Conferences in Research and Practice in
Information Technology

Volume 92

Database Technologies 2009

Australian Computer Science Communications, Volume 31, Number 2

AUSTRALIAN
COMPUTER
SOCIETY

acsw

COmputing
Research
& Education

Database Technologies 2009

Proceedings of the
Twentieth Australasian Database Conference
(ADC 2009), Wellington, New Zealand,
January 2009

Athman Bouguettaya and Xuemin Lin, Eds.

Volume 92 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.

Database Technologies 2009. Proceedings of the Twentieth Australasian Database Conference (ADC 2009), Wellington, New Zealand, January 2009

Conferences in Research and Practice in Information Technology, Volume 92.

Editors:
Athman Bouguettaya
CSIRO ICT Centre
GPO Box 664
Canberra ACT 2601,
Australia
Email: Athman.Bouguettaya@csiro.au

Xuemin Lin
School of Computer Science and Engineering
The University of New South Wales
Sydney, NSW 2052,
Australia
Email: lxue@cse.unsw.edu.au

Series Editors:
Vladimir Estivill-Castro, Griffith University, Queensland
John F. Roddick, Flinders University, South Australia
Simeon Simoff, University of Western Sydney, NSW
crpit@infoeng.flinders.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 92.
ISSN 1445-1336.
ISBN 978-1-920682-73-6.

Printed, January 2009 by Flinders Press, PO Box 2100, Bedford Park, SA 5042, South Australia.
Cover Design by Modern Planet Design, (08) 8340 1361.

The *Conferences in Research and Practice in Information Technology* series aims to disseminate 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 Twentieth Australasian Database Conference (ADC 2009), Wellington, New Zealand, January 2009

Invited Papers

Contributed Papers

Preface

The series of Australasian Database Conference is an annual forum for exploring novel technical developments and applications of database systems. The 20th Australasian Database Conference, ADC 2009, is held in Wellington, New Zealand, as part of Australasian Computer Science Week.

ADC 2009 invited submissions of original contributions in all research areas of databases and its applications. The program committee received forty three submissions of full research papers; each was thoroughly reviewed by at least three PC members or external reviewers. Seventeen papers have been selected for presentation at the conference. In addition, the program committee invited two prominent researchers, Dr. Dimitrios Georgakopoulos (CSIRO) and Dr. Heng Tao Shen (UNSW) for the traditional ADC invited talks.

The ADC PC chairs have also carefully evaluated all accepted papers for the conference's Best Paper Award. This year's Best Paper award goes to two papers. The first paper that won the award is "*Access Control: What is Required in Business Collaboration?*" by *Daisy Daiqin He, Michael Compton, Jian Yang and Kerry Taylor*. The second paper that won the award is "*Score Aggregation Techniques in Retrieval Experimentation*" by *Sri Devi Ravana and Alistair Moffat*. Congratulations to both teams! We are grateful to the EII for donating the prize for the best paper award.

We would like to take this opportunity to thank all the authors who submitted papers and conference participants for the fruitful discussions. We are grateful to the members of the program committee and external referees for their timely expertise and effort in carefully reviewing the papers. We are thankful to Mr. Haichuan Shang and Mr Chuan Xiao from the University of New South Wales for their excellent work on maintaining the conference web site and the paper reviewing system.

Athman Bouguettaya
CSIRO

Xuemin Lin
University of New South Wales

ADC 2009 Programme Chairs
January 2009

Programme Committee

Chairs

Athman Bouguettaya, CSIRO (Australia)
Xuemin Lin, University of New South Wales (Australia)

Members

James Bailey, University of Melbourne (Australia)
Boualem Benatallah, University of New South Wales (Australia)
Elisa Bertino, Purdue University (USA)
Sanjay Chawla, University of Sydney (Australia)
Stijn Dekeyser, University of Southern Queensland (Australia)
Gill Dobbie, University of Auckland (New Zealand)
Annika Hinze, University of Waikato (New Zealand)
Yoshiharu Ishikawa, Nagoya University (Japan)
Mark Cameron, CSIRO, (Australia)
Chen Li, University of California · Irvine (USA)
Xue Li, University of Queensland (Australia)
Tok Wang Ling, National University of Singapore (Singapore)
Sebastian Link, Victoria University of Wellington (New Zealand)
Chengfei Liu, Swinburne University of Technology (Australia)
Qing Liu, CSIRO (Australia)
Jixue Liu, University of South Australia (Australia)
Fred Lochovsky, The Hong Kong University of Science and Technology (Hong Kong SAR, China)
Sebastian Maneth, NICTA (Australia)
Brahim Medjahed, University of Michigan (USA)
Beng Chin Ooi, National University of Singapore (Singapore)
Liam O'Brien, NICTA (Australia)
Mourad Ouzzani, Purdue University (USA)
Helen Paik, University of New South Wales (Australia)
Krithi Ramamrithami, Indian Institute of Technology (India)
Uwe Roehm, University of Sydney (Australia)
Shazia Sadiq, University of Queensland (Australia)
Quan Sheng, University of Adelaide (Australia)
Markus Stumptner, University of South Australia (Australia)
Heng Tao Shen, University of Queensland (Australia)
John Shepherd, University of New South Wales (Australia)
Yufei Tao, Chinese University of Hong Kong, (Hong Kong SAR, China)
Zahir Tari, Royal Melbourne Institute of Technology (Australia)
Paul Thomas, CSIRO (Australia)
Anthony Tung, National University of Singapore (Singapore)
Wei Wang, University of New South Wales (Australia)
Gerald Weber, University of Auckland (New Zealand)
Raymond Wong, University of New South Wales (Australia)
Jian Yang, Macquarie University (Australia)
Jeffrey Yu,, Chinese University of Hong Kong, (Hong Kong SAR, China)
Qi Yu, Rochester Institute of Technology (USA)
Rui Zhang, University of Melbourne (Australia)
Jenny Zhang, Royal Melbourne Institute of Technology (Australia)
Yanchun Zhang, Victoria University (Australia)
Xiaofang Zhou, University of Queensland (Australia)
Aoying Zhou, China East Normal University (China)
Xuan Zhou, CSIRO (Australia)

Additional Reviewers

Mohamad Eunus Ali
Zhifeng Bao
Yueguo Chen
Ryan Choi
Georg Grossmann
Yanan Hao
Mike Ma
Bo Ning
Sebastian Obermeier
Lu Qin
Yacine Sam
Martin Stradling
Evi Syukur
Stijn Vansummeren
Xin Wang
Yanbo Wu
Huayu Wu
Yong Yang
Zhenjie Zhang
Jixue Liu

Organising Committee

Co-Chairs

Dr Alex Potanin
Professor John Hine

Venues

Dr David Pearce

Operations

Dr Peter Komisarczuk
Mrs Suzan Hall
Mr Craig Anslow

Finance and Program

Dr Stuart Marshall

Communications

Dr Ian Welch
Mr Craig Anslow

Events

Professor John Hine

Welcome from the Organising Committee

We would like to welcome you to ACSW2009 hosted by Victoria University of Wellington, New Zealand.

Wellington is set on the edge of a stunning harbour and surrounded by rolling hills. The earliest name for Wellington, from Maori legend, is Te Upoko o te Ika a Maui. In Maori it means the head of Maui's fish. Caught and pulled to the surface by the Polynesian navigator Maui, the fish became the North Island. Wellington is the capital city of New Zealand and home to the seat of parliament. But this vibrant and dynamic city also has many other capital claims including Culture capital, Creative capital and Events capital. It is a compact, walkable city waiting to be explored. The conference venue is less than fifteen minutes walk to accommodation, Courtenay Place with its wide range of bars, and the harbour with its restaurants and activities such as sea kayaking. The conference venue itself is in the Museum of New Zealand Te Papa Tongarewa, offering visitors a unique and authentic experience of this country's treasures and stories. Over five floors, you can explore the nation's nature, art, history, and heritage - from the shaping of its land to the spirit of its diverse peoples, from its unique wildlife to its distinctive art and visual culture.

Victoria University of Wellington - Te Whare Wānanga o te Ūpoko o te Ika a Māui - is over a century old. Victoria College was founded through an Act of Parliament in 1897, the year of Queen Victoria's Diamond Jubilee celebrations, and named in her honour. Victoria is a thriving community of almost 25,000 people. Situated in the capital city across four campuses, Victoria can take advantage of connections and values its relationships with iwi, business, government, the judiciary, public and private research organisations, cultural organisations and resources, other universities and tertiary providers and the international community through the diplomatic corps. ACSW2009 coincides with the opening of the new School of Engineering and Computer Science as part of the Faculty of Engineering at Victoria University of Wellington - combining a long history of research and teaching of the software engineering and network engineering in the Computer Science department and computer system engineering and electronic engineering in the Physics department. Professor John Hine, co-chairing ACSW2009, is the current Dean of Engineering and the inaugural Head of School of Engineering and Computer Science.

ACSW2009 consists of the following conferences:

- Australasian Computer Science Conference (ACSC) (Chaired by Bernard Mans),
- Australasian Computing Education Conference (ACE) (Chaired by Margaret Hamilton and Tony Clear),
- Australasian Database Conference (ADC) (Chaired by Athman Bouguettaya and Xuemin Lin),
- Australasian Symposium on Grid Computing and e-Research (AUSGRID) (Chaired by Wayne Kelly and Paul Roe),
- Computing: The Australasian Theory Symposium (CATS) (Chaired by Prabhu Manyem and Rod Downey),
- Asia-Pacific Conference on Conceptual Modelling (APCCM) (Chaired by Markus Kirchberg and Sebastian Link),
- Australasian Information Security Conference (AISC) (Chaired by Ljiljana Brankovic and Willy Susilo),
- Australasian Workshop on Health Informatics and Knowledge Management (HIKM) (Chaired by Jim Warren),
- Australasian User Interface Conference (AUIC) (Chaired by Gerald Weber and Paul Calder),
- Australasian Computing Doctoral Consortium (ACDC) (Chaired by David Pearce and Vladimir Estivill-Castro).

The nature of ACSW requires the co-operation of numerous people. We would like to thank all those who have worked to ensure the success of ACSW2009 including the Organising Committee, the Conference Chairs and Programme Committees, our sponsors, the keynote speakers and the delegates.

Dr Alex Potanin and Professor John Hine
ACSW2009 Co-Chairs
Victoria University of Wellington
January, 2009

CORE - Computing Research & Education

CORE welcomes all delegates to ACSW2009 in Wellington. CORE, the peak body representing academic computer science in Australia and New Zealand, is responsible for the annual ACSW series of meetings, which are a unique opportunity for our community to network and to discuss research and topics of mutual interest. The original component conferences – ACSC, ADC, and CATS, which formed the basis of ACSW in the mid 1990s – now share the week with seven other events, which build on the diversity of the Australasian CS community.

This year, we have chosen to feature a small number of plenary speakers chosen from across the discipline, Ronald Fagin, Ian Foster, Mark Guzdial, and Andy Hopper. I thank them for their contributions to ACSW'09.The efforts of the conference chairs and their program committees have led to strong programs in all the conferences – again, thanks. And thanks are particularly due to Alex Potanin, John Hine, and their colleagues for organising what promises to be a memorable ACSW.

In Australia, 2008 has been a busy year for academia, with the incoming Labor government instituting major reviews in areas such as the higher education sector, research funding, postgraduate study, and national curricula. However, while the reviews have exposed severe shortcomings in the funding of higher education and research, they have not as yet been translated into definite action, and the sector as a whole is shrinking. Although there is a widespread perception of a shortage of IT staff, and graduate salaries remain strong, student interest in ICT continues to be low. Moreover, per-place funding for computer science students has dropped relative to that of other physical and mathematical sciences. Several forums and initiatives involving industry, government, and academia have attempted to address the issue of the ongoing difficulties of attracting students to the discipline, but with little perceptible effect. New initiatives that seek to address the issues of students and funding will be a CORE priority in 2009.

During 2008, CORE continued to work on journal and conference rankings, with much of the activity driven by requests for information from the government. A key aim is now to maintain the rankings, which are widely used overseas as well as in Australia, a challenging process that needs to balance competing special interests as well as addressing the interests of the community as a whole. A new activity in 2008 was a review of computing curriculum, which is still ongoing, with the intention that a CORE curriculum statement be used for accreditation of degrees in computer science, software engineering, and information technology. ACSW'09 includes a forum on computing curriculum to discuss this process.

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 2008; in particular, I thank Jenny Edwards, Alan Fekete, Tom Gedeon, Leon Sterling, Vanessa Teague, and the members of the executive and of the curriculum and ranking committees.

Justin Zobel
President, CORE
January, 2009

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.

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

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

2008. Volume 30. Host and Venue - University of Wollongong, NSW.
2007. Volume 29. Host and Venue - University of Ballarat, VIC. First running of HDKM.
2006. Volume 28. Host and Venue - University of Tasmania, TAS.
2005. Volume 27. Host - University of Newcastle, NSW. APBC held separately from 2005.
2004. Volume 26. Host and Venue - University of Otago, Dunedin, New Zealand. First running of APCCM.
2003. Volume 25. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Adelaide Convention Centre, Adelaide, SA. First running of APBC. Incorporation of ACE. ACSAC held separately from 2003.
2002. Volume 24. Host and Venue - Monash University, Melbourne, VIC.
2001. Volume 23. Hosts - Bond University and Griffith University (Gold Coast). Venue - Gold Coast, QLD.
2000. Volume 22. Hosts - Australian National University and University of Canberra. Venue - ANU, Canberra, ACT. First running of AUIC.
1999. Volume 21. Host and Venue - University of Auckland, New Zealand.
1998. Volume 20. Hosts - University of Western Australia, Murdoch University, Edith Cowan University and Curtin University. Venue - Perth, WA.
1997. Volume 19. Hosts - Macquarie University and University of Technology, Sydney. Venue - Sydney, NSW. ADC held with DASFAA (rather than ACSW) in 1997.
1996. Volume 18. Host - University of Melbourne and RMIT University. Venue - Melbourne, Australia. CATS joins ACSW.
1995. Volume 17. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Glenelg, SA.
1994. Volume 16. Host and Venue - University of Canterbury, Christchurch, New Zealand. CATS run for the first time separately in Sydney.
1993. Volume 15. Hosts - Griffith University and Queensland University of Technology. Venue - Nathan, QLD.
1992. Volume 14. Host and Venue - University of Tasmania, TAS. (ADC held separately at La Trobe University).
1991. Volume 13. Host and Venue - University of New South Wales, NSW.
1990. Volume 12. Host and Venue - Monash University, Melbourne, VIC. Joined by Database and Information Systems Conference which in 1992 became ADC (which stayed with ACSW) and ACIS (which now operates independently).
1989. Volume 11. Host and Venue - University of Wollongong, NSW.
1988. Volume 10. Host and Venue - University of Queensland, QLD.
1987. Volume 9. Host and Venue - Deakin University, VIC.
1986. Volume 8. Host and Venue - Australian National University, Canberra, ACT.
1985. Volume 7. Hosts - University of Melbourne and Monash University. Venue - Melbourne, VIC.
1984. Volume 6. Host and Venue - University of Adelaide, SA.
1983. Volume 5. Host and Venue - University of Sydney, NSW.
1982. Volume 4. Host and Venue - University of Western Australia, WA.
1981. Volume 3. Host and Venue - University of Queensland, QLD.
1980. Volume 2. Host and Venue - Australian National University, Canberra, ACT.
1979. Volume 1. Host and Venue - University of Tasmania, TAS.
1978. Volume 0. Host and Venue - University of New South Wales, NSW.

Conference Acronyms

ACE. Australian/Australasian Computing Education Conference.
ACSAC. Asia-Pacific Computer Systems Architecture Conference (previously Australian Computer Architecture Conference (ACAC).
ACSC. Australian/Australasian Computer Science Conference.
ACSW. Australian/Australasian Computer Science Week.
ADC. Australian/Australasian Database Conference.
AISW. Australasian Information Security Workshop.
APBC. Asia-Pacific Bioinformatics Conference.
APCCM. Asia-Pacific Conference on Conceptual Modelling.
AUIC. Australian/Australasian User Interface Conference.
AusGrid. Australasian Workshop on Grid Computing and e-Research.
CATS. Computing - The Australian/Australasian Theory Symposium.
HDKM. Australasian Workshop on Health Data and Knowledge Management.
HIKM. Australasian Workshop on Health Informatics and Knowledge Management (former HDKM).

Note that various name changes have occurred, most notably the change of the names of conferences to reflect a wider geographical area.

ACSW and ADC 2009 Sponsors

We wish to thank the following sponsors for their contribution towards this conference. For an up-to-date overview of sponsors of ACSW 2009 and ADC 2009, please see `http://www.mcs.vuw.ac.nz/Events/ACSW2009/Sponsors`.

CityLink, New Zealand,
www.citylink.co.nz

New Zealand Computer Society,
www.nzcs.org.nz

Victoria University of Wellington,
www.victoria.ac.nz

Australian Computer Society,
www.acs.org.au

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

Xero,
www.xero.com

Security Assessment, New Zealand,
www.security-assessment.com

Catalyst, New Zealand,
www.catalyst.net.nz

Helium, New Zealand,
www.heliumnz.co.nz

ARC Research Network in Enterprise Information Infrastructure,
www.eii.edu.au

CSIRO,
www.csiro.au

School of Computer Science and Engineering,
www.cse.unsw.edu.au

Invited Papers

Large-scale Video Sequence Indexing: Impacts, Ideas and Trends

Heng Tao Shen
School of Information Technology and Electrical Engineering
The University of Queensland
QLD 4072, Australia

Shenht@itee.uq.edu.au

Abstract

With the advances of hardware (e.g., wide availability of Webcam) and software (e.g., video editing or instant messaging software), the amount of video data has grown rapidly in many fields, such as broadcasting, advertising, filming, personal video archive, and medical/scientific video repository. In addition, Web has generated enormous impact by popularizing video publishing and sharing (e.g., social networking websites). Online delivery of video content has surged to an unprecedented level. The wide availability of video data fuels many novel applications, such as near-duplicate video detection, in-video advertising, video recommendation, web video search, etc. With these demanding applications, how to manage large-scale video databases and search similar video content is of uttermost importance. Although content-based video search has recently attracted plenty of attention, the high complexity of video data, coupled with large volume, poses huge challenges towards large-scale video sequence search. As the volume of video data continues to grow rapidly, the demand of efficient indexing on large-scale video databases from database community is increasingly imperative.

