference, OzCHI '12*. ACM, New York, NY, USA, pp. 381–390.
- Ministry of Education, n.d. Ministry of Education - Deciles Information. Ministry of Education. URL <http://minedu.govt.nz>. Accessed 15 October 2013.
- Shuler, C., 2013. Back to school in an e-reading world understanding the e-reading habits of children aged 2-13, with a focus on educational ebooks and ebooks in the classroom., *ABCs of kids and ebooks*. [S.l.] : Digital Book World, 2013., New York.
- Sims, R., 1998. Interactivity or narrative? A critical analysis of their impact on interactive learning, in: *Proceedings of ASCILITE*. pp. 627–637.
- Tamim, R.M., Bernard, R.M., Borokhovski, E., Abrami, P.C., Schmid, R.F., 2011. What Forty Years of Research Says About the Impact of Technology on Learning. *Review of Educational Research* 81, 4 –28.
- Timpany, C., Vanderschantz, N., 2011. Learning outcome dependency on contemporary ICT in the New Zealand middle school classroom, in: *Proceedings of the 12th Annual ACM SIGCHI-NZ Conference on Human-Computer Interaction*. New York, ACM Press, pp. 65–72.
- Yelland, N., 1999. Technology as Play. *Early Childhood Education Journal* 26, 217–20.

Understanding Saudi Arabian students' engagement in E-learning 2.0 in Australian Higher Education

¹Omar Mayan, ²Judy Sheard and ³Angela Carbone

¹
Faculty of Information
Technology
Monash University, Australia
Sponsored by: Ministry of Higher
Education: Saudi Arabia

ohmay1@student.monash.edu

²
Faculty of Information
Technology
Monash University
Australia

Judy.Sheard@monash.edu

³
Office of the Pro Vice-
Chancellor
(Learning & Teaching)
Monash University
Australia

Angela.Carbone@monash.edu

Abstract

This paper focuses on understanding Saudi Arabian students' engagement in e-learning 2.0 in Australian higher education. Eight Saudi students enrolled in the Australian Higher Education were interviewed to discuss their experiences and attitudes towards e-learning 2.0 using Semi-structured interviews. A qualitative approach was adopted to analyse the gathered data. The approach was based largely upon Charmaz's constructivist grounded theory.

Key findings indicated that Saudi Arabian students were able to utilise the e-learning 2.0 settings in their respective universities as tools in which they interacted with other people while preparing themselves to become more interactive in their classes. At the same time, e-learning 2.0 served as a means for these students to steadily get over the socio-cultural barriers that might hinder them from making the most out of their education in Australia. However, it was also found that the language barrier that persisted even in the e-learning 2.0 environment made it more challenging for students to break through other barriers. Furthermore, it was found that the gender segregation culture that Saudi Arabian students have been used to still affected them in Australia, even in taking advantage of the e-learning 2.0 opportunities. This paper presents a discussion of four axial codes /categories that were identified that shape the Saudi students' attitudes, experiences and their engagement with e-learning 2.0. Specific attention is given to the 'Engaging in learning through technology' axis.

Keywords: E-learning 2.0, Web 2.0, Educational Technology, Engaging in e-learning 2.0, Australian Higher Education, Saudi Arabian Students, Qualitative Research, Grounded Theory.

1 Introduction

This paper is part of a large project investigating the experiences and attitudes of Saudi Arabian higher education students in Australia towards e-learning 2.0, as well as the opportunities and challenges emerging from these experiences and attitudes. It examines the role of culture in shaping attitudes and experiences of these students. One of the aspects of difference that foreign students face is the change in learning environment. In this digital age, western universities have led and continue to lead developments in e-learning, the utilisation of different online resources in order to better facilitate learning. One of the latest developments in this area is e-learning 2.0, which is a shift in the focus of using online learning resources from delivering knowledge to students, to enabling students to build knowledge on their own (Downes, 2007). This shift is far different from what students with Middle Eastern or Asian backgrounds may be used to, as educational institutions in the Middle Eastern region, for instance, tend to be teacher-centric, and non-participatory (Mahrous and Ahmed, 2010, Tubaishat et al., 2006). As such, there are many issues which foreign students in western or westernised universities have to deal with in terms of this new learning environment. It is in the investigation of some of these issues that the topic of this dissertation is situated.

There is strong impetus for research to be conducted on how foreign students experience e-learning 2.0 in western or westernised universities, and how these experiences shape or change students' attitudes about e-learning 2.0. It is likewise important to determine how these experiences and attitudes affect the challenges and opportunities that foreign students face in e-learning 2.0 environments, and how different cultural factors influence their perception and use of e-learning 2.0 resources.

Since 2005, the Ministry of Higher Education in Saudi Arabia launched a programme that sought to help raise higher education achievement among the Saudi Arabian citizenry. This is the King Abdullah Scholarship Programme (KASP) which to date has provided over 150,000 full scholarships for Saudi Arabian students to study in prestigious higher education institutions around

the world (Ministry of Higher Education in Saudi Arabia, 2012). Approximately 11% of students who are approved for the scholarship go to study in Australia (Ashrqia Chamber, 2011).

Saudi Arabian foreign students were selected for the focus of this study for two reasons. Firstly, Saudi Arabian cultural norms are very different from western culture which predominates in Australia, in many aspects, such as in matters of propriety, liberty and gender roles. Secondly, the educational systems in the two countries are very different, specifically, in higher education. For example, in Saudi Arabia, the higher education system is based on gender segregation (AlMunajjed, 1997), while in Australia, all university education is co-educational. Further, the learning environment in Saudi Arabia is teacher-centric and non-participatory (Mirza and Al-Abdulkareem, 2011). In contrast, the Australian learning environment is participatory and democratic, where both students and instructors have rights (Bradley et al., 2008). The concern is finding out how this contrast affects students' experiences and attitudes.