In this talk, we will look at the problem of effective indexing supports for large-scale video sequence search in various forms, such as clip matching, subsequence matching and continuous stream matching. Its impacts and challenges will be discussed, followed by our recent ideas and developments to tackle this problem. Its future trends in next age will also be discussed.

Keywords: video search, video database, indexing, sequence matching, similarity measure, query processing.

Engineering Agile Systems

Dimitrios Georgakopoulos
CSIRO ICT Centre
GPO Box 664, Canberra ACT 2601, Australia
dimitrios.georgakopoulos@csiro.au

Abstract

The majority of today's software systems and organizational/business structures have been built on the foundation of solving problems via long-term data collection, analysis, and solution design. This traditional approach of solving problems and building corresponding software systems and business processes, falls short in providing the necessary solutions needed to deal with many problems that require agility as the main ingredient of their solution. For example, such agility is needed in responding to an emergency, in military command control, physical security, price-based competition in business, investing in the stock market, video gaming, network monitoring and self-healing, diagnosis in emergency health care, and many other areas that are too numerous to list here. The concept of Observe, Orient, Decide, and Act (OODA) loops is a guiding principal that captures the fundamental issues and approach for engineering agile information systems that deal with many of these problem areas. However, there are currently few software systems that are capable of supporting OODA. In this talk, we advocate a combination of complex event processing, service computing, and OODA principles for building agile systems, and provide a tour of corresponding research issues and state of the art solutions. We also provide specific examples of agile systems from the video surveillance, emergency response, and intelligence gathering domains.

Keywords: Service computing, complex event processing, OODA.

Contributed Papers

The Effect of Sparsity on Collaborative Filtering Metrics

Jesus Bobadilla and Francisco Serradilla
Universidad Politecnica de Madrid
Crta. De Valencia, Km 7, 28031 Madrid, Spain

jbobi@eui.upm.es, fserra@eui.upm.es

Abstract

This paper presents a detailed study of the behavior of three different content-based collaborative filtering metrics (correlation, cosine and mean squared difference) when they are processed on several ratio matrices with different levels of sparsity. The total number of experiments carried out is 648, in which the following parameters are varied: metric used, number of k-neighborhoods, sparsity level and type of result (mean absolute error, percentage of incorrect predictions, percentage of correct predictions and capacity to generate predictions). The results are illustrated in two and three-dimensional representative graphs. The conclusions of the paper emphasize the superiority of the correlation metric over the cosine metric, and the unusually good results of the mean squared difference metric when used on matrices with high sparsity levels, leading us to interesting future studies.

Keywords: recommender systems, sparsity, collaborative filtering, metric

1 Introduction

At present, Recommender Systems (RS), are broadly used to implement Web 2.0 services (Janner 2007) as mentioned by Knights and Lin (2007), based on Collaborative Filtering (CF). RS make predictions about the preferences of each user based on the preferences of a set of "similar" users.

This way, a trip to Canary Islands could be recommended to an individual who has rated different destinations in the Caribbean very highly, based on the positive ratings about the holiday destination of "Canary Islands" of an important number of individuals who also rated destinations in the Caribbean very highly.

There are a large number of applications based on RS (Jinghua 2007, Baraglia 2004, and Fesenmaier 2002), some of which are centered on the movie recommendation area (Konstan 2004, Antonopoulus 2006, Li 2005).

The quality of the results offered by a RS greatly depends on the quality of the results provided by its CF (Adomavicius 2005, Herlocker 2004) phase; i.e. it is essential to be capable of adequately selecting the group of users most similar to a given individual.

The similarity among users can be computed in three different ways: content-based methods, model-based methods and hybrid approaches. Content-based methods (Breese 1998, Kong 2005) use similarity metrics (Herlocker 2004) which operate directly on the individual user's ratios (in the trip recommender example, that is each value voted for each travel destination). Model-based methods (Breese 1998) use user ratios to create a computable model (Bayesian classifier (Cho 2007), neural network (Ingoo 2003), fuzzy system [16], etc.) and from this model they predict the clusters of similar users.

At present, for reasons of predictability and efficiency, commercial RS (Linden 2003) are implemented using content-based CF metrics. Model-based CF can usually be found in non-commercial research phases.

The majority of CF research aims to increase the accuracy and coverage (Giaglis 2006, Li 2005, Fuyuki 2006, and Manolopoulus 2007); nevertheless, it is advisable to improve certain other factors: effectiveness of recommendations, searching for good items, credibility of recommendations, precision and recall measures, etc.).

Memory-based methods work on two-dimensional matrices of U users who have rated a number of items I. We can consider a RS running in an e-travel agency, where, over the years, thousands of travelers have rated hundreds of destinations, for example.

An important problem in obtaining effective predictions using RS is the fact that most of the users only rate a very small proportion of the items; this is known as the sparsity problem. When the matrix is very sparse, it means there are many users who have rated very few items and this leads to two main negative effects:

- The set of similar users (k-neighborhoods) (Herlocker 2002) does not suitably match the preferences of the recommended user (there are not enough common rated items to establish a reliable similarity result between two users).
- It is not easy to recommend items to the user, as you are not likely to find enough k-neighborhoods who had rated the same items positively.

Consequently, the accuracy and the majority of the main effectiveness measures of the CF predictions drop when they are applied to extremely sparse matrices, leading to the users losing confidence in the RS service as a whole.

The sparsity problem has traditionally been tackled using user profile information to reinforce the similarity measure. The CF techniques called demographic filtering (Pazzani 1999) use all the possible additional information to establish the similarity among users such as gender, age, education, area code, etc.

Another approach in order to reduce the sparsity problem is the use of a dimensionality reduction technique such as Singular Value Decomposition (SVD) (Sarwar 2000).

The demographic filtering approach has two important restrictions:

- More often than not there is no demographic information (or not enough demographic information) in the RS database.
- Establishing similarities based on demographic information is very risky and can easily lead to incorrect recommendations.

The dimensionality reduction approach removes unrepresentative users or items. At present some research works use statistical techniques such as Principle Component Analysis (PCA) (Goldbergh 2001) and information retrieval techniques such as Latent Semantic Indexing (LSI) (Deerwester 1990, and Hofmann 2003). The main problem with the reduction approach is the inherent loss of information in the reduction process.

Alternatively, other approaches exist with which to deal with the sparsity problem, such as the use of trust inferences (Papagelis 2005), attraction-weighted information filtering (Bruyn 2004) and topographic organization of user preferences patterns (Polcicova 2004).

2 Content-Based Metrics

Content-based methods work on a table of *U* users who have rated a number of items *I*. The prediction of a non-rated item *i* for a user *u* is computed as an aggregate of the ratings of the *K* most similar users (k-neighborhoods) for the same item *i*, where $\tilde{K}$ denotes the set of k-neighborhoods.

The most common aggregation approaches are the average (1) and the weighted sum (2).

$$r_{u,i} = \frac{1}{|\tilde{K}|} \sum_{k \in \tilde{K}} r_{k,i} \qquad (1)$$

$$r_{u,i} = \mu \sum_{k \in \tilde{K}} sim(u,k)\, r_{k,i} \qquad (2)$$

Where μ acts as a normalizing factor, usually computed as:

$$\mu = 1 \Big/ \sum_{k \in \tilde{K}} sim(u,k) \qquad (3)$$

The similarity approaches usually compute the similarity between two users *x* and *y: sim(x,y)* based on their ratings of items that both users have rated (4).

$$i \in I \mid r_{x,i} \neq \phi \text{ and } r_{y,i} \neq \phi \qquad (4)$$

The most popular similarity metrics are Pearson correlation (5) and cosine (6), although we will complete the experiments in this paper by adding the least known Mean Squared Difference (MSD) metric (7).

$$sim(x,y) = \frac{\sum_i (r_{x,i} - \bar{r}_x)(r_{y,i} - \bar{r}_y)}{\sqrt{\sum_i (r_{x,i} - \bar{r}_x)^2 \sum_i (r_{y,i} - \bar{r}_y)^2}} \qquad (5)$$

$$sim(x,y) = \frac{\sum_i r_{x,i}\, r_{y,i}}{\sqrt{\sum_i r^2{}_{x,i}}\sqrt{\sum_i r^2{}_{y,i}}} \qquad (6)$$

$$sim(x,y) = \frac{1}{I} \sum_{i=1}^{I} (r_{x,y} - r_{y,i})^2 \qquad (7)$$

The research work shown in this paper is based on comparative experiments using Pearson (5), cosine (6) and MSD (7) metrics, the average aggregation approach (1), and the Mean Absolute Error (8).

MSD has been selected due to its unique behavior, which is very different to the correlation and cosine metrics, mainly when it is used in sparse ratio matrices.

3 Design of Experiments

In order to discover the behavior of each of the three metrics analyzed, we used the *MovieLens* database [1] as a base, which has been a reference for many years in research carried out in the area of CF.

The database contains 943 users, 1682 items and 100,000 ratings, with a minimum of 20 items rated per user. The items represent cinema films and the rating ranges vary from 1 to 5 stars.

In all the experiments carried out, for each film that each user has rated, the average value of the ratios given by their k-neighborhoods for that film has been calculated and the prediction has been compared with the value rated by the user, thus obtaining the calculation of the mean absolute error (MAE) [8].

[1] Our acknowledgements to the GroupLens Research Group

$$\left|\overline{E}_{user}\right| = \frac{\sum_{i=1}^{I}\left|\frac{1}{\tilde{K}}\sum_{k\in\tilde{K}} r_{k,i} - r_{user,i}\right|}{I} \qquad (8)$$

The previous process was carried out for each of the following k-neighborhoods values: 15, 30, 60, 90, 120, 150, 180, 210 and 240, covering from 1.6% to 25.4% of the total number of users.

In order to obtain comparable results based on different sparsity levels, we have made several reductions on the original database containing 100,000 ratings; each reduction process has removed a fixed number of database ratios using a random function. In this way, we have obtained five additional databases: 80,000 ratings, 60,000 ratings, 40,000 ratings, 20,000 ratings and 10,000 ratings.

The MovieLens original database presents a 100-10^7/(943*1682) percentage of sparsity, the 80,000 database presents a 100-8*10^6/(943*1682) percentage of sparsity, and so on. Therefore, the sparsity range covered in the experiments is: 93.7%, 94.96%, 96.22%, 97.48%, 98.74% and 99.37%.

In turn, all of these calculations have been carried out 3 times (one for each metric included in the study).

The total number of experiments carried out is 648 (9 k-neighborhoods * 6 sparsity levels * 3 metrics * 4 types of results).

The experiments have been grouped in such a way that the following can be determined:

- Accuracy.
- Number of recommendations made.
- Number of perfect predictions.
- Number of bad predictions.

We consider a perfect prediction to be each situation in which the prediction of the number of stars recommended for one user in one film matches the value rated by that user for that film.

We consider a bad prediction to be each situation in which the prediction of the number of stars recommended for one user in one film is different by more than 2 stars from the value rated by that user for that film.

We consider a recommendation made to be each situation in which a user has rated an item and at least one of the user's k-neighborhoods has also rated it, in such a way that a prediction could be made and an MAE error obtained.

4 Results

The results section has been divided into six subsections: the first one shows comparatives of the three metrics studied, processed using different levels of sparsity; in this case no detailed information on k-neighborhoods is included as each result (each graph) has been obtained by calculating the average of the individual results of all nine (15 to 240) k-neighborhoods.

The remaining three subsections refer to each one of the three metrics, respectively, and they contain all the detailed information obtained when processing all the possible variations: range of k-neighborhoods / range of sparsity levels.

The last subsections illustrate the details obtained by comparing the correlation metric with the cosine and the MSD metrics.

4. 1 Comparison of CF Metrics using different sparsity levels

The first results presented here refer to accuracy, processed using the Mean Absolute Error. The x-axis represents the different percentages of sparsity.

Figure 1 shows better results with the correlation metric than with the cosine metric. In fact, there is an improvement in the correlation results, in contrast to the cosine results, where the error increases as the sparsity percentage increases.

The Mean Squared Difference metric shows much better results than the cosine and correlation metrics; nevertheless it is necessary to adjust this good result with the very poor behavior obtained in Figure 2.

The MAE values indicate the mean absolute difference between predictions and real rated values, consequently, a value of 0.5 on the MAE axis means a half-star error in the values predicted from the 'MovieLens' database.

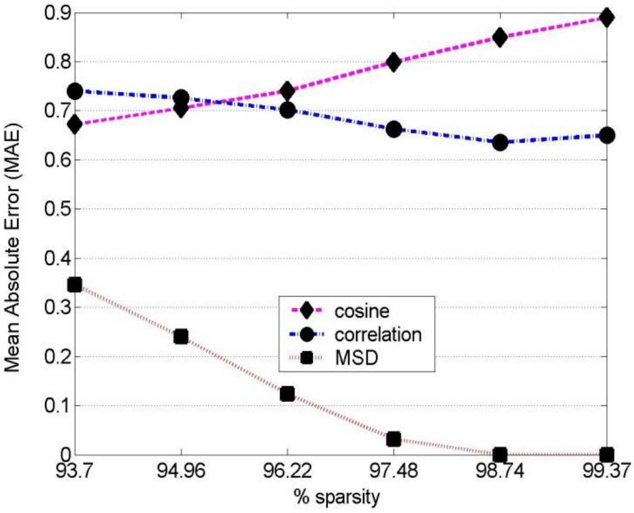


Figure 1. Mean Absolute Error

Figure 2 shows the percentage of recommendations that each metric is able to produce. These percentages are obtained by dividing each number of predictions obtained by computing the different levels of sparsity by the number of ratios of each database (10,000, 80,000, etc.), for example, using the 10,000-ratio database (99.37% of sparsity) the cosine metric was able to compute an average of 8688 predictions, thus the last diamond position in Figure 2 has the value 86.88%.

The correlation metric once again shows better behavior than the cosine metric as it is able to obtain a larger number of recommendations; nevertheless, as expected, the amount of predictions decreases as the sparsity level increases (it is more difficult to obtain recommendations when the quantity of information available to make predictions decreases).

The rising section of the cosine function in Figure 2 can be explained by the erroneous behavior of the cosine metric when the sparsity of the vectors is too high, which is the case when the sparsity of the database is high.

The MSD metric provides very poor results as it obtains a low quantity of predictions. This is this metric's Achilles' heel and is the aspect that should be improved in any metric derived from it.

Figure 3 shows that the correlation metric is able to achieve a greater number of prediction hits than the cosine metric. Whereas the cosine hits drop in line with the sparsity level, the correlation metric even manages to improve its results when the percentage of sparsity is high.

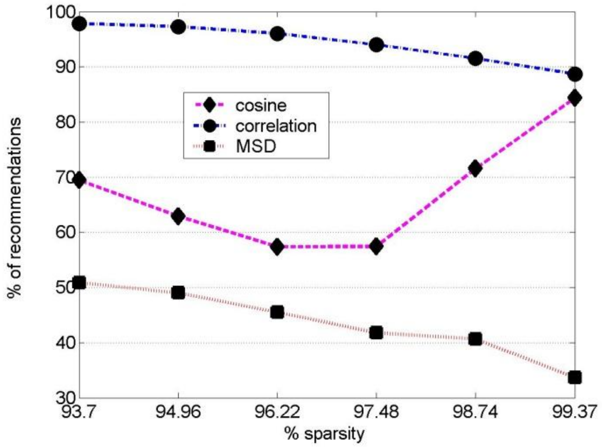


Figure 2. Percentage of recommendations made

It is important to realize that the cosine metric improves the MAE and the percentage of perfect prediction results compared to the correlation metric when the sparsity percentage is not very high (database of 100,000 ratings).

The MSD exhibits excellent behavior when the sparsity levels are high; nevertheless, it is important to realize that the overspecialization effect (recommending items that are too well-known) can be easily produced.

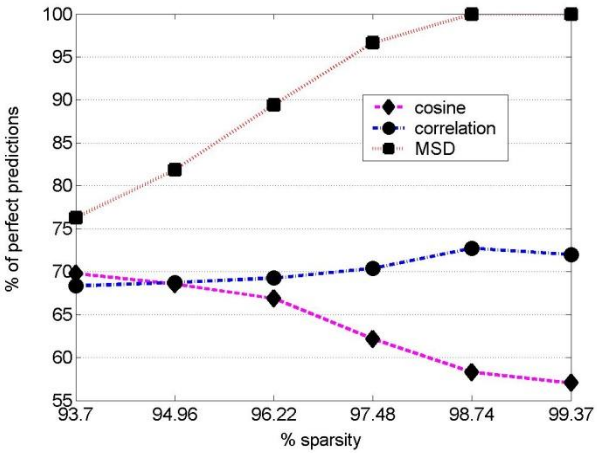


Figure 3. Percentage of perfect predictions

A very important objective of CF metrics is to avoid incorrect recommendations, to prevent users from losing confidence in the system.

Figure 4 presents the percentage of incorrect recommendations (more than two stars of difference between predictions and real ratios). As we can see, the cosine metric does not respond well to an increase in sparsity, whereas the correlation metric responds well. The MSD metric does not produce a high quantity of predictions (Figure 2), but it appears to achieve a good number of hits with its recommendations (Figures 3 and 4), particularly when the sparsity levels are high.

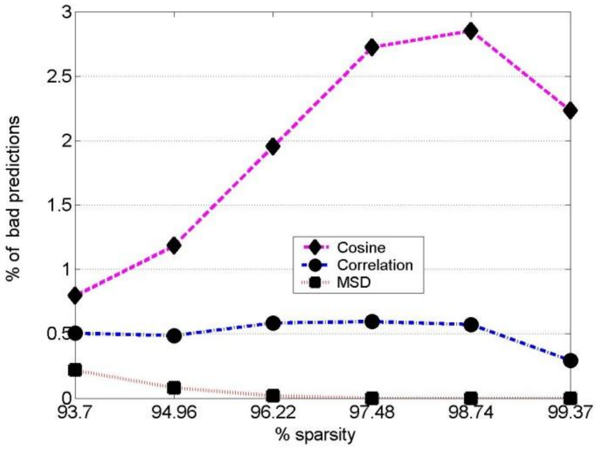


Figure 4. Percentage of bad predictions

4.2 Correlation Metric

This section shows the detailed results obtained from the correlation metric experiments. As in the previous section, the aspects of study are: MAE accuracy, percentage of recommendations made, percentage of perfect predictions and percentage of bad predictions.

Each result is presented as a three-dimensional graph where the *x-axis* represents the number of k-neighborhoods computed in each experiment and the *z-axis* represents the percentage of sparsity (i.e. the 100,000, 80,000, … databases used).

Figure 5a) shows an even decline of the MAE when the sparsity percentage increases (as we saw in Figure 1). In this case we can observe that correlation works better when the number of k-neighborhoods is not low.

Figure 5b) shows the poorest results when the number of k-neighborhoods is low (less than 60). The evolution presented in Figure 2 would improve by selecting more than 60 k-neighborhoods. The same is true when the correlation metric obtains perfect predictions (Figure 5c) and bad predictions (Figure 5d). As a result of this, we can highlight the good results obtained by this metric, especially when the number of neighborhoods is not low and the sparsity level is high.

Figure 5c) shows how the rising correlation slope presented in Figure 3 can be enhanced by selecting a number of k-neighborhoods higher than 120.

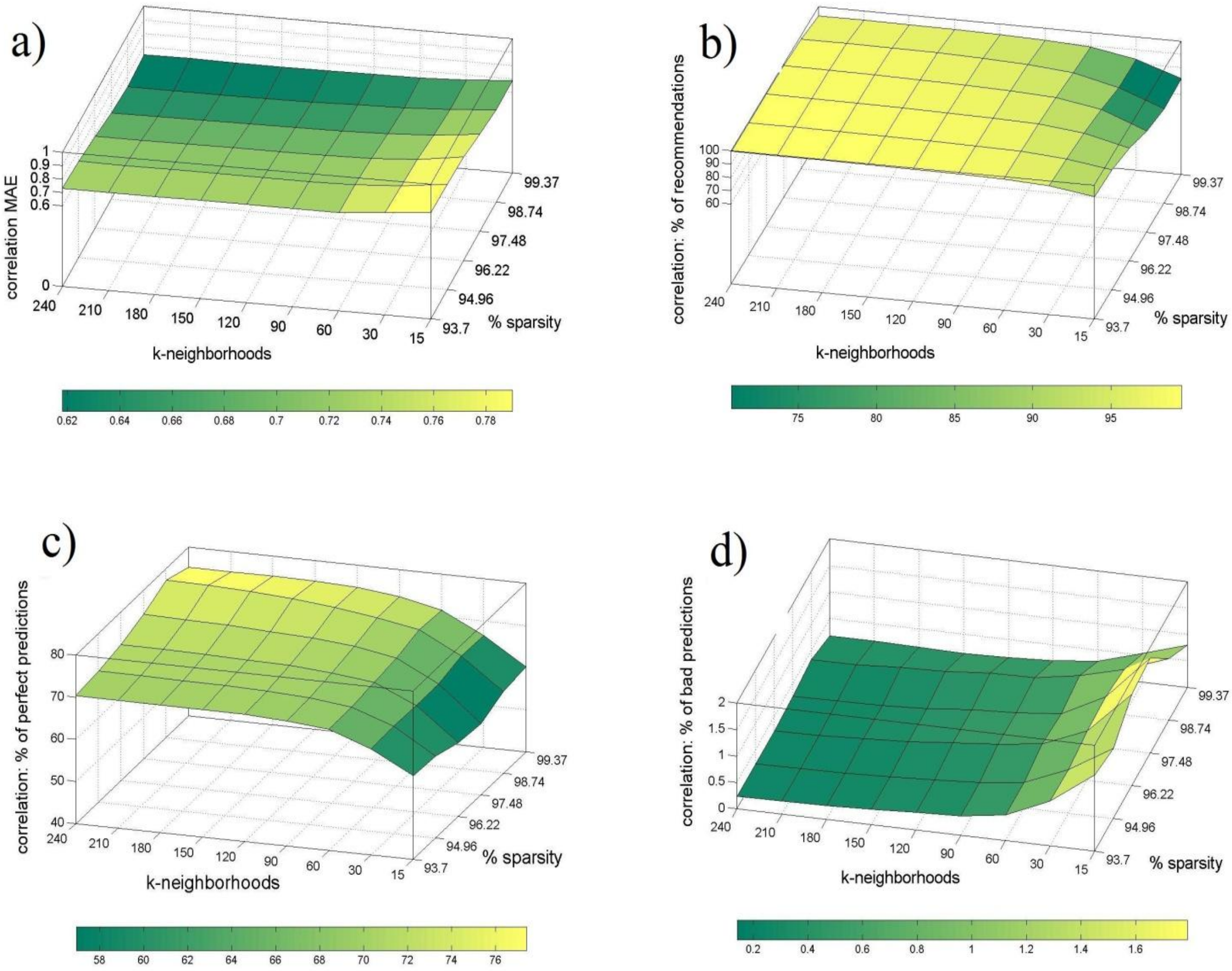


Figure 5. Results of the correlation metric: a) Mean Absolute Error, b) percentage of recommendations made, c) percentage of perfect predictions, d) percentage of bad predictions

4.3 Cosine Metric

Although it was previously established that the correlation metric presented better behavior than the cosine metric, it is relevant to point out the details of the cosine, which is much less regular than the Pearson metric.

In general, it can be said that the cosine metric works better when the sparsity level is low and the number of k-neighborhoods is high. This fact can be observed in Figure 6, where the best results are given in the quadrant: k-neighborhoods from 150 to 240 and sparsity from 93.7 to 96.22.

Figure 6a) shows the best results when the values are smaller (from 0.65 to 0.75); the same situation is presented in Figure 6d) (from 0.5 to 1.5). Figures 6b) and 6c) give the best results when the values are larger (from 80 to 100 and from 68 to 78, respectively).

By studying Figures 6a) to 6d) (cosine) we can observe that the slopes of the surfaces are higher than those corresponding to Figures 5a) to 5d) (correlation), both on the sparsity axis and the k-neighborhoods axis (when k>60); this means that the influence of both parameters is higher in the cosine metric.

4.4 Mean Squared Differences Metric

The results obtained by applying the MSD metric are significantly different to those obtained by the cosine and correlation metrics. The mean absolute error (Figure 7a) presents very low (good) values for all the k-neighborhoods and the percentage of sparsity ranges. We can observe that the best results (lowest errors) are obtained by selecting the lowest k-neighborhood values and, particularly, when the sparsity percentage is high.

There can be no doubt that the weak point of the MSD metric is its poor capacity to generate a large

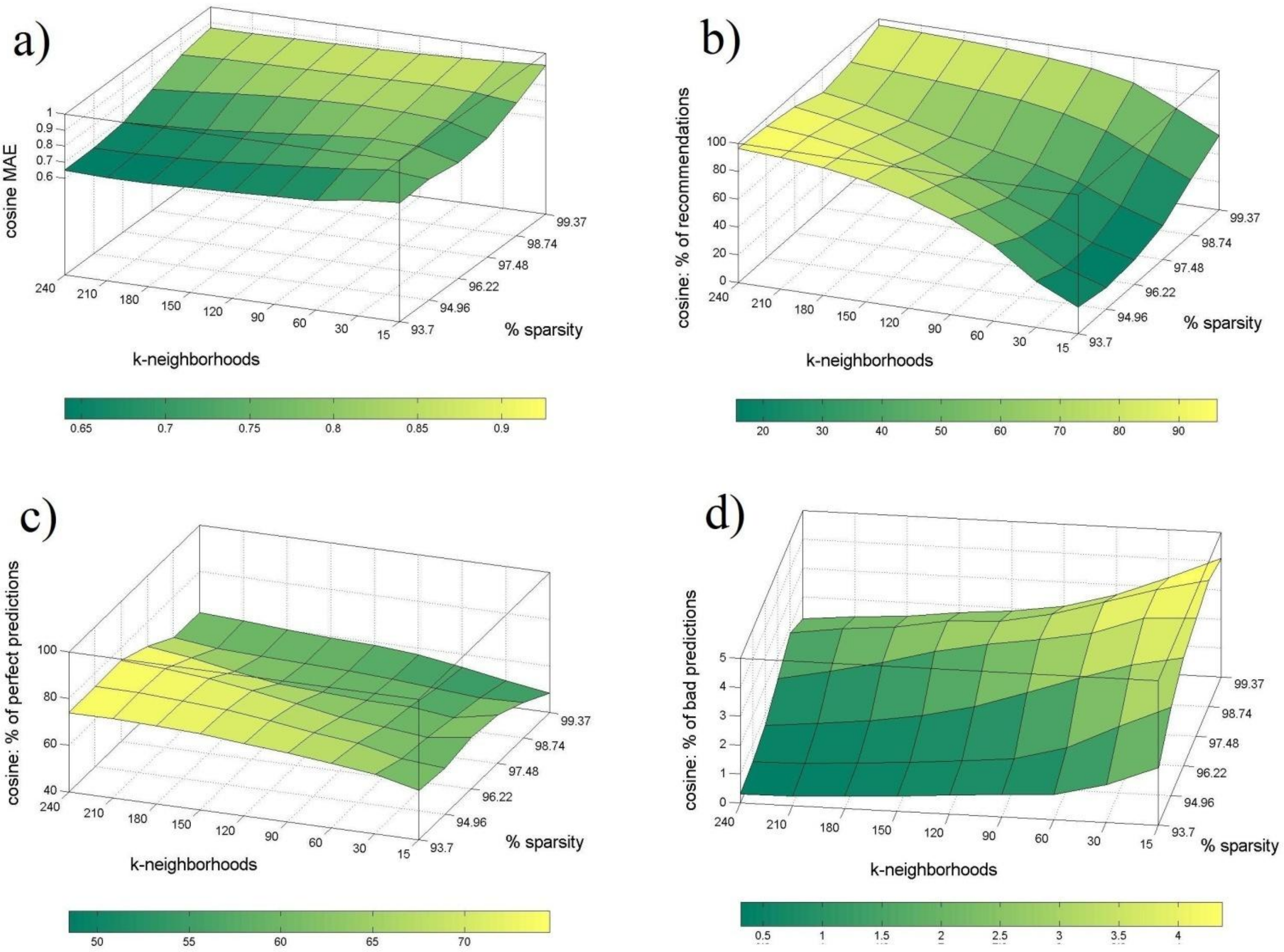


Figure 6. Results of the cosine metric: a) Mean Absolute Error, b) percentage of recommendations made, c) percentage of perfect predictions, d) percentage of bad predictions

number of predictions. As can be seen in Figure 7b), the percentage of recommendations obtained using the MSD metric is lower than that obtained using the cosine metric and even more so in the case of the correlation metrics. In this aspect, it can be observed that the function's slope is much more significant in the k-neighborhoods axis than in the percentage of sparsity axis; therefore, this aspect can be improved by choosing a high k-neighborhood value as opposed to working with high sparsity databases.

The quality of the recommendations (measured as high levels of perfect predictions combined with low levels of bad predictions) is very good when using the MSD metric, in comparison to the cosine and correlation metrics; this is mainly due to the low percentage of bad recommendations. By observing Figures 7c) and 7d) it can be determined that the best results (more perfect predictions and fewer bad predictions) are obtained at the highest values of sparsity. The poorest results occur when the highest k-neighborhood values are combined with the lowest percentages of sparsity levels.

In short, when using the MSD metric with low values of sparsity, it is necessary to choose a suitable k-neighborhood value to obtain a balance between quality (Figures 7a,c,d) and capacity to recommend (Figure 7b); the highest values of the k-neighborhood parameter offer us a better capacity for recommendation, while the lowest values of the k-neighborhood parameter lead to an improvement in the quality.

The most interesting observation in Figure 7 is that all the objectives (low error, high capacity to recommend, high percentage of perfect predictions and low percentage of bad predictions) are improved at the same time when the sparsity value increases. This characteristic confers a special importance to the MSD metric to be used in very sparse RS databases and it opens a way to creating new specialized MSD-based metrics.

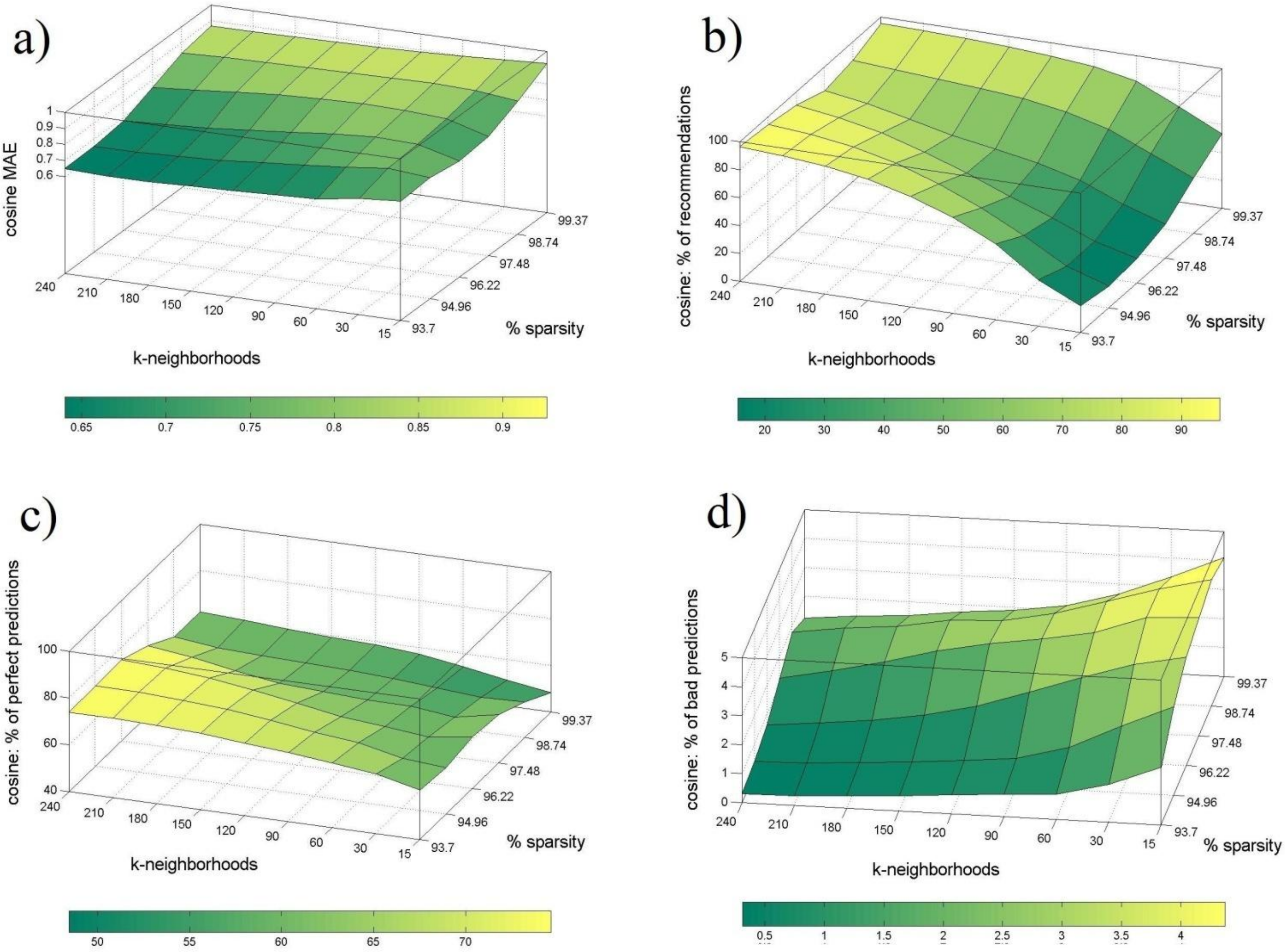


Figure 6. Results of the cosine metric: a) Mean Absolute Error, b) percentage of recommendations made, c) percentage of perfect predictions, d) percentage of bad predictions

5 Conclusions

The sparsity levels of RS databases have an important influence on the results of content-based collaborative filtering metrics. The impact of the sparsity influence depends on the k-neighborhood value selected, the main objective we want to maximize (MAE, capacity to recommend, etc) and, logically, on the metric used.

When the sparsity level increases:

- The Pearson correlation metric improves its MAE and has a negative effect on the capacity to recommend. In addition, its percentage of good predictions shows a slight increase.
- The cosine metric has a negative effect on all the aspects studied (MAE, capacity to recommend, correct predictions, incorrect predictions); this negative behavior can be reduced by selecting high k-neighborhood values.
- The Mean Squared Difference (MSD) greatly improves all the results except for the capacity to recommend.