Understanding Saudi Arabian students' engagement in e-learning 2.0 resources in the Australian educational setting can help improve both learning and teaching practice by providing such students with a deeper understanding about the opportunities that they can draw from e-learning 2.0 resources. In addition, it can help educators in Australia gain greater knowledge of the learning needs of their foreign students and how to utilise e-learning 2.0 resources in order to maximise the benefits for these students and help them overcome the challenges that emerge. This benefit is in line with the new international paradigm of inclusive education proposed in Mitchell (2005, p. 32), where it is recognised that people from a foreign culture are introduced into a new educational system, educators within that system must be capable of modifying the system to fit such foreign students' diverse needs. Thus, this work can serve as a stepping stone for other studies relating to how students from foreign backgrounds experience e-learning 2.0 in the Australian setting.

2 Background

2.1 The Concept of E-learning 2.0

E-learning 2.0 is dependent on the tools of Web 2.0 that have "blurred the line between producers and consumers of content", and made possible access to other people rather than merely access to information (Brown and Adler, 2008, p. 18). Brown and Adler (2008, p. 19) believe that the most important contribution that Web 2.0 is making to e-learning is the creation of a participatory medium that is able to support social learning.

Social learning is not simply about "what we learn, but how we learn" – and Brown and Adler (2008, p. 20) point to the way apprentices are gradually inducted into becoming "full participants in their field." In attempting to describe the most important features of e-learning 2.0, Ehlers (2009) suggested that "self-directed" learning is a key feature of e-learning 2.0. Another feature of e-learning 2.0 described by Ehlers (2009) is that it enables

learners to use informal, networked, and electronically supported learning.

2.2 Main Web 2.0 Tools for E-learning 2.0

There are a number of Web 2.0 tools that are used in e-learning 2.0 settings. Some examples of these are blogs and microblogs, which facilitate the discussion of course-related topics (Borau et al., 2009): wikis which enable individual members of a class to contribute their own inputs in developing a class-wide knowledge base on course content (Hoewe et al., 2012, Young and Pérez, 2012): and podcasts, and content sharing tools which enable students to share and absorb a wide range of multimedia information relevant to their course content (Krauskopf et al., 2012).

2.3 Affordances of E-learning 2.0

According to Faiola and Matei (2010), affordances of an educational construct are qualities that allow students that perform certain actions. E-learning 2.0, through the use of Web 2.0 resources, has a variety of affordances for students. Firstly, such an environment allows students to develop their own learning content, enabling them to have a deeper familiarity with the subject matter. The student comes to treat the course as much more than just a subject that he or she has to take. Secondly, e-learning 2.0 allows the student take greater control of the learning process, and reduces the imbalances in the power relationships between students and teachers. Thirdly, e-learning 2.0 broadens the scope of student interactivity, and empowers the development of a learning community.

2.4 E-learning 2.0 in Saudi Arabia and Australia

The development of e-learning in Saudi Arabia has been relatively slow compared to western countries (Al-Shehri, 2010). However, considerable progress has been made in the last five years, during which a "national centre for e-learning was established and e-units or departments have been set-up in almost every university". Still, these developments in e-learning seem to be focused on *E-learning 1.0* rather than e-learning 2.0. E-learning 1.0 platforms have been developed to enable universities to offer online courses and empower faculties to produce online learning content. However, studies on the possible utility of Web 2.0 tools in Saudi Arabian higher education settings could not be found. As reported by Harrison (2008), the culture of teaching and learning in Saudi Arabian schools still tended to be teacher-centred, which is in line with the e-learning 1.0 platform but not e-learning 2.0 environments.

The differences in the e-learning 2.0 environments that are present in the two countries, Saudi Arabia and Australia are evident. Australian universities have been developing their e-learning systems since the late 90's (Marshall, 2011), and have come to accept and integrate changes brought about by the development of e-learning 2.0 in those systems (Kirkwood, 2010). On the other hand, Saudi Arabian higher education institutions have been considerably slower in terms of establishing their own e-learning infrastructures (Mirza and Al-Abdulkareem, 2011), and have not yet even begun

adapting to changes brought about by the development of e-learning 2.0. Based on this difference, it can be surmised that Saudi students are less accustomed to Web 2.0 technologies being applied to e-learning 2.0 than western students.

2.5 Opportunities and Challenges of E-learning 2.0

Based on the descriptions of e-learning 2.0 provided in the preceding sections, it is clear that Saudi Arabian students can potentially gain access to many opportunities through immersion in the e-learning 2.0 environment. They would be able to gain access to insights of different people across the world; people who are involved in the same fields of study. In so doing, such students would be able to gain greater, more comprehensive knowledge in their degree programmes. At the same time, they would be able to expand their social and professional networks, gaining access to possible collaborative endeavours or future employment opportunities. However, Saudi Arabian students also need to engage in the challenges of e-learning 2.0, which is basically that it is built on principles of openness and liberty that may be in direct contrast with their own cultural upbringing.

3 Methodology

3.1 Qualitative research design

The qualitative research design was selected for this study. According to Merriam (2009, p. 6), the qualitative research design is a "social" research design that mainly utilises an *interpretive* paradigm of truth-seeking. First, this means that the design is only applicable to studies that involve people. This is because data gathered from qualitative research are not completely objective, but are dependent on the subjective perspective of the respondent. Secondly, qualitative research accepts that different people experience the world in different ways. As such, it is in how people interpret their experiences that truth is found.

In terms of how research is conducted in a qualitative design, Merriam (2009) explained that research is carried out holistically, through a process of organising and identifying patterns from narrative descriptions of people who are of relevance to the phenomenon of interest.

3.2 Grounded Theory

Sociology researchers, Barney Glaser and Anselm Strauss originated their qualitative research design of Grounded Theory in 1967 (Glaser and Strauss, 1977). This is a research method, which is quite different to the usual analysis, because it does not start with a hypothesis, and a theoretical framework. Rather, after collecting data, and coding it, the codes are grouped in various formations, and systematically analysed, until relationships can be established and a theory developed.

There are three main types of grounded theory approaches: classic grounded theory by Glaser (1992, Glaser, 1998), the modern approach by Strauss and Corbin (1998, 1990) and recently, Charmaz (2006) has introduced a constructivist approach.

Of the three different grounded theory approaches, it was Charmaz's constructivist approach that was ultimately applied to the analysis of the data in this study. Glaser's is not strictly a qualitative approach, and encourages gathering data from various sources using various instruments in order to come up with a universal whole of the environment of interest (Glaser, 1998). The Strauss and Corbin approach is similar to the Charmaz approach. What separated the two approaches from one another is their perspective on how the theory should emerge from the data. Strauss and Corbin maintain that in the conduct of the different levels of coding, the theory should emerge on its own, with little help on the part of the researcher (Strauss and Corbin, 1998). Charmaz (2006) criticizes this view several times (Chapter 1), saying that the outlook is positivistic, assuming that there is an underlying objective reality which the researcher merely has to uncover. Charmaz's approach is more in touch with post-modernist thinking, which makes the notion of the researcher's "objectivity" problematic. Charmaz is a constructivist, which means that she believes the researcher must acknowledge his/her own subjectively, and that the researcher plays a part in constructing the theory. In the end, this seemed the most appropriate approach.

3.3 Steps in Grounded Theory

According to Charmaz (2006) coding is essential to the development of a grounded theory. Charmaz (2006, p. 46) describes the coding stages as *"the pivotal link between collecting data and developing an emergent theory to explain these data. Through coding, you define what is happening in the data and begin to grapple with what it means."* Charmaz (2006) identify a four step coding process in data analysis. These are *initial coding*, *focused coding*, *axial coding* and *theoretical coding*.

- Initial coding phase "involves naming each word, line or segment of data" (Charmaz., 2006, p. 65). It can be done through word-by-word, line-by-line segment-by-segment coding which later form basic statements (lines of code) that conveyed singular ideas. This involves a close reading of the data and remaining open to all possible theoretical data (Charmaz., 2006). Charmaz (2006, p. 166) also stresses on following Glaser's (1978) guidelines of using *"gerunds"* (verbs ending in 'ing') when naming the codes.
- Focused coding. This is the second phase which is a "focused selective phase that uses the most significant or frequent initial codes to sort, synthesize, integrate and organize" the Data gathered (Charmaz., 2006, p. 65).
- Axial coding. Borgatti (2005, p. 6) defines axial coding as "the process of relating codes (categories and properties) to each other, via a combination of inductive and deductive thinking." Charmaz (2006, p. 107 & 116) discusses that the researcher can treat focused codes as "tentative" axial codes, and then decide which can become conceptual categories or axes. This can be seen as two steps:
 1. Focused codes = tentative axial codes/categories

2. Researcher picks out main focused codes, which become conceptual axial codes/categories

Thus, at the axial coding phase, the researcher attempts to fully describe the axial codes, by spelling out the properties and dimensions, and even relationships with other axial codes (Charmaz., 2006). Corbin & Strauss (1990, p. 12) indicated that each axis emerged "can be broken down into specific properties and their dimensions", and each property can also be broken down into sup-property.

- Theoretical coding is where "to specify possible relationships" between axial codes developed throughout the focused and axial coding stages (Charmaz., 2006, p. 84). Glaser (1978, p. 72) explained that "theoretical coding conceptualises how the substantive codes may relate to each other as hypothesis to be integrated into a theory."

Figure 1, shows the substantive strategy of Grounded theory analysis as identified in Charmaz (2006).

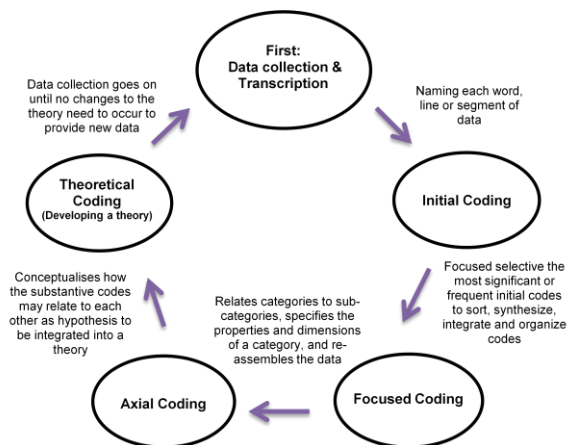


Figure 1: Grounded theory analysis strategy

4 Implementation of Grounded Theory

The following section describes the research techniques used. It provides a detailed description of the coding stages which were based largely upon Charmaz's constructivist grounded theory.

The data was gathered via semi-structured interviews. Although, this work is based on eight interviews, the full study will involve interviewing more participants until data saturation is reached. Having used the Grounded Theory (GT) method for analysing the data under a qualitative research paradigm, this study is expected to comprise 20-30 semi-structured interviews, as indicated in Creswell (1998, p. 64).

4.1 Selection of Participants

The selection of participants was done by establishing inclusion criteria, which were that the participant should be Saudi Arabian, and should be studying in a university located in Australia. An email attached with the Explanatory Statement and the Consent Form in English that provide essential details about the study was sent to all Saudi students in a Google group and a Facebook

group. Both groups were created by the researcher in preparation for the study.

4.2 Interviews

The data was gathered from the participants using semi-structured interviews. Around one hour duration interview were conducted to gather data from the respondents. As explained by Bryant and Charmaz (2007) and Charmaz (2006), semi-structured interviews work best because they provide the researcher with some control over the relevance of the expected answers to the research questions, while at the same time allowing respondents considerable flexibility in answering the questions. The same type of instrument was used in the other grounded theory studies reviewed (Omli and Wiese-Bjornstal, 2011, Sbaraini et al., 2011). In the conduct of this stage, the researcher scheduled an interview with each respondent; just as was done in Sbaraini, et al. (2011). Respondents were provided a wide array of choices for the medium of the interview, so that they could choose the medium that was most accessible and convenient for them.

Several major questions were prepared for the semi-structured interview and further sub-questions were developed to ask if it was deemed appropriate based on the participant's initial response. However, the interviewer could also deviate from these questions and their order, if they wanted to follow a specific line of questioning. The interviews were digitally recorded and transcribed (verbatim).

4.3 Response Overview

From the emails sent to participants, 21 replies were received from respondents. Out of this number, 8 interviews were scheduled with four males and four females. Their details are summarised in Table 1.