The correlation metric obtains much better results than the cosine metric when working with sparse RS databases, especially when the k-neighborhood value is not high (preferably 60 and 90).

By using databases with a high degree of sparsity, the MSD metric obtains better results than the correlation metric in all the aspects studied except for the capacity to generate a large number of predictions.

The MSD metric presents unusually good behavior when applied to sparse RS ratio matrices. However, it should be used with caution due its very poor capacity to generate recommendations and its high probability of suffering from the effects of overspecialization; nevertheless, the MSD metric offers a serious alternative to the standard metrics when it is used in sparse ratio matrices and can be selected as a reference in designing new content-based CF metrics capable of satisfactorily tackling the RS sparsity problem.

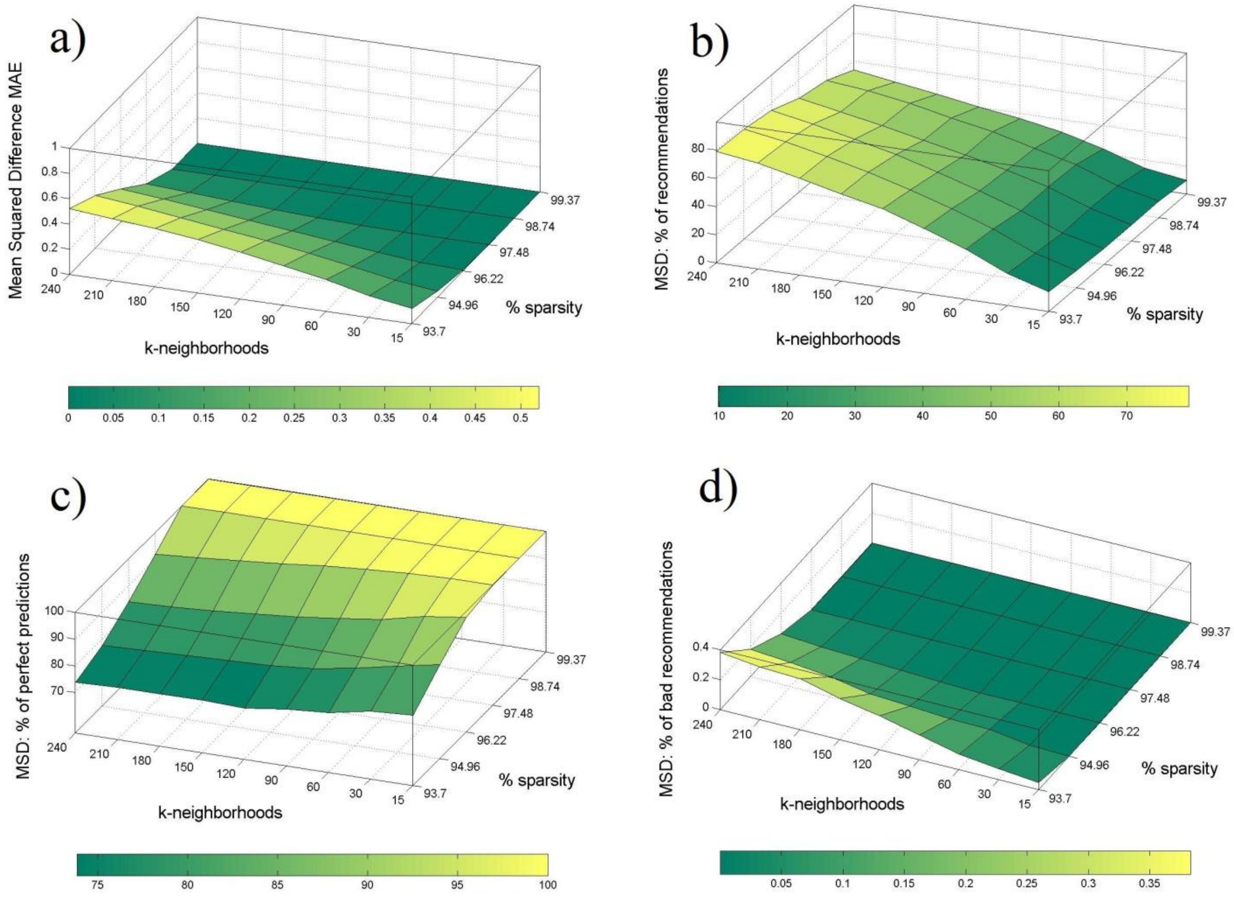


Figure 7. Results of the MSD metric: a) Mean Absolute Error, b) percentage of recommendations made, c) percentage of perfect predictions, d) percentage of bad predictions

6 References

1. Knights, M. Web 2.0, IET Communications Engineer, (February-March 2007), 30-35

2. Lin, K.J., Building Web 2.0, Computer, (May 2007), 101-102

3. Janner, T., Schroth, C. Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services, IT Pro, (May-June 2007), 36-41

4. Jinghua, H., Kangning, W., Shaohong, F. "A Survey of E-Commerce Recommender Systems", in Proceedings of the International Conference on Service Systems and Service Management, 2007, 10.1109/ICSSSM.2007.4280214, 1-5

5. Baraglia, R., Silvestri, F. "An Online Recommender System for Large Web Sites", in Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, 2004, 10.1109/WI.2004.10158, 199-205

6. Fesenmaier, D.R., Gretzel, U., Knoblock, C., Paris, C., Ricci, F., Stabb, S., Werther, H., Zipf, A. Intelligent Systems for Tourism, Intelligent Systems, vol. 17, no. 6, (nov/dec 2002), 53-66

7. Konstan, J.A., Miller, B.N., Riedl, J. PocketLens: Toward a Personal Recommender System, ACM Transactions on Information Systems, vol. 22, no. 3, (July 2004), 437-476.

8. Antonopoulus, N., Salter, J., CinemaScreen Recommender Agent: Combining Collaborative and Content-Based Filtering, IEEE Intelligent Systems, (January/February 2006), 35-41

9. Li, P., Yamada, S. "A movie recommender system based on inductive learning" in Proceedings of the IEEE Conference on Cybernetics and Intelligent Systems, vol. 1, 318-323

10. Adomavicius, Tuzhilin, A. Toward the Next Generation of Recommender Systems: a survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Enginnering, vol. 17, no. 6, (June 2005), 734-749

11. Herlocker, J. L., Konstan, J.A., Riedl, J.T., Terveen, L.G. Evaluating Collaborative Filtering Recommender Systems, ACM Transactions on Information Systems, vol. 22, no. 1, (January 2004), 5-53

12. Breese, J.S., Heckerman, D., Kadie, C. "Empirical Analysis of Predictive Algorithms for Collaborative Filtering", in Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, 1998, 43-52

13. Kong, F., Sun, X., Ye, S. A Comparison of Several Algorithms for Collaborative Filtering in Startup Stage, In Proceedings of the IEEE networking, sensing and control, (March 2005), 25-28

14. Cho, S.B., Hong, J.H., Park, M.H. Location-Based Recommendation System Using Bayesian User's Preference Model in Mobile Devices, Lecture Notes on Computer Science, vol. 4611, (August 2007), 1130-1139

15. Ingoo, H., Kyong, J.O., Tae, H.R. The Collaborative Filtering Recommendation Based on SOM Cluster-Indexing CBR, Expert Systems with Applications, vol. 25, 2003, 413-423

16. Yager, R.R. Fuzzy Logic Methods in Recommender Systems, Fuzzy Sets and Systems, vol. 136, no. 2, (June 2003), 133-149

17. Linden, G., Smith, B., York, J., Amazon.com Recommendations, IEEE Internet Computing, (January-February 2003), 76-80

18. Giaglis, G.M., Lekakos, Improving the Prediction Accuracy of Recommendation Algorithms: Approaches Anchored on Human Factors, Interacting with Computers, vol. 18, no. 3, 2006, 410-431

19. Li, Y., Nayak, R., Weng, L.T., Xu, Y., “An Improvement to Collaborative Filtering for Recommender Systems”, in Proceedings of the IEEE International Conference on Computational Intelligence for Modelling, Control and Automatitation, 2005

20. Fuyuki, I., Quan, T.K., Shinichi, H., “Improving Accuracy of Recommender Systems by Clustering Items Based on Stability of User Similarity”, in Proceedings of the IEEE International Conference on Intelligent Agents, Web Technologies and Internet Commerce, 2006

21. Manolopoulus, Y., Nanopoulus A., Papadopoulus A.N., Symeonidis P. Collaborative Recommender Systems: Combining Effectiveness and Efficiency, Expert Systems with Applications, 2007. in press.

22. Herlocker, J., Konstan J., Riedl, J. An empirical Analysis of Design Choices in Neighborhood-based Collaborative Filtering Algorithms, Information Retrieval, vol.5, no. 4, , 2002, 287–310.

23. Pazzani, M. A Framework for Collaborative, Content-Based and Demographic Filtering, Artificial Intelligence Rev., December 1999, 393-408.

24. Sarwar, B., Karypis, G., Konstan J., Riedl, J. Application of Dimensionality Reduction in Recommender Systems-A Case Study, in Proceedings of ACM WebKDD Workshop, 2000.

25. Goldbergh, K., Roeder, T., Gupta D., Perkins D., Eigentaste: A Constant Time Collaborative Filtering Algorithm, Information Retrieval, Vol. 4, 2001, 133-151..

26. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer T.K., Harshman, R., Indexing by Latent Semantic Analysis, in Proceedings of the JASIS, Vol 41(6), 1990.

27. Hofmann, T., Collaborative Filtering via Gaussian Probabilistic Latent Semantic Analysis, in Proceedings of the 26th ACM SIGIR Conference on Research and Development in Information Retrieval, 2003.

28. Papagelis, M., Plexousakis D. Kutsuras, T., Alleviating the Sparsity Problem of Collaborative Filtering Using Trust Inferences, Lectures Notes on Computer Science, Vol. 3477, 2005, 224-239.

29. Bruyn, A.D., Giles C.L., Pennock, D.M., Offering Collaborative-Like Recommendations when Data is Sparse: the Case of Attraction-Weighted Information Filtering, in Proceedings of Lecture Notes in Computer Science, Vol. 3137, 2004, 393-396.

30. Polcicova G., Peter Tino, Making Sense of Sparse Rating Data in Collaborative Filtering via Topographic Organization of User Preference Patterns, Neural Networks, Vol. 17, 2004, 1183-1199.

31. http://www.movielens.org.

Solving the Golden Transaction Problem for ARIES-based Multi-level Recovery

Jayson Speer[1] **Markus Kirchberg**[2,1] **Faizal Riaz-ud-Din**[3,1]
Klaus-Dieter Schewe[1]

[1] Information Science Research Centre, Palmerston North, New Zealand

[2] Institute for Infocomm Research (I^{2}R), A*STAR, Singapore

[3] Al Ain Men's College, Higher Colleges of Technology, Al Ain, United Arab Emirates

Contact: `MKirchberg@i2r.a-star.edu.sg` (Markus Kirchberg)

Abstract

Transaction throughput is a crucial issue for database systems. Multi-level transactions have been proposed in an attempt to offer improved concurrency of transaction processing by allowing operations that would otherwise be performed serially to take place concurrently. Therefore, it is vital that recovery algorithms do not impede this concurrency by artificially introducing restrictions that otherwise do not need to exist. The ARIES recovery algorithm has had a significant impact on the current thinking on database transaction logging and recovery. Its corresponding extension to multi-level transactions, i.e. ARIES/ML, preserves the unique features of ARIES but places significant restrictions on rollback processing. In this paper, we present an algorithm that solves the so-called *'golden transaction' problem* of the ARIES/ML algorithm.

1 Introduction

Transaction throughput is a crucial issue for database systems (DBSs). While concurrency is utilised to increase performance, it is vital to oversee the execution of transactions (to avoid data inconsistencies). In the event of a large number of inter-transaction conflicts the rate of transaction throughput decreases thereby making the DBS susceptible to poor response times.

Multi-level transactions (Beeri et al. 1989) have been introduced in an attempt to increase the rate of concurrency in DBSs. A *multi-level transaction (MLT)* is where there is more than one level of operations for a transaction. The lowest level of a MLT is the physical level that is synonymous with the physical level operations of single-level (i.e. traditional) transactions. There is, at least, one level of operations between the main transaction (highest level) and the physical level in a MLT.

Every level of a multi-level system realises a set of abstract states, each represented by a number of lower level states. Each level comprises of operations that are provided by the level immediately below, and provides a set of operations to the level immediately above. A level, therefore, supplies a sequence of operations provided by the level below it, into an operation that it supplies to the level above it. Lower levels of a transaction are also known as *sub-transactions*.

A multi-level system using locks permits restrictive low-level locks of a sub-transaction to be replaced with less restrictive high-level locks when subtransactions commit. This is possible because subtransactions can be undone via high-level compensation actions rather than restoring a prior lower level state – thus, enhancing concurrency.

The introduction of MLTs leads to the issue of finding an appropriate recovery mechanism. One of the most popular methods that is used by single-level DBSs to implement recovery is known as ARIES (Algorithm for Recovery and Isolation Exploiting Semantics) (Mohan et al. 1992). ARIES/ML (Schewe et al. 2000) is an extension of ARIES for multi-level systems. It deals with the added complexities of MLTs by introducing various log records and a new pointer that are used to represent the tree-like structure of MLTs. During crash recovery, ARIES/ML employs the same three basic phases (i.e. Analysis, Redo, and Undo) as the original ARIES algorithm. This allows ARIES/ML to preserve the same desirable properties of ARIES while providing recovery capability to multi-level transaction databases.

1.1 Motivation and Contribution

The purpose of the MLT model is to offer improved concurrency of transaction processing by allowing operations that would otherwise be performed serially to take place concurrently. That is, transactions are allowed to release 'low-level' locks before commit causing problems should such transactions need to reacquire these locks later to facilitate rollback processing. The major problem that this introduces is the possibility of a deadlock between transactions, where two or more such transactions are in rollback. It is not possible to resolve such a deadlock by simply aborting one of the involved transactions as they are being rolled back already. ARIES/ML avoids this problem altogether by allowing only a single, so-called *'golden transaction'* to be in rollback at any point in time. However, it is vital that any recovery algorithm does not impede the degree of concurrency by artificially introducing restrictions that otherwise do not need to exist. Although the ARIES/ML algorithm poses no such restrictions during normal processing, it places significant restrictions during rollback processing. In this paper, we propose an algorithm that solves ARIES/ML's *'golden transaction' problem*.

1.2 Outline

This paper is organised as follows: Section 2 reviews existing multi-level recovery algorithms and, in particular, introduces the ARIES/ML algorithm

Copyright ©2009, Australian Computer Society, Inc. This paper appeared at the 20th Australasian Database Conference (ADC 2009), Wellington, New Zealand, January 2009. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 92, Athman Bouguettaya and Xuemin Lin, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

in greater detail. Section 3 proposes the Golden Deadlock Algorithm (GDA) that solves ARIES/ML's golden transaction problem. An example is presented in Section 4 before Section 5 concludes this paper.

2 Related Work

From an internal point of view, users access databases in terms of transactions. Fast response times and a high transaction throughput are crucial issues for all DBSs. Hence, transactions are executed concurrently. The transaction management system ensures a proper execution of concurrent transactions. It implements concurrency control and recovery mechanisms to preserve the well-known ACID principles. A further improvement in both response time and transaction throughput can be achieved by employing a more advanced transaction management system, e.g. a system that is based on the multi-level transaction model (Beeri et al. 1989, Weikum 1986, 1991). This model is counted as one of the most promising transaction models. It schedules operations of transactions based on information that is obtained from multiple levels. Since there are usually less conflicts on higher levels, lower level conflicts[1] can be ignored. Hence, their detection increases the rate of concurrency. For instance, assume that two higher level operations op_1 and op_2, which belong to different transactions, increment Integer values A and B, respectively. Furthermore, let A and B reside on the same physical page. An increment will be executed as a page read followed by a page write. In the event that scheduling only considers operations on pages, op_1 will have to wait for op_2 or vice versa. Considering both levels, i.e. incrementations and read and write operations, we can allow op_1 and op_2 to execute concurrently. The information obtained from the higher level indicates that different portions of the page are accessed. Thus, the corresponding read and write operations do not affect each other. Protecting such individual operations by short-term locks (i.e. latches) is sufficient.

In (Weikum 1991) the execution of concurrent transactions is described by means of a multi-level schedule. Level-by-level (conflict-)serialisability is ensured by employing one-level schedulers, i.e. level-by-level schedulers, on each level. A multi-level schedule is multi-level (conflict-)serialisable, if all level-by-level schedules are (conflict-)serialisable.

The ARIES algorithm (Mohan et al. 1992) has had a significant impact on the current thinking on (single-level) database transaction logging and recovery. It has been incorporated into IBM's DB2 Universal Database, IBM's Lotus Notes and Domino, Microsoft SQL Server and NT file system, Apache Derby and a number of other systems (Mohan 2004). There are a few recovery algorithms for multi-level systems, i.e. ARIES/NT, MLR^W, MLR^L, and ARIES/ML, most of which aim at preserving advantages of the ARIES algorithm.

ARIES/NT (Rothermel & Mohan 1989) is an extension of the ARIES recovery algorithm to the nested transaction model. It permits semantically-rich modes of locking, operation logging, and efficient recovery. It also allows arbitrary parallelism between related and unrelated transactions. That is, a transaction may run concurrently with its superiors, inferiors, siblings, and all other unrelated transactions. Concurrency control schemes are supported, allowing upward and downward inheritance of locks. That is, child operations may inherit locks from their parents and vice-versa. Save-points at each transaction level are also supported. That is, top-level transactions as well as sub-transactions may establish save-points. However, ARIES/NT is not specifically designed for multi-level transaction processing. As a result, it does not utilise inverse operations, which are possible in multi-level transactions. Also, locks are not released after finishing operations that are not transactions.