Interviewee	G	Age	Year Level	Medium	Length
I-1	M	18-25	2nd year Bachelor	Skype	60 mints
I-2	M	26-30	1st year PhD	Skype	90 mints
I-3	M	26-30	2nd year PhD	Face to face	60 mints
I-4	M	26-30	Master - Coursework	Face to face	60 mints
I-5	F	26-30	Master - Coursework	Skype	60 mints
I-6	F	26-30	Master- Coursework	Skype	60 mints
I-7	F	26-30	Master- Coursework	Skype	55 mints
I-8	F	18-25	Master- Coursework	Skype	60 mints

Table 1: Descriptive details of participants

As shown from Table 1, the respondents were diverse. This diversity is favourable in capturing the full range of experiences that Saudi Arabian higher education students have in Australia.

4.4 Coding Process

The raw transcriptions were imported into NVivo 10, a qualitative analysis software program. The transcripts of each respondent were carefully read in depth, and broken down into basic statements (lines of code) that conveyed singular ideas. That is, each statement code contained only one significant idea from the respondent. From the eight interviews, over (400) basic statements were subsequently extracted forming (65) open/initial codes.

In the process of going through the stages of coding, open/initial codes were numbered (from 1- 65) in order to keep track of them, as they went through various modifications and consolidations. For example, when a repetition in the open/initial codes was found, a consolidation was done.

The second stage of the coding process is focused coding. Frequently occurring initial/open codes which show promise in answering the proposed research questions become the focused codes.

The third level of analysis is axial coding stage (also called categories by Charmaz). In this study, the term "axial coding" is used. An axis, as explained by Bryant & Charmaz (2007), is a statement or idea relevant to the study that can best describe what a set of codes are about.

Currently, the analysis of the data has resulted in four axial codes. These are:

- Overcoming the language barrier
- Influence of cultural practices
- Adapting to the Australian education system
- Engaging in learning through technology

4.5 Properties and dimensions of the axial codes

Each of the four emerged axial codes was broken down into specific properties and their dimensions. The purpose of identifying the properties and dimensions of each axial code is to define, characterize, examine and re-examine the emerged/emerging axial codes.

Table 5 shows example of some of the properties and dimensions associated with the axial code 'Engaging in Learning through Technology'.

Properties	Sub- properties	Dimensions
Being productive through technology	Gathering information is faster and easier using technology	More efficient less efficient
Improving interaction with opposite gender through use of e-learning 2.0	Has experienced better relations with opposite sex through use of e-learning 2.0	Better relations worse relations

Table 2: Properties and dimensions associated with 'Engaging in Learning through Technology' axis

For example 'being productive through technology' is one of these properties in terms of respondents' experience of learning through technological interactivity in the Australian educational setting and e-learning 2.0. The student respondents have expressed that gathering information is faster and easier using technology, I5-F "*I think the technology of web 2.0 made it very easy for me to know more information and to find anything where I'd like to find it quickly.*" At the same time, the respondent's

experiences with E-learning 2.0 in Australia mainly consisted of being able to discuss topics that were not completely tackled in class more extensively through Web 2.0 tools.

These can show how using technology in general and Web 2.0 tools, in particular, seemed to serve the purpose of increasing and extending class involvement, understanding and productivity in course lessons of these foreign students beyond the time constraints of the typical classroom. However, comparing this opportunity of making use of technology provided in the Australian learning environment with another emerged axial code, such as, 'the language barrier' axis gives better understanding and "strengthens the emerging analysis." (Charmaz, 2012, p. 9). This discussion shows an example of how the researchers (through memos) define the axes, explicates the properties of each axial code, describes its consequences and shows how these axial codes relate to each (2006).

4.6 Constructing an understanding (Theoretical Coding)

As explained by Charmaz and Bryant (2010), this is the stage where the researchers have some freedom to theorise as long as the theorisation incorporates grounded information drawn from the previous stages. In order to carry out theoretical coding, a deeper understanding of the notion of axial codes is required. This is to be done by focusing on the axial codes that arise from analysing the focused codes. As such, in the last stage of analysis, the validated conceptual axial codes are critically and reflectively analysed for their interconnections. From the synthesis of the axial codes, grounded theories are drawn to address each of the research questions posed (Charmaz, 2006, p.83). The results of this final stage that relate to the ultimate goals of this study will be presented at the end of this project.

5 Findings

This section provides a description of these four emergent axial codes/categories. Specific attention will be given to the 'Engaging in learning through technology' axis. This axial code is over-arching, involving the other three axes. The discussion provided is based on the respondents' basic statements that related to each axial code. The checking process of all axial codes is still ongoing and will be continued until the data is saturated.

Key Note: Respondents' statements in all following discussion are identified based on respondents' codes (I1 to I8) and gender (M or F) from the raw transcriptions.

5.1 Overcoming the language barrier

This axial code describes the opinions and experiences of Saudi Arabian students regarding the need for English language skills while studying in Australia, particularly with regard to e-learning 2.0.

The language barrier was found to be a limiting agent that prevented students from making full use of e-learning 2.0. Since e-learning 2.0 typically required them to write in order to interact with other students, they felt conscious about their writing abilities and were generally hesitant about interacting because of fear that their

English would not be good enough. Specifically, I3-M said that he was very careful about what he wrote since *"all the people in that group will read it."* Similarly, I6-F *"I can't explain it (in) English in really good way"...., "when I write on the board I can't proof my English and no one will correct my grammar."*, this explained how difficulties in English made it hard for her to interact online.

5.2 Influence of Cultural Practices

The 'Influence of Cultural Practices' axis describes the respondents' opinions and experiences regarding cultural norms, ways of life, practices or beliefs related to students' choice and adapt to life in Australia. It also describes the differing reactions to mixing with the opposite sex, interaction with the sense of freedom. Finally, this category describes students' practices or beliefs regarding the lack of confidence and shyness in using e-learning 2.0.

The students were found to face various challenges in their stay in Australia. Some of the most prominent had to do with struggling with cultural differences. Specifically, many of the respondents expressed difficulty in adjusting to an environment where males and females were not segregated. As explained by I5-F, *"in Saudi Arabia we are like completely separate"..."I feel like I can't talk, I can't do anything because I wasn't raised to studying like in this environment."* Similar inputs were made by I6-F and I7-F. This difficulty was not limited to females, as explained by I4-M, *"But here we talk with them face to face and sometimes we have to shake hands"*, which was similarly expressed by I1-M and I2-M.

5.3 Adapting to the Australian Education System

The axial code 'Adapting to the Australian Education System' describes the respondents' opinions and experiences about educational models they have experienced, as well as their willingness or otherwise to accept a new educational model through comparing the Australian education system, environment, relationship, learning activities and student rights with KSA and Australia.

A set of challenges revolved around the respondents' perceived differences between the Australian and the Saudi Arabian educational model. As explained by I5-F, *"At the beginning it was very difficult for me to engage with Australian education system."* Specifically, differences pointed out were greater freedom over *"selection of subjects"* and the need to go beyond simply memorising lesson content. Similarly, I6-F discussed how in the Australian setting lectures push students to search to find information, *"In Saudi Arabia we just, ahh, the teachers they just give us information and we just to memorize it, that's all."*

5.4 Engaging in Learning through technology

The 'Engaging in learning through technology' axis discusses the respondents' opinions and their experiences of being more productive through technological interactivity in the Australian educational setting and e-learning 2.0, including previous knowledge about it and

the benefits and challenges of interacting in the new e-learning 2.0 environment.

The data suggests that there were clear contrasts between students' experiences with E-learning 2.0 in Saudi Arabia and in Australia. While some of the respondents did make use of Web 2.0 tools prior to going to Australia, I5-F *"Facebook because I think this is the only thing I used when was in Saudi Arabia before I came to Australia"*, I4-M *"In Saudi Arabia, some of them are used only figure them out and also to communicate with friends, on an informal basis, but for education, No."*, none of them claimed to have used Web 2.0 tools in Saudi Arabia for academic reasons. It was clear that E-learning 2.0 was non-existent in the basic education level for the respondents (I1-M, I2-M, I3-M, I4-M, I5-F, I6-F, I7-F and I8-F).

Some of the students were found to be able to grasp the fundamental concept of E-learning, such as I3-M *"I think E-learning is using anything electronically,...using technology in education in learning whatever this technology"*, and I6-F, *"I think it's about using technology to teach students as much as you can as a teacher"*. However, other students seemed to have less accurate notions of the concept, such as I1-M *"I guess is when we use the university email"*, I8-F *"e-learning...you learn something from a distance"*, and I4-M *"E-learning in general is wherever and whenever."*

On the other hand, none of the respondents showed explicit understanding of what E-learning 2.0 is. Furthermore, there was some confusion found between respondents understanding of E-learning and E-learning 2.0 (I1-M, I2-M, I3-M and I5-F), while some students assuming that any use of technology to learning referred to E-learning 2.0 (I6-F and I8-F).

In contrast to this, other students actually did have the idea of what E-learning 2.0 was, but attributed these ideas to E-learning in general instead of E-learning 2.0 specifically. For example, when asked about their idea regarding E-learning, I2-M said *"I think it how to share information with other"*, but when asked about E-learning 2.0, he said *"I can't make any assumptions about what E-learning 2.0 means, because sometime I got confused"* (L:368). Similarly, I5-F explained E-learning as *"to use the internet or wireless technology or like new devices to gain new knowledge or share some information on something new"*, but about E-learning 2.0 said *"Actually, I'm not sure, could you explain it to me"* (L:164). These inputs show that students' formal understanding of E-learning 2.0 and implicitly, its role in their studies in Australia may be quite limited, which is consistent with literature on the underdeveloped nature of E-learning in Saudi Arabia (Al-Shehri, 2010)..

However, in spite of this supposed limitation, the respondents were able to extensively describe their increased use of Web 2.0 tools in Australia as well as their E-learning 2.0 experiences. Notably, respondents generally described considerably increased usage of Web 2.0 tools in Australia. These tools were used as means of communication with family and friends in Australia and Saudi Arabia, such as stated by I1-M *"Usually, I use them to keep in touch with friends, share information with"*

people I know. For example sharing pictures with friends I know, and keep touch family in Saudi”, and similarly by I3-M, I5-F, I6-F and I7-F. They were also used to socialise by both males, I1-M “It’s good if you looking for new friends”, and females I5-F “I can discuss and meet friends and make a group discussion and all of this from home”, I6-F, I7-F and I8-F.

Different inputs referred to the usage of these tools as enjoyable, I6-F “I use YouTube for fun to see different cultures...to see people from around the world”, I3-M “I find it very fun and helpful when you can contact with them all the day and send messages and you know it is also much cheaper than other way like by phone”, but there were also inputs that evaluated these tools negatively, I7-F “I don’t like Facebook. I have one I created two years ago just to find out what it is about, but I think it’s useless for me.” Despite these varied perspectives about Web 2.0 tools, all of the students seemed to have been exposed considerably to these tools in their academic work in Australia.

These tools have been used as sources of information for assigned research, I8-F “I used YouTube because most of the lecturers they would give us all the time YouTube, they would give us links to open and download software use it at home so most of these tools I use them here.”, I4-M, I2-M, as well as for sharing and discussing lesson content and other course-related information, I1-M “for example last semester we had like an essay group and we have been using Facebook as three people and we used Facebook (group) to share information that we found for our essay group”, I3-M and I8-F. A summary of the Web 2.0 tools used by the respondents in Australia and those that they used in E-learning 2.0 activities is provided in Table 12.

Main Web 2.0	Tools Used for E-learning 2.0	Dominated	Dominantly mentioned
Social network	San Diego, Facebook	Facebook	I1, I2, I3, I4, I5, I7, I8
Microblogging	Tumblr, Twitter, Hictu	Twitter	I1, I2, I4, I5, I7, I8
Media sharing	YouTube, Broadcast	YouTube	I1, I2, I3, I4, I5, I6, I7, I8
Google Apps	Google Docs, Blogger, Google group, Google Search Engine, Google translator.	Google translator	I2, I4, I6
		Google Docs	I4, I6, I8
Online group Mailer	Google and Facebook groups, Uni email groups	Facebook groups	I1, I2, I3, I4, I5, I8
Blogs	Blogger.com	Blogger	I2, I3
Page editing	Wiki.com, Wikipedia	Wiki.com	I2, I4, I5, I6, I7, I8
Forums	Message Board	Message Board	I1, I6, I7, I8
LMSs/ VLEs	Blackboard, Moodle, Ed-Modo	Blackboard	I1, I6, I7, I8
Online call	Skype Meeting	Skype	I8

Table 3: Different types of Web 2.0 used in Australia for learning

Table 4 shows that there was clearly a wide range of Web 2.0 tools that the respondents were exposed to,

highlighting a stark difference between the Saudi Arabian and Australian educational settings that the respondents experienced.