The *Multi-level Recovery (MLR^W)* algorithm (Weikum et al. 1990) has some similarities to ARIES-based approaches but:

- It requires level-specific recovery mechanisms. A single technique is more desirable as it maintains only one log file and does not require any additional communication mechanisms between level-specific recovery mechanisms; and
- It does not avoid the undo of rollback actions.

The *Multi-Level Recovery (MLR^L)* algorithm (Lomet 1992) works with nested and multi-level transactions. As with ARIES, MLR^L supports operation logging while providing a flexible cache management. Undo recovery must be performed level-by-level but can be implemented with a single backward scan of the transaction log. MLR^L is not sensitive to the constraints that are enforced by the concurrency control protocol. As such, the only locks required are those needed for compensation and the MLR^L framework holds these locks until a sub-transaction's completion. The WAL protocol must be observed but whether the log is forced at (sub-)transaction commit is optional. In order to acquire higher level locks, extra overhead is incurred. In addition, MLR^L utilises inverse operations but, unfortunately, assumes them to exist in any case. If they do not exist, as in most practical situations, the restrictions of ARIES/NT are retained.

Finally, there is the ARIES/ML algorithm (Schewe et al. 2000), which is similar to MLR^L but is not necessarily coupled with a locking protocol. It differentiates between operations for which there exists an inverse, and those for which inverses do not exist. The multi-level transaction model allows transactions to release low-level locks before commit, which can cause problems should such transactions need to re-acquire these locks at some later stage to facilitate a rollback. The major problem this introduces is the possibility of a deadlock between two or more transactions that are rolling back. In such a case, resolving the deadlock is not possible by aborting one of the transactions, since they are already in rollback. A fundamental flaw of the ARIES/ML algorithm is that it does not offer any method of dealing with such deadlocks between two (or more) aborting transactions. It avoids the problem altogether by allowing only a single *'Golden' transaction* to be in rollback at any point in time. Since any deadlock involving the golden transaction can be resolved by aborting the other normal transactions, the above mentioned problem never arises. Clearly, however, this is not a desirable solution since it has a significant negative impact on the performance of the algorithm. In fact, concurrency of transaction rollback is eliminated entirely. Section 3 proposes enhancements to the ARIES/ML algorithm that allows deadlocks between one or more aborting transactions to be resolved.

2.1 Overview of ARIES/ML

As previously mentioned, ARIES/ML (Schewe et al. 2000) is similar to MLR^L (Lomet 1992) but avoids some of MLR^L's restrictions. The salient features of ARIES/ML include:

[1] Such conflicts are referred to as *pseudo-conflicts*, which are low-level conflicts that do not stem from a higher level conflict.

- The main features of ARIES such as the ability to execute partial rollbacks, support of different locking granularities etc. are preserved;
- It may be coupled with all types of concurrency control protocols;
- It supports operation logging in addition to logical logging – this is a result of its ability to explicitly distinguish between operations for which an inverse exists and those for which there is none; and
- Deadlocks are resolved with less effort by enabling the undo and redo of a single (sub-) transaction i.e. it does not necessitate the use of savepoints.

Data Structures. Log records that make up the transaction log mainly describe operations of subtransactions. The only operations that are logged are those that potentially result in data objects being updated. Each of the log records belongs to a specific *backward chain (BW-chain)*, which is a related group of log records that make up a sub-transaction. Subtransactions that have committed are reflected in the BW-chain of their parent's log entries as CCR log records. The primary ARIES/ML log records are:

- *Update Log Records (ULRs)*, which reflect changes to data objects;
- *Compensation Log Records (CLRs)*, which reflect the undo of an update operation or a compensation operation;
- *Child-Committed Log Records (CCRs)*, which reflect the commit of a lower level sub-transaction;
- *Prepare Log Records (PLRs)*, which reflect the beginning of the first phase of a commit protocol for the corresponding top-level transaction; and
- *End Log Records (EndLRs)*, which reflect the end of the top-level transaction.

The primary fields that are used in the log records are as follows:

- *PrevLSN*, which is the address of the preceding log record in the BW-chain of the corresponding (sub-)transaction. This value is *null* for the first log record written by the corresponding (sub-) transaction;
- *PageID*, which is the page identifier of the page that was updated by the operation that the log record relates to;
- *ChildID*, which is the identifier of a committed sub-transaction;
- *LastLSN*, which is the address of the last log record in the BW-chain of the committed (sub-) transaction; and
- *UndoNextLSN*, which is the address of the next log record that has to be undone in the event of a rollback.

Figure 1 outlines an example of how ARIES/ML log records and their fields are utilised in order to chain log records together.

ARIES/ML also utilises the same two in-memory data structures as ARIES, i.e. the transaction table and the dirty page table. The *transaction table* is where references to currently active transactions are stored. References to the parent transactions of these active transactions are also stored in the transaction table. This is done in the event that the parent of a transaction in the transaction table needs to be undone. The *dirty page table* contains references to buffered pages that have not yet been written to disk. Entries in the dirty page table include the PageID of the page that has been updated, and the recovery LSN (i.e. *RecLSN*) of the first log record that caused an update to the page that was not written to disk (or made persistent).

Normal Processing. A transaction table entry is written when a (sub-)transaction creates its first log record. This entry is removed when the (sub-)transaction either commits successfully or fully aborts. A (sub-)transaction is committed successfully once it appends a CCR log record to its parent's BW-chain. The following tasks are performed when a data object is updated:

- The corresponding page is fixed in the buffer and latched in exclusive mode;
- The dirty page table is updated if this page was not dirty before;
- The update is applied according to the concurrency control approach being implemented;
- An ULR log record is appended to the (sub-) transaction's BW-chain;
- The log record's LSN is placed in the PageLSN field of the corresponding page;
- The (sub-)transaction's transaction table entry is updated; and
- The page is unlatched and unfixed.

Rollback. The rollback process refers to the abortion of all effects of a transaction. This involves undoing all top-level operations of a transaction including all committed and currently active sub-transactions. The *BWC-tree* is represented by the BW-chains of top-level transactions and aborted/active subtransactions. The *BWC-forest* consists of a transaction's BWC-tree and the BWC-trees of that transaction's active sub-transactions. The rollback of a transaction corresponds to the undo compensation of the log records belonging to the transaction's BWC-forest. ARIES/ML supports partial rollbacks through the use of save-points (or separate roll back of a subtransaction) for faster resolution of deadlocks.

Rollback is initialised by providing a set of transaction identifiers (i.e. TransIDSet), which have to be rolled back, and a save-point. The following tasks are then executed:

- A rollback list is created that contains the LastLSN from all active (sub-)transactions that have a parent in TransIDSet.
- Undo operations are processed in order of decreasing LSN following the PrevLSN entries in log records. LSN values less than the save-point are not considered. Processing is complete when the last entry in TransIDSet has been visited.
- Any updates described by ULR log records are undone and a corresponding CLR log record is created.
- When a transaction's BW-chain is completely rolled back, an EndLR log record is created, and the corresponding transaction table entry is deleted.

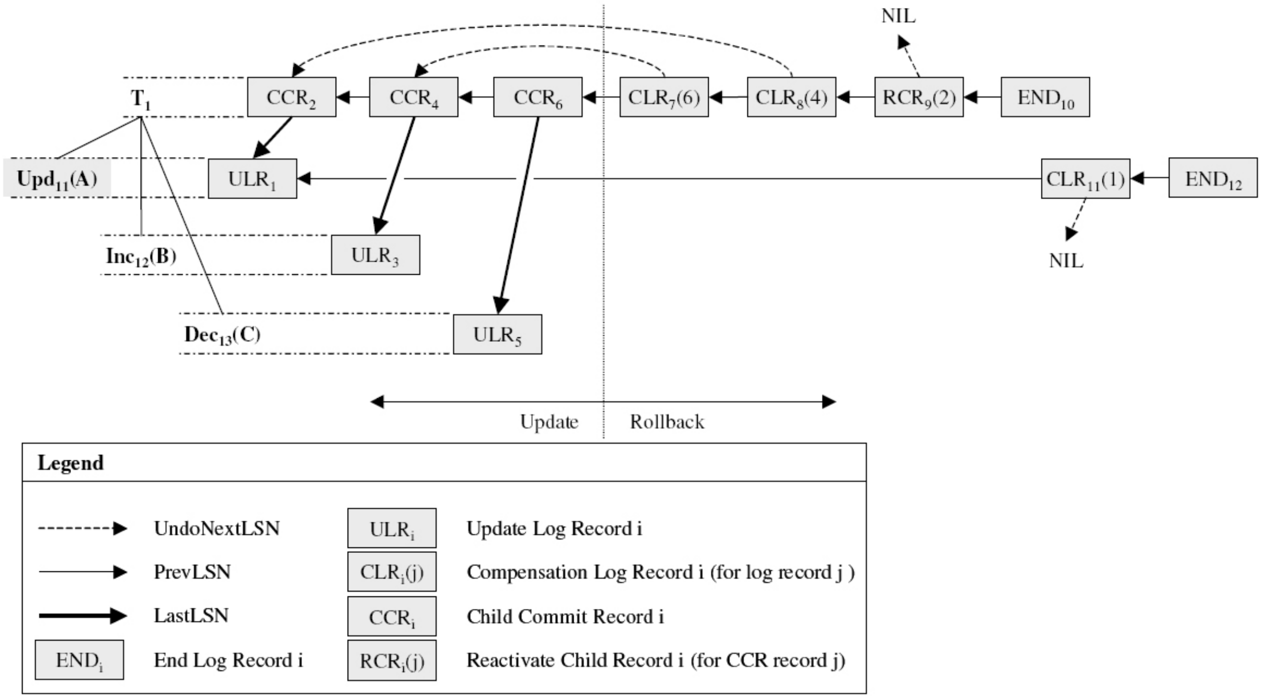


Figure 1: ARIES/ML Log Structure Example

- When a CCR log record is encountered, a previously committed sub-transaction has to be undone. Different methods of undo may be used depending on the existence of an inverse operation for the corresponding operation.
- When a *reactive-child log record (RCR)* or a PLR log record is encountered, the corresponding PrevLSN value in the rollback list is updated.

Example 1 *Consider Figure 1 where transaction T_1 is shown with three child operations Upd_{11}, Inc_{12} and Dec_{13}. If transaction T_1 completes all child operations before being aborted, this will require the rollback of all child operations in reverse order, where operations Inc_{12} and Dec_{13} can be rolled back via an inverse operation, but operation Upd_{11} cannot.*
Also shown in Figure 1 is the log record structure for both the updates and the rollback performed by transaction T_1. The log is split into two parts, being firstly the part of the log that records updates made by transaction T_1 and secondly the part of the log that records the rollback of those updates. Each of the update operations has the same log record structure, being a ULR log record written to represent the updates performed by the child operation followed by a CCR log record representing the completion of that operation. For example, operation Upd_{11} is represented by ULR_1 and CCR_2 (that has a LastLSN pointer referring back to ULR_1). However, the log record structure used to record the rollback of operation Upd_{11} differs from operations Inc_{12} and Dec_{13}, since Upd_{11} cannot be rolled back by applying an inverse operations whereas Inc_{12} and Dec_{13} can. Therefore, the rollback of operations Inc_{12} and Dec_{13} are each recorded by a single CLR log record corresponding to the child commit log records (i.e. CCR_4 and CCR_6 respectively), while the rollback of Upd_{11} is recorded by an RCR log record followed by a CLR log record corresponding to the update log record ULR_1.

Checkpointing. Checkpoints are used to expedite crash recovery. This consists of three steps:

1. A *begin-checkpoint log record* is appended to the log;
2. An *end-checkpoint log record* containing copies of the transaction table and the dirty page table is appended to the log; and
3. The log is forced to stable storage, and the LSN of the begin-checkpoint log record is stored in the Master log record, which is also stored on stable storage.

Restart Processing. Crash recovery follows the same principles as the original ARIES algorithm. The Analysis pass reconstructs the transaction table as well as the dirty page table. The starting LSN for the subsequent redo phase is set to the lowest LSN in the dirty page table. The redo pass restores the state of the database to the state it was in immediately before the system crash. Finally, the undo pass rolls back any effects of the transactions which were active at the time of the crash.

For more details on the ARIES/ML algorithm refer to (Schewe et al. 2000, Drechsler 1998, Riaz-ud-din 2002).

3 The Golden Deadlock Algorithm (GDA)

As discussed in Section 2, the fundamental flaw of the ARIES/ML algorithm is that it offers no method for dealing with deadlocks between two or more aborting transactions. Instead, it avoids the issue entirely by allowing only a single 'golden' transaction to be in rollback at any one time. While this avoids the possibility of deadlocks between transactions in rollback, it places unreasonable restrictions on performance. In this section, we propose the *Golden Deadlock Algorithm (GDA)*, which augments the ARIES/ML recovery algorithm in a way that permits the resolution of deadlocks between a set of transactions in rollback.

The GDA assumes that the set of transactions T_D in deadlock have locks acquired on a set of objects O_D and that a set of (tie-break) objects $O_T \subseteq O_D$

can be chosen to break the deadlock. That is, if no transaction in T_D requires a lock on any object in O_T, the deadlock will be resolved. Given the set of deadlocked transactions T_D and the tie-break objects in O_T, it is the objective of the GDA to roll back all updates made to the tie-break objects in O_T by transactions in T_D in a single action. Hence, removing the need for any transaction in T_D to maintain its lock on any tie-break object in O_T. The consequence of this is, provided the tie-break objects were chosen appropriately, that the deadlock is broken. Each transaction is then able to continue the normal rollback process. During the course of rolling back updates to the tie-break objects, it may be necessary for the GDA to trigger the cascading abort of one or more transactions T_C whose updates to tie-break objects must also be rolled back. This is determined by detecting conflicts between the deadlocked transactions and all other transactions.

In order to accommodate the GDA, a number of areas of the ARIES/ML algorithm must be augmented, these being: Lock acquisition, logging, data structures, and rollback algorithms.

3.1 Lock Acquisition

Before the GDA can roll back any operations performed on the tie-break objects in O_T, it must first acquire an exclusive lock on each object in O_T. In order to avoid the GDA itself becoming involved in a deadlock situation, it is necessary to observe the following rules when acquiring locks:

1. The GDA takes precedence over all transactions. This also means that any transaction that currently has a lock on a tie-break object is forced to relinquish it. Ultimately, this may lead to the transaction being aborted.
2. If another invocation of the GDA algorithm already has a lock on any tie-break object, then this invocation must wait.
3. The GDA must acquire exclusive locks on all tie-break objects atomically.

Clearly there exists the possibility that some transaction(s) not in T_D will hold some form of lock on objects in O_T at the time the GDA is invoked. To reduce the number of cascading transactions (T_C), it is possible to shelve some transactions rather than aborting them. Shelving involves the following actions:

- Temporarily pausing the transaction's processing;
- Forcing the transaction to relinquish its locks on all tie-break objects; and
- Once the GDA has completed processing, the transaction can be allowed to reacquire its locks and continue processing.

Rules for deciding whether a transaction should be shelved or aborted are:

- Abort: If the transaction has a lock on any object in O_T that may lead to a dirty read[2] then it must be added to the set of cascading transactions T_C.
- Shelve: If the transaction has no such locks, it may be added to the set of shelved transactions T_S.

Shelving a transaction does not guarantee that the GDA will not trigger its abort at some later point. If the GDA subsequently determines that a transaction in T_S must be aborted, its identifier can be removed from T_S and placed in T_C.

Example 2 *Consider the scenario where two transactions T_1, $T_2 \notin T_D$ hold the following locks on some object in O_T: T_1 holds Inc and Dec locks whereas T_2 holds only a Read lock.*
Transaction T_1 can be added to the set of shelved transactions T_S, since it has not acquired any locks on a tie-break object that can lead to a dirty read. However, transaction T_2 must be added to the set of cascading transactions T_C, since it has acquired a lock on a tie-break object that might lead to a dirty read.

3.2 Logging

The introduction of the GDA algorithm requires the following changes and additions to the log records written by the ARIES/ML algorithm.

Addition of the GDA Start Record (GSR). The purpose of the GSR log record is to record the invocation of the GDA algorithm. The GSR log record contains sufficient information to allow the algorithm to determine the correct course of action in the case of crash recovery. Its fields are:

- GID[3]: The GDA identifier.
- T_D: Set of deadlocked transactions that caused the GDA to be invoked.
- O_T: Set of tie-break objects.
- SaveLSN: The save-point for the rollback.

Addition of the GDA End Record (GER). The purpose of the GER log record is to record the successful completion of the GDA. The GER log record is written once all updates to the tie-break objects in O_T by transactions in both T_D and T_C have been undone. The only field contained in the GER log record is GID.

Addition of the GDA Cascading Aborts Record (GCR). The purpose of the GCR log record is to record any transaction that the GDA was forced to abort due to conflicts with transactions in T_D or T_C. The fields contained in the GCR log record are: GID and TransId, the identifier of the transaction aborted.

Introduction of the ObjectLastLSN Pointer. The ObjectLastLSN pointer is added to ULR, CCR, CLR, and RCR log records and points to the last log record that records an update to the corresponding object. The value of the ObjectLastLSN pointer can be taken directly from the object table (refer to Section 3.3). The ObjectLastLSN pointer in conjunction with both the LastLSN and PrevLSN pointers provide a linked list of log records that record updates to each object.

Figure 2 shows the use of the ObjectLastLSN pointer in conjunction with both the LastLSN and PrevLSN pointers previously defined by (Schewe et al. 2000, Drechsler 1998).

[2] A dirty read is interpreted as any operation reading an object whose outcome may be affected by the value read. For example, an increment operation is not considered a dirty read since its outcome is unaffected by the object's initial value.

[3] GID is used to uniquely identify each invocation of the GDA algorithm.

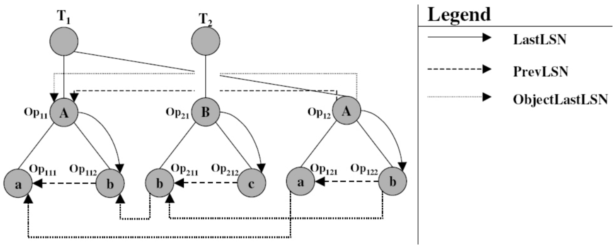


Figure 2: ObjectLastLSN Example

3.3 Data Structures

In order to accommodate the GDA, the following additional data structures are introduced:

Object Table. The object table is used to store the ObjectLastLSN pointers for each database object O_i. If no entry for object O_i exists, the ObjectLastLSN pointer is set to *null*. The fields contained in the object table are:

- ObjectID: The object identifier;
- ObjectLastLSN: LSN of the object's last log record that describes an update;
- ObjectFirstLSN[4]: LSN of the object's first log record describing an update; and
- PinCount: Number of top-level transactions that have updated this object.

Whenever a transaction writes a log record LSN$_j$ for an object (e.g. O_i) for the first time, it creates or updates an object table entry as follows:

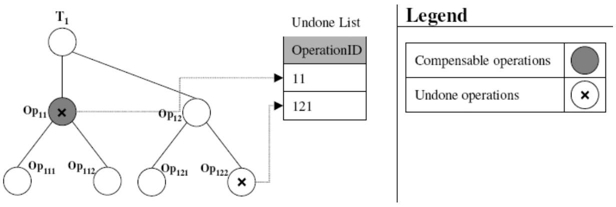
- If no entry exists for object O_i, create entry $(O_i, LSN_j, 1)$; or
- If an entry exists, set its ObjectLastLSN value to LSN_j and increment PinCount.

Whenever a transaction commits, it decrements the PinCount of all objects that it wrote a log record for. If this results in PinCount being 0, the entry is deleted from the object table.

Object Undo List. The purpose of the object undo list is to determine the order in which the updates to the tie-break objects (O_T) will be undone. The fields contained in the object undo list are as follows: ObjectID and ObjectLastLSN.

Each invocation of the GDA has its own object undo list, which is initialised by adding an entry for each object in O_T. The value of ObjectLastLSN for each object is taken from the object table.

In order to preserve database consistency, the GDA undoes operations in reverse order. This is achieved by always undoing the operation that has the highest ObjectLastLSN value in the object undo list. Each time an operation is undone, the ObjectLastLSN value in the object undo list is replaced by the ObjectLastLSN value in the log record whose changes have been undone. If the new value of ObjectLastLSN is *null*, the entry is deleted from the object undo list.



OperationID
11
121

Figure 3: Undone List Example

Undone List. The purpose of the undone list is to ensure that no operation is rolled back more than once. Without the inclusion of the undone list, it would be possible for an operation belonging to some transaction T_i to be undone by the GDA and then subsequently be undone again by another invocation of the GDA or as transaction T_i proceeds with the normal rollback process.

To achieve this, each time the GDA rolls back an operation, it stores the identifier of that operation in the undone list. Each entry in the undone list persists until the transaction to which it belongs is no longer active. By consulting the undone list during rollback, each transaction and invocation of the GDA can avoid rolling back an operation a second time.

Applying a compensation operation rather than applying before images at the lowest level can be used to roll some types of operations back, these operations are referred to as *compensable*. In such cases, the GDA need not roll back all the child operations. Instead, rollback is achieved by applying the corresponding compensation operation. In such cases, only the identifier of the parent operation will appear in the undone list. Therefore, whenever a transaction checks whether or not a particular operation has been undone, it must also check whether or not its parent operation has been undone. Since operation identifiers are extensions of their parent identifier, this process is relatively simple.

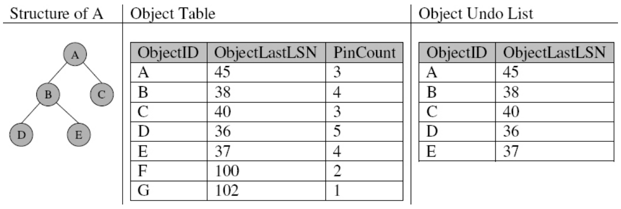
Example 3 *Consider the scenario in Figure 3 where operations Op_{122} and Op_{11} have been rolled back by some invocation of the GDA. Since operation Op_{11} is compensable, it has been undone by applying a compensating operation, which means the undone list will not contain the identifiers of its child operations. Therefore, when transaction T_1 checks whether or not operations Op_{111} and Op_{112} have been rolled back, it will also need to check whether or not their parent operation (i.e. Op_{11}) has been undone.*

3.4 Rollback Algorithms

The purpose of this algorithm, given the set of deadlocked transactions (T_D) and the set of tie-break objects (O_T), is to roll back all updates made to the tie-break objects by the transactions in T_D. In doing so, some transactions may be found to be in conflict and will be added to the set of cascading transactions (T_C). In the case where a save-point is specified, only updates that took place after the save-point will be undone. The GDA algorithm is split into three phases:

Initialisation Phase. The initialisation phase is responsible for priming the object undo list in preparation for use by the undo phase and acquiring the necessary exclusive locks on the tie-break objects. This involves adding an entry to the object undo list for each of the tie-break objects and their constituent objects. The constituents of any tie-break object must also be added to the object undo list

[4] The ObjectFirstLSN field is required only for GDA's distributed version (Speer 2005), but it is also offered as an optional enhancement to the (centralised) GDA algorithm.



ObjectID	ObjectLastLSN	PinCount
A	45	3
B	38	4
C	40	3
D	36	5
E	37	4
F	100	2
G	102	1

ObjectID	ObjectLastLSN
A	45
B	38
C	40
D	36
E	37

Figure 4: Initialisation Phase Example

since all updates to these constituents by the deadlocked transactions (T_D) must be rolled back.

Once the object undo list has been constructed and the locks acquired, the initialisation phase writes a GDA Start (GSR) log record, which includes the tie-break objects (O_T), the save-point (SaveLSN), the identifiers of the deadlocked transactions (T_D) and the unique GDA identifier. If a decision has been made to allow the algorithm to shelve transactions, this will be done in the initialisation phase by adding these transactions to the set T_S.

Example 4 *Consider the scenario in Figure 4 where the tiebreak object set is $O_T = \{A\}$ and object A has constituent objects $\{B, C, D, E\}$. Given the structure of object A and the object table, at the completion of the initialisation phase the object undo list will appear as shown.*

Undo Phase. The undo phase of the GDA algorithm is responsible for rolling back the operations performed on the tie-break objects by applying either before images or compensation operations. As each operation is rolled back, the appropriate log records are written. Figure 5 illustrates the undo phase by use of a flow chart. Explanations for each step are as follows:

0: If there are no more entries in the object undo list, all log records have been processed and the algorithm prepares to terminate; otherwise, it continues.

1: Retrieve the next log record, i.e. the one whose LSN is the same as the highest of all ObjectLastLSN values in the object undo list, to process.

2: If this log record belongs to any transaction in T_C or T_D, the operation that it represents must be rolled back; otherwise, it may be left alone.

3: Determine whether or not this operation has previously been rolled back. There are two situations under which this may occur:

   (a) A previous invocation of the GDA has rolled back this operation; or

   (b) This GDA invocation has previously rolled back the operation's parent.

   In the former situation, the Undone List can be used to determine whether or not an operation has been undone. In the latter situation, this GDA invocation keeps a record of all operations it has undone and can therefore determine whether or not it has previously rolled back the operation's parent.

4: The transaction manager determines whether or not this operation is in conflict with any other operation. If so, some cascading aborts will need to be triggered; otherwise, the rollback process continues unabated.

5: Determine whether or not this is a parent operation. The update operation is a parent operation if the log record that represents it is a CCR. Otherwise, the update operation is a child operation (represented by a ULR log record).

6: Determine whether or not this operation is compensable. If it is not, undo the effects of this operation by rolling back its child operations.

7: Determine whether or not any of the operation's child operations have already been rolled back. The only situation under which this can occur is if a previous GDA invocation has rolled back some of this operation's child operations. The Undone List can be used to test this condition.

8: This step rolls back a parent operation that is compensable. That is, the operation is compensable and none of its child operations have been previously rolled back. Rollback is achieved via the following steps:

   (a) Write a CLR log record that contains all necessary data; and

   (b) Perform the compensation operation required to roll back this operation.

9: Prepare a non-compensable parent operation for rollback. This step writes a RCR log record for a parent operation that cannot be rolled back via a compensation operation and subsequently activates the child operation.

10: This step rolls back a child operation that resides on level 0 (i.e. the lowest level) and therefore has no child operation of its own. Rollback is achieved via the following steps:

   (a) Write a CLR log record that contains all necessary data; and

   (b) Apply a before image to the object if the operation is not compensable or, otherwise, apply a compensation operation.

11: All operations that are in conflict with an operation that is to be rolled back must be scheduled for rollback themselves. Furthermore, these conflicting operations must be scheduled in such a way that the GDA will roll back all operations in reverse order to which they were originally performed. This can be achieved by updating the object undo list as follows:

   - If an operation Op_i is in conflict with any subsequent operation(s) performed on some object $Ob_j \in O_T$, update Ob_j's entry in the object undo list (ObjectLastLSN := largest LSN of any conflicting operation).

12: In the case where an operation that is to be rolled back is in conflict with subsequent operations, the transactions to which those operations belong must be added to T_C and triggered for abort. If any of these transactions are shelved, they must be removed from T_S.

13: Once the GDA invocation has completed rolling back all necessary operations, it stores the identifiers of those operations in the Undone list.

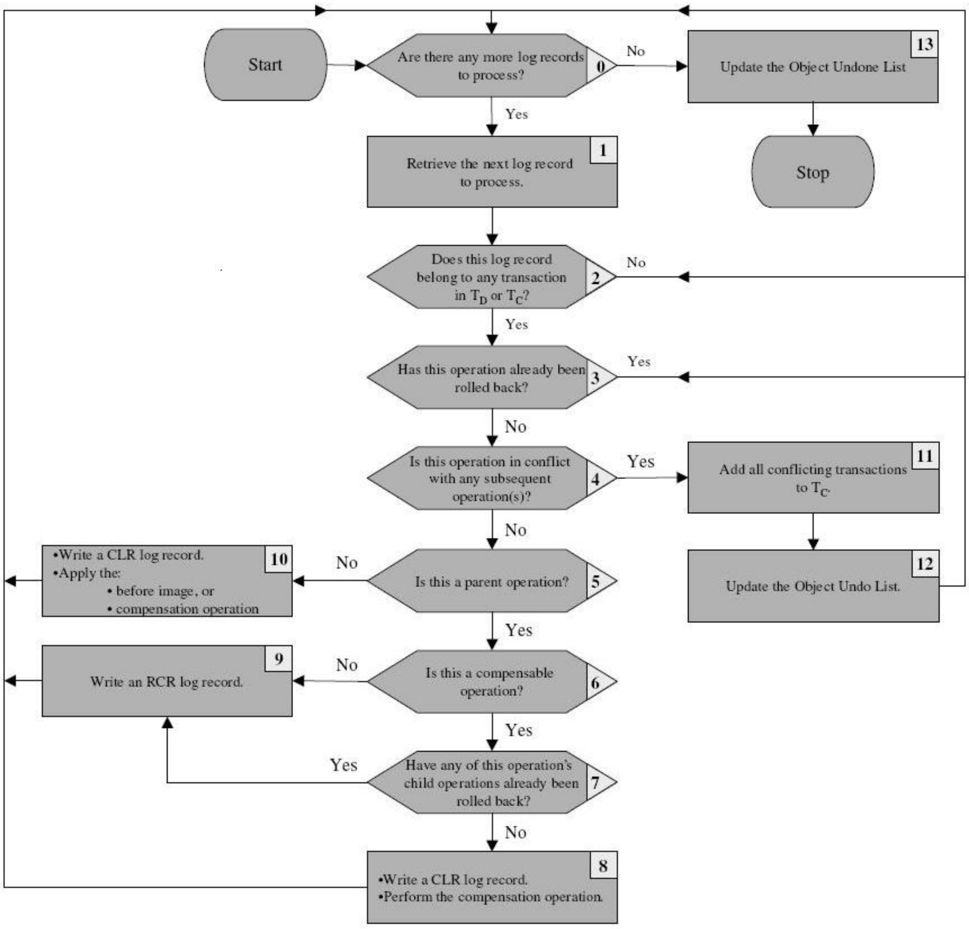

Start
Are there any more log records to process?
0
No
13
Update the Object Undone List
Stop
Yes
1
Retrieve the next log record to process.
Does this log record belong to any transaction in T_D or T_C?
2
No
Yes
Has this operation already been rolled back?
3
Yes
No
Is this operation in conflict with any subsequent operation(s)?
4
Yes
11
Add all conflicting transactions to T_C.
12
Update the Object Undo List.
No
10
•Write a CLR log record.
•Apply the:
• before image, or
• compensation operation
No
Is this a parent operation?
5
Yes
9
Write an RCR log record.
No
Is this a compensable operation?
6
Yes
Yes
Have any of this operation's child operations already been rolled back?
7
No
8
•Write a CLR log record.
•Perform the compensation operation.

Figure 5: Undo Phase Algorithm Flowchart

Completion Phase. The completion phase is simply responsible for releasing all locks acquired during the initialisation phase and recording the successful completion of a GDA invocation. The latter is achieved by writing a GER log record. If transactions had been shelved, at this point, the GDA instructs all transactions in T_S to resume processing. Resumption of processing involves the re-acquisition of the locks that the shelved transactions were forced to relinquish.

Termination of Algorithm. At first glance, it may be difficult to determine when the GDA algorithm will actually terminate. In fact, it may seem initially that it will not terminate until the beginning of the log has been reached, which would, of course, be disastrous. However, the nature of the object table ensures that this is not the case. Since the object table only maintains the LastLSN value for an object as long as it remains pinned by some active transaction, each time an object is unpinned by all active transactions, its LastLSN value will be reset to *null*. The result of this is that the GDA will only traverse backwards through the log until it reaches the first reference to a tie-break object in O_T from any active transaction; or until the save-point is reached.

Although the GDA algorithm does not traverse backwards to the beginning of the log, it does however process more records than actually required. This leads to the introduction of an optimisation (refer below) that ensures that the GDA algorithm processes exactly those log records required.

Example 5 *Consider the scenario illustrated by the partial log shown in Figure 6, where updates to the tiebreak object $O_T = \{A\}$ performed by the deadlock and cascading transactions $T_D \cup T_C = \{T_5, T_6, T_7\}$ must be rolled back by the GDA algorithm.*
Once invoked, the GDA will ascertain the last log record written for object A (i.e. LSN_{25}) by consulting the object table and then will traverse backwards through the log using the ObjectLastLSN pointers stored in each log record. Since log record LSN_{10} has a null ($\perp$) ObjectLastLSN pointer, the GDA algorithm will terminate at this point. Log record LSN_{10} has a null ObjectLastLSN pointer since both transactions T_1 and T_2 unpinned object A in the Object Table when they each committed, therefore resetting the ObjectLastLSN pointer for object A to null at that point. Although the GDA algorithm did not traverse backwards to the beginning of the log, it did however process more records than were actually required. In particular, it processed log records LSN_{15} and LSN_{16}, neither of which was written by a deadlocked or cascading transaction. The conclusion that can be drawn from this is that while the GDA algorithm will terminate at some point before the beginning of the log is reached, it will process more log records than actually required.

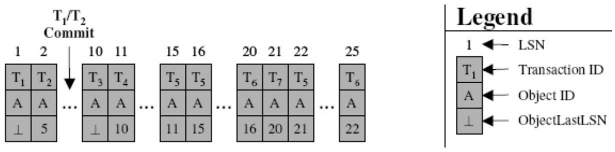
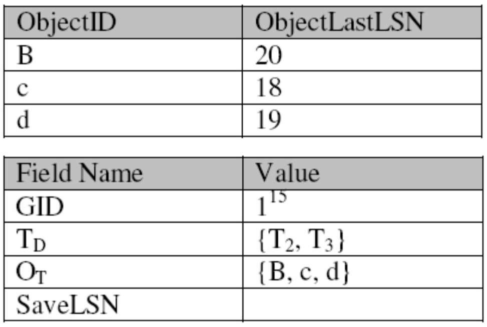


Figure 6: Termination of the GDA Algorithm

Termination Refinement. This refinement requires the collection of some additional data during



ObjectID	ObjectLastLSN
B	20
c	18
d	19

Field Name	Value
GID	1^{15}
T_D	$\{T_2, T_3\}$
O_T	{B, c, d}
SaveLSN	

Figure 8: Object Undo List and GSR Log Record

normal processing[5]. The transaction manager is required to maintain the FirstLSN value for each transaction, where *FirstLSN* is the log sequence number of the first log record written by a transaction. This data can easily be added to the transaction table, which already stores data pertinent to all active transactions. The newly defined FirstLSN subsequently allows the definition of TermLSN for each GDA invocation, where *TermLSN* is defined as the lowest FirstLSN value of all deadlocked transactions (T_D) or cascading transactions (T_C). TermLSN must be calculated by the GDA during the initialisation phase once the set of deadlocked transactions (T_D) is known and then recalculated each time the set of cascading transactions (T_C) is updated. To ensure that the GDA algorithm terminates at the correct point, the following rule for updating an entry in the object undo list must be modified: *Each time an operation is undone, the ObjectLastLSN value in the object undo list is replaced by the ObjectLastLSN value in the log record whose changes have been undone. The entry should be removed from the object undo list if the new value of ObjectLastLSN is null or less than TermLSN.*

4 GDA Invocation – An Example

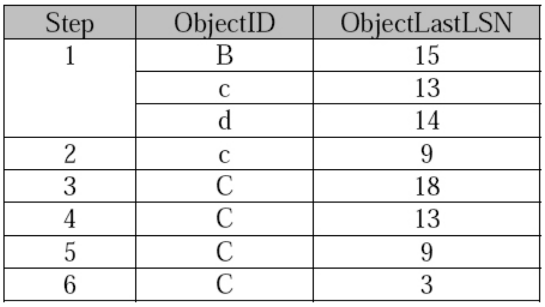
Consider the partial schedule and object table in Figure 7, where the GDA has been invoked in order to break a deadlock between transactions $T_D = \{T_2, T_3\}$ on the tie-break object $O_T = \{B\}$ using a *null* savepoint. It is assumed for this example that the constituents of object B are $\{c, d\}$ and the conflict relation for the partial schedule is $\{(Op_{212}, Op_{121})\}$.