5.5 Synthesis

The data suggests that e-learning 2.0 opportunities acted as tools that enabled the students to bypass the barriers of their Saudi Arabian perspective of education and enabled them to begin to accept the new environment that was being offered by the Australian higher education system. The e-learning 2.0 system enabled students to interact with one another online as a way of getting students to become used to interacting with other people at their university. I1-M stated that “we can discuss with student or with teacher if you want.” I7-F “you can discuss on twitter and that giving you some advice even in your study.” These statements, along with several others I2-M, I3-M, I4-M, I6-F, I8-F, describe how e-learning 2.0 applications that focused on enabling online interaction where students started to test out interacting with other students and began to see that such group interactivity is good and contributes positively to their studies. As explained by I7-F, the e-learning 2.0 activities that she was immersed in enabled her to become better at talking and treating people and discussing matters with the class; it also helped her in making friends.

Further, the data suggests that e-learning 2.0 was found to be especially useful in enabling the Saudi Arabian students to overcome their enculturation to gender segregation. I5-F states that when she began in Australia, she felt shy about dealing with people of the opposite gender, but this eventually passed due to interacting with people of the opposite gender online. This effect is best captured by I8-F “I think, e-learning 2.0 doesn’t care about male or female”...“For example in my group I could share anything, explain anything”...“in class I would feel a bit shy.” Similarly, I4-M also admitted that it was difficult at first to deal with females online, but that it eventually became normal for him.

Previous discussion of the data shows that Saudi Arabian students in Australia were introduced to a very different setting, where teachers were more accommodating, where gender segregation did not exist, and where the learners were at the centre of the learning environment. However, this shift in the environment was difficult for many students, and many of them regarded various differences in cultural and educational environment as challenging.

However, this role of e-learning 2.0 as enabler environments can be compromised by the language barriers that many Saudi Arabian students still have when they enter Australian settings. At the same time, it was also shown how the gender segregation culture that Saudi Arabians have been used to can still find its way in the e-learning 2.0 setting and cause some students to retract from interactivity.

6 Conclusions

The paper provided insights on how Saudi Arabian students studying in Australia engage in e-learning 2.0 environments in Australian higher education. Firstly, it was found that students have a variety of experiences

with E-learning 2.0 in Australia. These include the use of such Web 2.0 tools as Twitter, Facebook, YouTube and wikis. Second, it was found that the e-learning 2.0 environment provides a good way for Saudi Arabian students to ease into the Australian education setting. Through the affordances provided by e-learning 2.0 resources, such as the ability to communicate to classmates and teachers without needing to face them, Saudi Arabian students feel more comfortable with engaging in online interaction, and become better prepared to do so in face to face settings later on. The students generally expressed that E-learning 2.0 gave them the opportunity to learn from their classmates, and made them appreciate the value of peer knowledge.

On the other hand, when we look at emerged factors through the discussion of other axial codes; 'Influence of Cultural Practices' and 'Overcoming the language barrier', two major challenges were identified; gender segregation and language barriers. While e-learning 2.0 led to students participating in a gender-neutral environment, the prevalence of their enculturation on gender segregation hindered some of the students from interacting as actively as they could. The language barrier was also found to be an extra, initial barrier that Saudi Arabian students needed to break through first before they could utilise e-learning 2.0 opportunities to their utmost potential to gain a deeper and wider understanding of their studies.

7 References

- Al-Shehri, A. M. (2010). E-learning in Saudi Arabia: 'To E or not to E, that is the question'. *Journal of family and community medicine*, 17, 147.
- AlMunajjed, M. (1997). *Women in Saudi Arabia Today*, Macmillan.
- Ashrqia Chamber. (2011). *King Abdullah scholarship program seeks to meet market needs of capable national cadres*. <http://www.chamber.org.sa/English/MediaCenter/News/Pages/KingAbdullahscholarshipprogramseekstomeetmarketneedsofcapablenationalcadres.aspx> Accessed 8 April 2011.
- Borau, K., Ullrich, C., Feng, J. & Shen, R. (2009). Microblogging for language learning: Using twitter to train communicative and cultural competence. *Advances in Web Based Learning-ICWL 2009*, 78-87.
- Borgatti, S. (2005). *Introduction to grounded theory*. <http://www.analytictech.com/mb870/introtogt.htm> Accessed 10 May 2012.
- Bradley, D., Noonan, P., Nugent, H. & Scales, B. (2008). Review of Australian higher education discussion paper June 2008.
- Brown & Adler, R. (2008). Open education, the long tail, and learning 2.0. *Educause review*, 43, 16-20.
- Bryant, A. & Charmaz, K. (eds.) (2007). *The SAGE Handbook of Grounded Theory*: SAGE Publications Ltd
- Charmaz, K. (2012). The Power and Potential of Grounded Theory. *A Journal of the BSA MedSoc Group*, 6, 1-15.
- Charmaz., K. (2006). *Constructing grounded theory: A practical guide through qualitative analysis*, Sage Publications Limited.
- Corbin, J. M. & Strauss, A. (1990). Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative sociology*, 13, 3-21.
- Creswell, J. W. (1998). *Qualitative inquiry and research design: choosing among five traditions*, Sage Publications.
- Downes, S. (2007). E-learning 2.0 in development. *Brandon Hall; Innovations in Learning Conference*. San Jose.
- Ehlers, U. D. (2009). Web 2.0-e-learning 2.0-quality 2.0? Quality for new learning cultures. *Quality Assurance in Education*, 17, 296-314.
- Faiola, A. & Matei, S. A. (2010). Enhancing human-computer interaction design education: teaching affordance design for emerging mobile devices. *International Journal of Technology and Design Education*, 20, 239-254.
- Glaser, B. G. (1978). *Theoretical sensitivity: advances in the methodology of grounded theory*, Sociology Press.
- Glaser, B. G. (1992). *Basics of Grounded Theory Analysis: Emergence Vs. Forcing*, Sociology Press.
- Glaser, B. G. (1998). *Doing Grounded Theory: Issues and Discussions*, Sociology Press.
- Glaser, B. G. & Strauss, A. L. (1977). *The discovery of grounded theory: Strategies for qualitative research*, Aldine Publ.
- Harrison, R. (2008). *Teaching in Saudi Arabia: 'I had to unlearn to learn'*. <http://archive.arabnews.com/?page=9§ion=0&article=112075&d=24&m=7&y=2008> Accessed March 17 2012.
- Hoewe, J., Bowe, B. J. & Zeldes, G. A. (2012). Using a Wiki to Produce Journalistic Best Practices. *Communication Teacher*, 26, 22-32.
- Kirkwood, K. (2010). The SNAP Platform: social networking for academic purposes. *Campus-Wide Information Systems*, 27, 118-126.
- Krauskopf, K., Zahn, C. & Hesse, F. W. (2012). Leveraging the affordances of Youtube: The role of pedagogical knowledge and mental models of technology functions for lesson planning with technology. *Computers & Education*, 58, 1194-1206.
- Mahrous, A. A. & Ahmed, A. A. (2010). A Cross-Cultural Investigation of Students' Perceptions of the Effectiveness of Pedagogical Tools. *Journal of Studies in International Education*, 14, 289.
- Marshall, S. (2011). Change, Technology and Higher Education: Are Universities Capable of Organisational Change? *Journal of Asynchronous Learning Networks*, 15, 13.