During the initialisation phase, the GDA algorithm takes actions as follows:

- Generate a unique identifier (GID);
- Determine the set of tie-break objects O_T (including the constituent objects);
- Construct the object undo list;
- Acquire an exclusive lock on the tie-break objects; and
- Write a GDA Start (GSR) log record.

The contents of the object undo list and GSR log record are shown in Figure 8.

Figure 7 shows GDA sequence numbers, which represent the sequence in which the GDA algorithm will process the log records. The following describes the actions taken by this GDA invocation during the undo phase:



Step	ObjectID	ObjectLastLSN
1	B	15
	c	13
	d	14
2	c	9
3	C	18
4	C	13
5	C	9
6	C	3

Figure 9: Overview of States of the Object Undo List

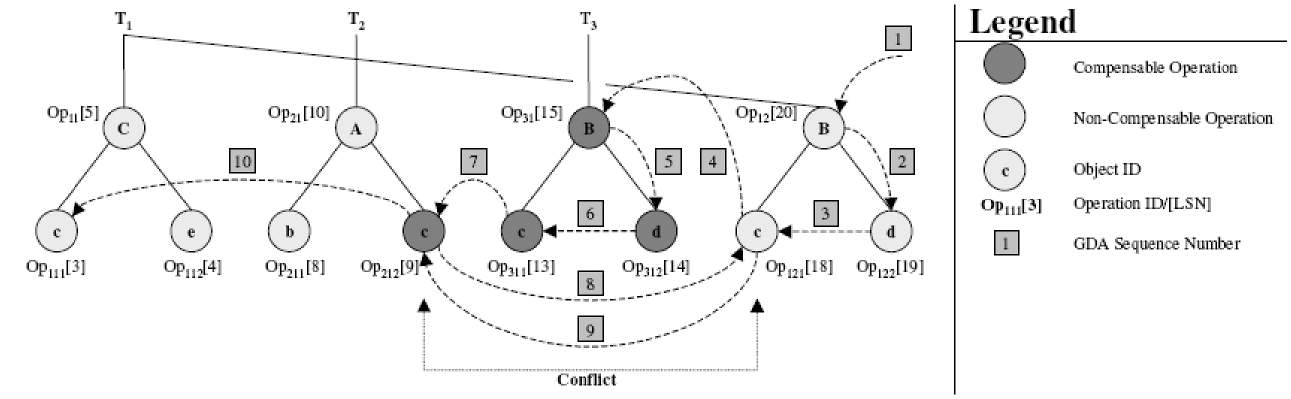
1. Process log records LSN_{20}, LSN_{19} and LSN_{18} without any undo since $T_1 \notin T_D \cup T_C$. The object undo list is updated as shown in Figure 9[Step 1].
2. Since $T_3 \in T_D$, the operations represented by log records LSN_{13}, LSN_{14} and LSN_{15} must be undone. The operation represented by LSN_{15} is compensable meaning that all three operations can be undone by applying a single compensation operation. The object undo list is updated as shown in Figure 9[Step 2]. Note, objects B and d are removed since their ObjectLastLSN pointers had *null* values.
3. Process log record LSN_9. Even though $T_2 \in T_D$, undo is postponed due to a conflict between Op_{212} and Op_{121}. T_1 is added to T_C and the object undo list is updated to ensure that Op_{121} is rolled back first (see Figure 9[Step 3]).
4. Process log record LSN_{18}, whose operation must be undone since $T_1 \in T_C$. The object undo list is updated as shown in Figure 9[Step 4].
5. Process log record LSN_{13}, which has previously been undone by this GDA invocation. Thus, simply update the object undo list (refer to Figure 9[Step 5]).
6. Process log record LSN_9 again, but this time operation Op_{212} can be undone since the conflicting operation (i.e. Op_{121}) has already been undone. The object undo list is updated as shown in Figure 9[Step 6].
7. Process log record LSN_3, whose operation must be undone since $T_1 \in T_C$. The object undo list is updated, which results in no entries being present.
8. Since there are no more log records to process, the GDA will update the Undone list as follows: Add operation identifiers 31, 121, 212, and 111. Neither operation Op_{311} nor Op_{312} appears in the Undone list since their effects were undone by a compensation operation that removed the effects of operation Op_{31}.

During the completion phase, the GDA algorithm releases all locks acquired during the initialisation phase and writes a GER log record. At this stage, no transaction in the deadlocked transaction set (T_D) needs to reacquire a lock on any object in the tie-break. Therefore, the deadlock has been successfully broken.

5 Conclusion

In this paper, we proposed an algorithm that solves the *'golden transaction' problem* present in the ARIES/ML (Schewe et al. 2000) recovery algorithm.

[5] This will invariably add to the overhead of the transaction processing system. As such, the merit of this refinement cannot be fully assessed without implementation and testing of the GDA algorithm with and without this refinement in place.



Object Table

ObjectID	ObjectLastLSN	PinCount
A	10	1
B	20	3
C	5	1
b	8	1
c	18	4
d	19	3
e	4	2

Figure 7: GDA Example

As a result, multi-level transaction management systems can now fully utilise the increased performance benefits coming with multi-level transaction support. Artificial restrictions are no longer placed on neither normal processing nor rollback processing.

The augmented ARIES/ML algorithm has the same performance characteristics during normal processing as the original ARIES/ML algorithm. Only in the event that the termination enhancement is being utilised, a small penalty has to be paid during normal processing. The small penalty corresponds to the maintenance of one additional field in the active transaction table. During rollback processing, however, the augmented ARIES/ML variant allows multiple rollbacks to process concurrently. This enhances not only the performance of rollback processing but also reduces the time transactions in normal processing mode have to wait until those objects that are accessed by transactions in rollback mode become available again for normal processing.

References

Beeri, C., Bernstein, P. A. & Goodman, N. (1989), 'A model for concurrency in nested transactions systems', *Journal of the ACM (JACM)* **36**(2), 230–269.

Drechsler, S. (1998), Kopplung des ARIES-recovery-systems mit hybriden mehrschichtenschedulern, Master's thesis, Department of Computer Science, Clausthal University of Technology, Germany. (in German).

Lomet, D. B. (1992), MLR: a recovery method for multi-level systems, *in* 'Proceedings of the ACM SIGMOD international conference on Management of data', ACM Press, pp. 185–194.

Mohan, C. (2004), 'ARIES family of locking and recovery algorithms', On the Internet at http://www.almaden.ibm.com/u/mohan/ARIES_Impact.html.

Mohan, C., Haderle, D. J., Lindsay, B. G., Pirahesh, H. & Schwarz, P. M. (1992), 'ARIES: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging', *ACM Transactions on Database Systems (TODS)* **17**(1), 94–162.

Riaz-ud-din, F. (2002), An implementation of the ARIES/ML recovery manager, Master's thesis, Department of Information Systems, Massey University, New Zealand.

Rothermel, K. & Mohan, C. (1989), ARIES/NT: a recovery method based on write-ahead logging for nested transactions, *in* 'Proceedings of the 15th international conference on Very large data bases', Morgan Kaufmann Publishers Inc., pp. 337–346.

Schewe, K.-D., Ripke, T. & Drechsler, S. (2000), 'Hybrid concurrency control and recovery for multi-level transactions', *Acta Cybernetica* **14**(3), 419–453.

Speer, J. (2005), Database recovery: Expanding the ARIES constellation, Master's thesis, Department of Information Systems, Massey University, New Zealand.

Weikum, G. (1986), A theoretical foundation of multi-level concurrency control, *in* 'Proceedings of the 5th ACM SIGACT-SIGMOD symposium on Principles of database systems', ACM Press, pp. 31–43.

Weikum, G. (1991), 'Principles and realization strategies of multilevel transaction management', *ACM Transactions on Database Systems (TODS)* **16**(1), 132–180.

Weikum, G., Hasse, C., Broessler, P. & Muth, P. (1990), Multi-level recovery, *in* 'Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems', ACM Press, pp. 109–123.

A Citation Analysis of the ADC 2006–2008 Proceedings, with Reference to the CORE Conference and Journal Rankings

Raymond Lister and Ilona Box
Faculty of Engineering and Information Technology
University of Technology, Sydney
Jones St. Broadway NSW 2007

raymond@it.uts.edu.au

Abstract

This paper compares the CORE rankings of computing conferences and journals to the frequency of citation of those journals and conferences in the Australasian Database Conference (ADC) 2006, 2007 and 2008 proceedings. The assumption underlying this study is that there should be a positive relationship between citation rates and the CORE rankings. Our analysis shows that CORE conference and journal rankings broadly reflect the ADC citations, but we note some anomalies. While these anomalies might be minor in the larger scheme of things, any anomalies need to be addressed, as the careers of individual academics may depend upon it. The concept of conference and journal rankings is probably here to stay, and this paper ends with some suggestions on how the rankings process should now evolve, so that it becomes more transparent.

Keywords: Citation Analysis, Excellence in Research for Australia (ERA), conference rankings, journal rankings.

1 Introduction

For several years, the Australian federal government has been developing a process for reviewing the quality and impact of publicly funded Australian research. The review was originally known as the Research Quality Framework, or simply RQF (DEST, 2007), but with a change of government some aspects of the review process changed, and the review process is now known as Excellence in Research for Australia, or simple ERA (ARC, 2008). As part of the RQF/ERA, the Computing Research and Education Association of Australasia (CORE) has developed a ranking scheme for computing-related conferences and journals (CORE, 2007).

Developing such a set of rankings is by no means straightforward. Most ranking systems include citations as a prominent factor. While there are indexes that record the number of citations for individual papers and for journals, only a small percentage of all computing papers are thus indexed. Since there was not an existing robust method for ranking conferences and journals, CORE formed committees that developed their own processes for ranking conferences and journals.

This paper evaluates the CORE rankings from the perspective of the Australasian database research community. Specifically, the paper poses the following research question: *do the conferences and journals cited most frequently in the three most recent ADC proceedings figure prominently in the CORE rankings*? More specifically, if a conference or journal is:

1) Ranked high by CORE and receives a high number of citations in the three ADC proceedings, then the ranking and the citation rate are consistent.
2) Ranked low by CORE and receives a low number of citations in the three ADC proceedings, then the ranking and the citation rate are consistent.
3) Ranked high by CORE but receives a low number of citations in the three ADC proceedings, then the interpretation of that data depends upon other factors:
    a) Such a highly ranked conference/journal may not be highly relevant to database research, so it would not be surprising that its citation rate within ADC proceedings is low.
    b) The conference/journal may be very new, and there simply hasn't been the opportunity for many papers from it to be cited yet.
    c) Otherwise, we would regard its high ranking as being inconsistent with its low ADC citation rate.
4) Ranked low by CORE but receives a high number of citations in the three ADC proceedings, then we would regard its low ranking as being inconsistent with its ADC citation rate.

Note that "inconsistent" does not mean "incorrect". There may be other factors that make the CORE ranking appropriate. Inconsistency merely indicates that further attention is warranted.

The ADC 2006, 2007 and 2008 conferences respectively contain 21, 17 and 18 full papers with citations, which is a total of 56 papers. To answer the above research question, all 1,332 citations in those 56 papers were examined, to see what conferences and journals were most frequently cited. Of the 1,332 citations, 362 (27%) were to journal papers and 711 (53%) were to conference papers. Table 1 shows a complete breakdown of the different types of citations in those ADC proceedings. (All tables appear at the end of the paper.)

2 Conference Citations and CORE Rankings

2.1 The CORE Conference Ranking System

The CORE website describes the broad structure and chronology of the conference rankings process (CORE, 2007b; CORE, 2007d). It began in late 2005. In the first few months, the Australian National University's Research Evaluation and Policy Project (REPP), using bibliometric analyses, created a master list of conferences. In March 2006, that master list was presented for discussion at a workshop of approximately 20 ICT researchers from a number of universities. After refinement of the list, it was released for consultation and feedback from the ICT community. In late 2006, a project reference group (of unspecified size and composition) produced the first draft of the conference ranking list. During 2007, consultation with the ICT community continued, and some changes were made to the list. At the time this paper was written, the most recent draft of the conference rankings was released in December 2007.

The CORE conference rankings are based upon four tiers, enumerated as A+, A, B, L and C (CORE, 2007c). There are two indications on the CORE website as to the significance of the various tiers. These two indicators are described in the following two subsections.

2.1.1 CORE Description of the Tiers

The following description of the conference tiers is taken verbatim from the CORE web site:

Tier A+: Typically a Tier A+ conference would be one of the very best in its field or subfield in which to publish and would typically cover the entire field/subfield. These are conferences where most of the work is important (it will really shape the field), where researchers boast about being accepted, and where attendees would gain value from attending even if they didn't have a paper themselves. Acceptance rates would typically be low and the program committee would be dominated by field leaders, including many from top institutions. Tier A+ conferences would be highly represented in the CV of a junior academic (assistant professor) aiming for tenure at a top 10 US university. These are the conferences where people from overseas congratulate you on getting in and you shout drinks to the research group.

Tier A: Publishing in a Tier A conference would add to the author's respect, showing they have real engagement with the global research community and that they have something to say about problems of some significance. Attending a Tier A conference would be worth travelling to if a paper was accepted. Typical signs of a Tier A conference are lowish acceptance rates and a program committee and speaker list which includes a reasonable fraction of well known researchers from top institutions (as well as a substantial number from weaker institutions), and a real effort by the program committee to look at the significance of the work.

Tier B: Tier B covers conferences where one has some confidence that research was done, so publishing there is evidence of research-active status (that is, there is some research contribution claimed, and a program committee that takes its job seriously enough to remove anything ridiculous or ignorant of the state of art), but it's not particularly significant. This is where PhD students might be expected to send early work; it also includes places whose main function is the social cohesion of a community. Typical examples would be regional conferences or international conferences with high acceptance rates, and those with program committees that have very few leading researchers from top international institutions.

Tier L: These are local conferences which may be important for the social cohesion of the local community and for networking. Many were "one off" but are included for historical reasons.

Tier C: All the rest.

2.1.2 DEST Publication Rates

The second indication on the CORE web site as to the significance of the various tiers is data from the Australian Government's Department of Education, Employment and Workplace Relations (DEWR) indicating approximate publication rates of Australian authors in each of the CORE ranks:

A+	6%	
A	27%	
B	31%	
U (sic)	29%	(meant to be C?)
C (sic)	6%	(meant to be L?)

The RQF/ERA processes have changed rapidly over the years of this project, and the information on the CORE web site has not always kept pace with those changes. While the CORE web site information above describes tiers "U" and "C", we suspect that those designations should be updated to "C" and "L" respectively.

2.2 ADC Conference Citations—Results

In the 56 papers of the three ADC proceedings surveyed, authors cited papers from 204 different conferences. Table 2 shows the number of citations to conferences within the four CORE tiers, and also to conferences not listed by CORE. The final row of Table 2 shows the DEST estimate of the publication rates of Australian authors in each of the conference ranks, which shows that—as one would expect—the rate at which ADC authors cite A and A+ conferences exceeds the frequency with which Australians publish in A and A+ conferences.

A striking feature of Table 2 is that almost one third (30%) of the conferences cited by ADC authors are not listed by CORE. However, it would be unreasonable to expect CORE to rank all those conferences, for at least two reasons: (1) for the purposes of the RQF/ERA, CORE was only asked to rank conferences and journals in which Australian-based academics had reported having recently published, and (2) as shown in Table 3, of the 204 different conferences cited, 128 of those conferences (62.7%) were cited in only paper in the ADC 2006–08 proceedings.

Table 2a is a comparison of the data from Table 2 for ADC with similar data from three other conferences that are part of the Australasian Computer Science Week. For details of the analysis of those other three conferences, see the analogous papers appearing in those respective conference proceedings (Lister & Box, 2008, 2009a, and

2009b). Table 2a shows that ADC has a higher percentage of citations to both A+ and A conferences than any of the other three conferences. Also, the percentage of unranked conferences cited in ADC (30%) is lower than for the other conferences—it is almost half the percentage of unranked conferences for ACE (55%).

Some conferences cited more than once were cited in only one paper. Also, the citation rate for a particular conference can be distorted by self-citation. Table 4 allows for these distortions, by (1) counting not the actual citations, but the number of different papers that cite a particular conference, and also by (2) excluding self-citations. Of the 176 conferences that received citations other than self-citations, over two thirds of those conferences (68.2%) were cited in only one paper and just under 90% (88.1%) of conferences were cited in three or less papers (i.e. an average of one paper or less per year for ADC2006–2008).