- Merriam, S. B. (2009). *Qualitative Research: A Guide to Design and Implementation*, Jossey-Bass.
- Ministry of Higher Education in Saudi Arabia. (2012). *The 3rd ieche Brochure 2012*. Riyadh International Convention & Exhibition Center: Riyadh Exhibition Company
<http://www.ieche.com.sa/web/index.php?lang=en>
 Accessed 16 Feb 2012.
- Mirza, A. A. & Al-Abdulkareem, M. (2011). Models of e-learning adopted in the Middle East. *Applied Computing and Informatics*.
- Mitchell, D. R. (2005). *Contextualizing inclusive education: evaluating old and new international perspectives*, RoutledgeFalmer.
- Omli, J. & Wiese-Bjornstal, D. M. (2011). Kids Speak: Preferred Parental Behavior at Youth Sport Events. *Research quarterly for exercise and sport*, 82, 702-711.
- Sbaraini, A., Carter, S. M., Evans, R. W. & Blinkhorn, A. (2011). How to do a grounded theory study: a worked example of a study of dental practices. *BMC medical research methodology*, 11, 128.
- Strauss, A. & Corbin, J. M. (1998). *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, SAGE Publications.
- Strauss, A. L. & Corbin, J. M. (1990). *Basics of qualitative research: grounded theory procedures and techniques*, Sage Publications.
- Tubaishat, A., El-Qawasmeh, E. & Bhatti, A. (2006). ICT experiences in two different Middle Eastern universities. *Issues in Informing Science & Information Technology*, 3, 667-678.
- Young, S. & Pérez, J. (2012). 'We-research': Adopting a wiki to support the processes of collaborative research among a team of international researchers. *International Journal of Music Education*, 30, 3-17.

Author Index

- Bainbridge, David, 57
Butzmann, Lars, 3
- Carbone, Angela, 135
Chattopadhyay, Subhrendu, 81
Chen, Xiliang, 105
Cunningham, Sally Jo, 125
- Darcy, Peter, 75
Delmas, Patrice, 99
- Friggens, David, 49
- Groves, Lindsay, 49
- Hadj-Alouane, Nejib Ben, 89
Hinze, Annika, 57, 125
Hunt, Doug P., 67
- Karmakar, Sushanta, 81
Klapaukh, Roman, 29
Klauck, Stefan, 3
- Lim, Ching Lih, 19
Lutteroth, Christof, 99
- Müller, Stephan, 3
Mammar, Amel, 89
Margono, Hendro, 115
Marshall, Stuart, 29
Mayan, Omar, 135
Meyer, Carsten, 39
- Moffat, Alistair, 19
- Nguyen, Hoang M., 99
- Parry, Dave, iii, vii, 67
Pearce, David J, 29
Plattner, Hasso, 3, 39
- Raikundalia, Gitesh K., 115
Rinck, Michael, 57
- Schliebs, Stefan, 67
Sheard, Judy, 135
Shinn, Tong-Wook, 13
Stantic, Bela, 75
- Takaoka, Tadao, 13
Tempero, Ewan, 105
Thomas, Bruce, iii, vii
Tucker, Steven, 75
- Vanderschantz, Nicholas, 125
- Wünsche, Burkhard C., 99
Wang, Alice Yuchen, 105
Wirth, Anthony, 19
Wust, Johannes, 39
- Yi, Xun, 115
- Zemni, Mohamed Anis, 89

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 124 - Database Technologies 2012

Edited by Rui Zhang, The University of Melbourne, Australia and Yanchun Zhang, Victoria University, Australia. January 2012. 978-1-920682-95-8.

Contains the proceedings of the Twenty-Third Australasian Database Conference (ADC 2012), Melbourne, Australia, 30 January – 3 February 2012.

Volume 125 - Information Security 2012

Edited by Josef Pieprzyk, Macquarie University, Australia and Clark Thomborson, The University of Auckland, New Zealand. January 2012. 978-1-921770-06-7.

Contains the proceedings of the Tenth Australasian Information Security Conference (AISC 2012), Melbourne, Australia, 30 January – 3 February 2012.

Volume 126 - User Interfaces 2012

Edited by Haifeng Shen, Flinders University, Australia and Ross T. Smith, University of South Australia, Australia. January 2012. 978-1-921770-07-4.

Contains the proceedings of the Thirteenth Australasian User Interface Conference (AUI2012), Melbourne, Australia, 30 January – 3 February 2012.

Volume 127 - Parallel and Distributed Computing 2012

Edited by Jinjun Chen, University of Technology, Sydney, Australia and Rajiv Ranjan, CSIRO ICT Centre, Australia. January 2012. 978-1-921770-08-1.

Contains the proceedings of the Tenth Australasian Symposium on Parallel and Distributed Computing (AusPDC 2012), Melbourne, Australia, 30 January – 3 February 2012.

Volume 128 - Theory of Computing 2012

Edited by Julián Mestre, University of Sydney, Australia. January 2012. 978-1-921770-09-8.

Contains the proceedings of the Eighteenth Computing: The Australasian Theory Symposium (CATS 2012), Melbourne, Australia, 30 January – 3 February 2012.

Volume 129 - Health Informatics and Knowledge Management 2012

Edited by Kerryn Butler-Henderson, Curtin University, Australia and Kathleen Gray, University of Melbourne, Australia. January 2012. 978-1-921770-10-4.

Contains the proceedings of the Fifth Australasian Workshop on Health Informatics and Knowledge Management (HIKM 2012), Melbourne, Australia, 30 January – 3 February 2012.

Volume 130 - Conceptual Modelling 2012

Edited by Aditya Ghose, University of Wollongong, Australia and Flavio Ferrarotti, Victoria University of Wellington, New Zealand. January 2012. 978-1-921770-11-1.

Contains the proceedings of the Eighth Asia-Pacific Conference on Conceptual Modelling (APCCM 2012), Melbourne, Australia, 31 January – 3 February 2012.

Volume 133 - Australian System Safety Conference 2011

Edited by Tony Cant, Defence Science and Technology Organisation, Australia. April 2012. 978-1-921770-13-5.

Contains the proceedings of the Australian System Safety Conference (ASSC 2011), Melbourne, Australia, 25th – 27th May 2011.

Volume 134 - Data Mining and Analytics 2012

Edited by Yanchang Zhao, Department of Immigration and Citizenship, Australia, Jiuyong Li, University of South Australia, Paul J. Kennedy, University of Technology, Sydney, Australia and Peter Christen, Australian National University, Australia. December 2012. 978-1-921770-14-2.

Contains the proceedings of the Tenth Australasian Data Mining Conference (AusDM'12), Sydney, Australia, 5–7 December 2012.

Volume 135 - Computer Science 2013

Edited by Bruce Thomas, University of South Australia, Australia. January 2013. 978-1-921770-20-3.

Contains the proceedings of the Thirty-Sixth Australasian Computer Science Conference (ACSC 2013), Adelaide, Australia, 29 January – 1 February 2013.

Volume 136 - Computing Education 2013

Edited by Angela Carbone, Monash University, Australia and Jacqueline Whalley, AUT University, New Zealand. January 2013. 978-1-921770-21-0.

Contains the proceedings of the Fifteenth Australasian Computing Education Conference (ACE 2013), Adelaide, Australia, 29 January – 1 February 2013.

Volume 137 - Database Technologies 2013

Edited by Hua Wang, University of Southern Queensland, Australia and Rui Zhang, University of Melbourne, Australia. January 2013. 978-1-921770-22-7.

Contains the proceedings of the Twenty-Fourth Australasian Database Conference (ADC 2013), Adelaide, Australia, 29 January – 1 February 2013.

Volume 138 - Information Security 2013

Edited by Clark Thomborson, University of Auckland, New Zealand and Udaya Paramalli, University of Melbourne, Australia. January 2013. 978-1-921770-23-4.

Contains the proceedings of the Eleventh Australasian Information Security Conference (AISC 2013), Adelaide, Australia, 29 January – 1 February 2013.

Volume 139 - User Interfaces 2013

Edited by Ross T. Smith, University of South Australia, Australia and Burkhard C. Wünsche, University of Auckland, New Zealand. January 2013. 978-1-921770-24-1.

Contains the proceedings of the Fourteenth Australasian User Interface Conference (AUI2013), Adelaide, Australia, 29 January – 1 February 2013.

Volume 140 - Parallel and Distributed Computing 2013

Edited by Bahman Javadi, University of Western Sydney, Australia and Saurabh Kumar Garg, IBM Research, Australia. January 2013. 978-1-921770-25-8.

Contains the proceedings of the Eleventh Australasian Symposium on Parallel and Distributed Computing (AusPDC 2013), Adelaide, Australia, 29 January – 1 February 2013.

Volume 141 - Theory of Computing 2013

Edited by Anthony Wirth, University of Melbourne, Australia. January 2013. 978-1-921770-26-5.

Contains the proceedings of the Nineteenth Computing: The Australasian Theory Symposium (CATS 2013), Adelaide, Australia, 29 January – 1 February 2013.

Volume 142 - Health Informatics and Knowledge Management 2013

Edited by Kathleen Gray, University of Melbourne, Australia and Andy Koronios, University of South Australia, Australia. January 2013. 978-1-921770-27-2.

Contains the proceedings of the Sixth Australasian Workshop on Health Informatics and Knowledge Management (HIKM 2013), Adelaide, Australia, 29 January – 1 February 2013.

Volume 143 - Conceptual Modelling 2013

Edited by Flavio Ferrarotti, Victoria University of Wellington, New Zealand and Georg Grossmann, University of South Australia, Australia. January 2013. 978-1-921770-28-9.

Contains the proceedings of the Ninth Asia-Pacific Conference on Conceptual Modelling (APCCM 2013), Adelaide, Australia, 29 January – 1 February 2013.

Volume 144 - The Web 2013

Edited by Helen Ashman, University of South Australia, Australia, Quan Z. Sheng, University of Adelaide, Australia and Andrew Trotman, University of Otago, New Zealand. January 2013. 978-1-921770-15-9.

Contains the proceedings of the First Australasian Web Conference (AWC 2013), Adelaide, Australia, 29 January – 1 February 2013.

Volume 145 - Australian System Safety Conference 2012

Edited by Tony Cant, Defence Science and Technology Organisation, Australia. April 2013. 978-1-921770-13-5.

Contains the proceedings of the Australian System Safety Conference (ASSC 2012), Brisbane, Australia, 23rd – 25th May 2012.