Table 5 lists the top 21 conferences, which are all cited in four or more papers, excluding self-citations. The columns show the CORE tier (column "Tier"), total number of citations to the conference (column "Cites"), total number of citations to the conference, excluding self-citations (column "CitesXSelf"), total number of papers that cited that conference (column "Papers"), and total number of papers that cited that conference, excluding self-citations (column "PapersXSelf"). The list is ordered on the last column (descending), then by tier, then by conference name. Of the 21 conferences listed in Table 5, two thirds (67%) are ranked as A+ by CORE, and 19 of the 21 conferences (90%) are ranked as either A+ or A.

Figure 1 is a log-log plot (to base e) of PapersXSelf versus the rank of the 21 conferences from Table 5 (i.e. ranked on PapersXSelf). This graph suggests that the distribution of the number of papers citing a particular conference is broadly consistent with the well known power law distribution for citations (Redner, 1998; Tsallis & de Albuquerque, 2000). Such power law distributions are often referred to as Zipf's Law. The slope of the line of best fit is -0.8. However, this plot is merely suggestive of a power law distribution, and not definitive, as the amount of data we have collected in this study is relatively low by the standards of citation analysis.

Table 5a shows five conferences with the word "database" in the conference titles, which were ranked highly by CORE but which received a low number of citations in the ADC 2006–2008 proceedings.

3 Discussion of Conference Rankings

Given the above results, we conclude that – in general, with caveats to follow – the citations to various conferences in the ADC 2006–08 proceedings are in general consistent with the CORE conference rankings. The remainder of this discussion focuses on the possible anomalies.

Authors of papers published in the WebDB and ER conferences might have some grounds for feeling aggrieved, as those are the only two conferences in Table 5 that are ranked lower than A. WebDB authors may be particularly aggrieved, as that conference appears in the middle of Table 5, but is only ranked as a "C" conference. By criteria 4 given in the introduction, the CORE ranking of WebDB is inconsistent with our ADC citation data. However, the rankings system is not based solely on citation data (from any source, let alone our analysis of only three ADC proceedings), and other factors may have reasonably led to the lower ranking of WebDB. However, an author of a WebDB conference paper should at least be able to find out CORE's reasons for ranking that conference so low, and currently CORE does not publish its reasons for the specific ranking of any conference (or journal).

An author of a paper published in ER may have fewer grounds for feeling aggrieved than an author of a WebDB paper. ER is ranked by CORE as a "B" conference, and it appears low in Table 5. Perhaps if we gathered citation data from a few more ADC conferences, we might then find that ER appeared even lower in an extended version of Table 5.

Surprisingly, ADC itself does not appear in Table 5. It received (non-self) citations in only two papers, in the ADC2006–2008 proceedings. In the analogous citation analysis of three other ACSW conferences—ACSC, AUIC and ACE—each of those conferences appeared in its respective Table 5 (Lister & Box, 2008, 2009a, and 2009b). If ADC authors are not citing each other's papers, then by whom are ADC papers intended to be read?

With regard to the five "database" conferences listed in Table 5a, and by criteria 3c given in the introduction, the CORE rankings for these five conferences are inconsistent with our ADC citation data. However, it does not necessarily follow that these CORE rankings are inappropriate, for three possible reasons: 1) Statistical fluctuation due to small sample size—for example, had only one more ADC paper cited a paper from DASFAA, then that conference would have been in Table 5; 2) these five may be niche conferences, and the focus of ADC, at least in the years 2006–2008, may have not been on the aspects of databases covered by these five conferences; and 3) citation rates are only one form of data that inform the CORE rankings.

Over two fifths (41%, see Table 2) of ADC citations to CORE ranked conferences are to conferences ranked B, L or C. That statistic challenges the premise of RQF/ERA conference rankings process—that the rank of a conference is a reliable proxy for the quality of all papers in that conference.

4 Journal Citations and CORE Rankings

4.1 The CORE Journal Ranking System

There are four tiers in the ERA journal ranking system—A*, A, B, and C. The Australian Federal Government's Australian Research Council (ARC, 2008) has indicated that, within each research discipline, the proportion of journals within each tier should be approximately:

```
A* - top 5%
A  - next 15%
B  - next 30%
C  - bottom 50%
```

Of the 834 journals ranked by CORE, as at June 2008, the percentages in each of the tiers is:

```
A* -  6% ( 47 journals)
A  - 18% (147 journals)
B  - 28% (233 journals)
C  - 49% (407 journals)
```

The CORE web site contains a "journal update template" which gives an indication of the type of information used by CORE to determine the ranking of a journal:

- The number of referees for each paper submitted to the journal.
- Whether the review process is blind, double blind, open, or performed by the editor.
- The acceptance rate.
- The composition of the editorial committee (e.g. the proportion that are leading researchers in the field, and whether they are from the premier institutions for the field).
- The quality of the papers presented (e.g. whether the work presented shapes the field, and whether the quality of papers is uniform or 'patchy').
- Whether the top researchers in the field publish in that journal.
- The citation rate for papers in that journal.

4.2 ADC Journal Citations—Results

In the three ADC proceedings surveyed, authors cited papers from 135 different journals. Table 6 shows the number of citations to journals within the four CORE tiers, and also to journals not listed by CORE. The final row of Table 6 shows the distribution across the tiers of all 834 journals ranked by CORE, which shows—as one would expect—that there is a bias among ADC authors toward citing the more highly ranked journals.

As with the analogous data for conferences, an immediately striking feature of Table 6 is that 33% of the journals cited in recent ADC papers are not ranked by CORE. However, as was the case with conferences, it would be unreasonable to expect CORE to rank all those journals, for at least two reasons: (1) for the purposes of the RQF/ERA, CORE was only asked to rank journals in which Australian-based academics had reported having recently published; and (2) as shown in Table 7, over one half (59%) of journals cited in the three ADC proceedings were cited only once. Also, among the journals cited in ADC papers are some that—while the citation may be germane to the paper in which the citation is made—one would not realistically expect the journal to be ranked by the CORE committee (e.g. *Drug Discovery Today*, and *Methods in Enzymology*).

Some journals cited more than once were cited in only one paper. Also, the citation rate for a particular journal can be distorted by self-citation. Table 8 allows for these distortions, by (1) counting not the actual citations, but the number of different papers that cite a particular journal, and also by (2) excluding self-citations. Two thirds (66%) of journals cited were cited in only one paper (excluding self citations) and 90% of the journals were cited in three or fewer papers (i.e. 90% of journals were cited, on average, in one or fewer papers per year over the three years of ADC proceedings analysed).

Table 9 lists the 20 journals cited by three or more papers, excluding self-citations. The columns show the CORE tier (column "Tier"), total number of citations to the journal (column "Cites"), total number of citations to the journal, excluding self-citations (column "CitesXSelf"), total number of papers that cited that journal (column "Papers"), and total number of papers that cited that journal, excluding self-citations (column "PapersXSelf"). The list is ordered on the last column (descending), then by tier, then by journal name. Of the 20 journals listed in Table 9, 6 (30%) are ranked as A* by CORE, and 15 (75%) are ranked as either A* or A.

Figure 2 is a log-log plot (to base e) of PapersXSelf versus the rank of the 20 journals from Table 9 (i.e. ranked on PapersXSelf). This graph suggests that the distribution of the number of papers citing a particular journal is broadly consistent with a power law distribution, as was also the case for conferences. The slope of the line of best fit is -0.7. However, this plot is merely suggestive of a power law distribution, as the amount of data we have collected in this study is relatively low by the standards of citation analysis.

5 Discussion of Journal Rankings

Given the above results, we conclude that—in general, with some caveats to follow—the citations to various journals in the ADC 2006–08 proceedings are broadly consistent with the CORE journal rankings.

The ranking of *SIGMOD Record* as a "C" journal is consistent with CORE's policy of ranking all ACM Special Interest Group (SIG) newsletters as "C" (personal communication with CORE). However, such a ranking for *SIGMOD Record* is inconsistent with its appearance at the top of Table 9.

The *Communications of the ACM* (CACM) appears second in Table 9, which seems inconsistent with CORE's ranking of it as a "B" journal. The ranking of CACM generated much discussion within CORE. In early drafts of the rankings, CACM was deliberately not ranked at all, since many within CORE argued that CACM is a magazine, not a research journal (personal communication with CORE). In the analogous citation analysis of three other ACSW conferences—ACSC, AUIC and ACE—CACM ranked first, first and third respectively (Lister & Box, 2008, 2009a, and 2009b). Polites and Watson (2008) used a more elaborate citation analysis technique than what we have used in this paper, and they found CACM to be a highly influential publication within Information Systems research—even more influential than MIS Quarterly. On this citation data in isolation, CACM's "B" ranking is hard to justify.

The non-ranking of the *SIGIR Forum,* which appears in Table 5*,* is an oversight by CORE, as it is (like *SIGMOD Record)* an ACM SIG newsletter.

Approaching one half (44%) of ADC citations to all ranked journals are to journals ranked B or C (see Table 6). As was also the case with the analogous conference statistic, this 44% statistic challenges a premise of the RQF/ERA – that the rank of a journal is a reliable proxy for the quality of all papers in that journal.

Table 6a is a comparison of the data from Table 6 for ADC with similar data from three other conferences that are part of the Australasian Computer Science Week

(Lister & Box, 2008; 2009a; 2009b). Table 6a shows that the 38% cumulative percentage of ADC citations in Tiers A* and A is higher than the 29% of ACSC and much higher than the same figure for AUIC and ACE. The percentage of unranked journals cited in ADC (33%) is much lower than the percentage for the other three conferences.

6 Age of Citations

The ERA specifies an audit period of six years. An examination of the age of citations in the ADC 2006–2008 proceedings shows that 70% of ADC conference citations are to conferences held in the year 2000 or more recently, and 55% of ADC journal citations are to papers that appeared in the year 2000 or more recently. The analogous percentages for ACSC are similar (69% and 50% respectively), but the analogous percentages for AUIC are much lower (25% and 25% respectively).

7 Discussion: Scholarship and Discourse

The schedule for developing the CORE rankings has been driven largely by the federal government's timetable for the RQF/ERA, which was faster than many in academia would have liked. Under such unfavourable circumstances, it was unavoidable that the ranking would be a relatively opaque committee/executive process.

While the rankings themselves may change, the concept of conference and journal rankings is here to stay. Furthermore, this ranking scheme will be used for purposes beyond the federal government's ERA exercise. For example, the rankings will become a routine part of applications for promotion in Australian universities.

It is now appropriate to consider the long-term strategy for the rankings. It is not in the best long-term interests of scholarship that the ranking remains an opaque committee/executive process. Scholarship would be better served by a more transparent process that allows for the ranking process itself to be improved, via open scholarly debate. As a first move toward developing such a scholarly process for routinely revising the rankings, we suggest a three-step process, described below.

7.1 All Policy and Data Should be Public

CORE have not yet documented their criteria for the journal rankings. Also, while there is a description of the conference rankings process on the CORE web site, it is not detailed. CORE should further document the criteria for journal and conference rankings.

The data for each conference and each journal (e.g. acceptance rates) should be made public, so that the computing community can check that the data is correct. Currently, given the high number of conferences and journals that have been ranked, and the short time in which the rankings were done, it is likely that a small number of journals and conferences have been ranked inappropriately because of bad data.

By using Web 2 technologies, CORE could make its ranking data public, and also push much of the effort for data acquisition and data cleaning onto the computing academic community.

7.2 Formal Models

A formal model—perhaps a points system—should be adopted for assigning preliminary rankings to conferences and journals. Such a model would make the ranking process far more transparent.

A formal model would offer a mechanism for providing a preliminary ranking for new conferences and journals. Currently, the conferences and journals that have been ranked are conferences and journals in which Australian-based academics have published in recent years. In the absence of a formal model, and irrespective of what the federal government and CORE may have intended, it is likely that unranked conferences and journals will be regarded as inferior; or at best dubious. The first Australian academic to publish in an unranked journal will have difficulty in establishing the quality of that journal to a promotions board, and the first Australian academic who has a paper accepted by an unranked conference will have difficulty in making a case to his/her department head for travel funding. The absence of a formal model will stifle Australian academics working in emerging research areas of computing.

7.3 A Documented Manual Review

Formal models will not capture the complexities of ranking. It is therefore appropriate that CORE continue to appoint committees that review the outputs of a formal model. When such a committee elects to manually alter the ranking from that assigned by the formal model, the reasons for doing so should be made public.

8 Conclusion

From our analysis of the CORE conference rankings, we conclude that the existing rankings are broadly consistent with the frequency of citation to conferences and journals in the three most recent ADC conferences. Our analysis shows that CORE conference and journal rankings broadly reflect the ADC citations, but we have noted some anomalies. It is important that anomalies be resolved, as the careers of individual academics may depend upon it.

Apart from the traditional intellectual skills associated with each academic discipline, all successful academics have found it necessary to acquire other skills—project management, and grant writing, to name just two of those skills. The RQF/ERA government initiatives have added a new required skill for the successful Australian academic—the ability to understand issues in bibliometrics well enough to participate in the discourse on conference/journal rankings, particularly with regard to the ranking of their preferred conferences and journals. If we do not master bibliometrics to that degree, then bibliometrics will master us.

The process of ranking conferences and journals is as complex as it is vexing. This aim of this paper is merely to begin a scholarly discourse within the ADC community on the CORE rankings. This paper is certainly not intended as the final word. Meanwhile, careers will rise and fall on the decisions made by the CORE ranking committees. It is therefore vital that the CORE ranking processes be open to informed discussion and peer review—why should we settle for a ranking process that is less rigorous in its scholarship than what we demand from the research published in a highly ranked conference or journal?

References

ACM (2007) *Policy on Pre-Publication Evaluation*. http://www.acm.org/pubs/prepub_eval.html / [Accessed October 2007].

ARC (2008) *Excellence in Research for Australia Initiative, Consultation Paper*. http://www.arc.gov.au/pdf/ERA_ConsultationPaper.pdf [4th September 2008].

CORE (2007a). *Final 2007 Australian ranking of ICT conferences.* [Internet]. Computing Research and Education. Available from: http://www.core.edu.au/rankings/Conference%20Ranking%20Main.html [4th September 2008].

CORE (2007b). *The initial process for an Australian ranking of ICT conferences.* [Internet]. Computing Research and Education. Available from: http://www.core.edu.au/rankings/ranking-process.pdf [4th September 2008].

CORE (2007c). *Tiers for the Australian ranking of ICT conferences.* [Internet]. Computing Research and Education. Available from: http://www.core.edu.au/rankings/conf-tiers.pdf [4th September 2008].

CORE (2007d). *The updating process for the Australian ranking of ICT conferences.* [Internet]. Computing Research and Education. Available from: http://www.core.edu.au/rankings/update-process.pdf [4th September 2008].

CORE (2008). *Journal rankings.* [Internet]. Computing Research and Education. Available from: http://www.core.edu.au/ [4th September 2008].

DEST (2007) Research Quality Framework: Assessing the quality and impact of research in Australia. http://www.dest.gov.au/sectors/research_sector/policies_issues_reviews/key_issues/research_quality_framework/#RQF_Implementation_2007 [Accessed Oct. 2007]

Lister, R. and Box, I. (2008). *A citation analysis of the ACE2005 -2007 proceedings, with reference to the June 2007 CORE conference and journal rankings*. In Proc. Tenth Australasian Computing Education Conference (ACE 2008), Wollongong, NSW, Australia. CRPIT, 78. Simon and Hamilton, M., Eds., ACS. 63-68. http://crpit.com/Vol78.html

Lister, R. and Box, I. (2009a). *A citation analysis of the ACSC 2006–2008 proceedings, with reference to the CORE conference and journal rankings*. In Proc. 32nd Australasian Computer Science Conference (ACSW 2009), Wellington, New Zealand, January 2009. CRPIT, 91. Bernard Mans (Ed.), ACS.

Lister, R. and Box, I. (2009b). *A citation analysis of the AUIC 2006–2008 proceedings, with reference to the CORE conference and journal rankings*. In Proc. 10th Australasian User Interface Conference (AUIC 2009), Wellington, New Zealand, January 2009. CRPIT, 93. Calder, P. and Weber, G., Eds., ACS.

Polites, G. L. and Watson, R. T. 2008. The centrality and prestige of CACM. *Commun. ACM* 51, 1 (Jan. 2008), 95-100. http://doi.acm.org/10.1145/1327452.1327454 [Accessed September 2008]

Redner S. (1998) How popular is your paper? An empirical study of the citation distribution, *Eur. J. Phys. B*, 4, 131-134. http://xxx.lanl.gov/abs/cond-mat/9804163

Tsallis C., de Albuquerque M. P. (2000) Are citations of scientific papers a case of nonextensivity ?, *Eur. Phys. J. B*, 13, 777-780, http://tsallis.cat.cbpf.br/biblio.htm

Types of Citation	ADC2006		ADC2007		ADC2008		Total	
	No. Citations	%age Citations	No. Citations	%age Citations	No. Citations	%age Citations	No. Citations	%age Citations
Journal	143	31%	111	23%	108	27%	362	27%
Conference	212	46%	272	57%	227	57%	711	53%
Book or Chapter	54	12%	34	7%	24	6%	112	8%
Web Page	27	6%	41	9%	14	4%	82	6%
Unpublished Report	15	3%	9	2%	13	3%	37	3%
Unpublished Thesis	10	2%	4	1%	11	3%	25	2%
Patent	0		2	<1%	0		2	<1%
Map	0		1	<1%	0		1	<1%
Total	461		474		397		1332	

Table 1: Number of different types of citation in the ADC2006, ADC2007 and ADC2008 proceedings.

	Tier					
	A+	**A**	**B**	**L(ocal)**	**C**	**Not Listed**
Number of conferences cited in ADC	30	53	38	2	19	60
Percentage of conferences cited in ADC	15%	26%	19%	1%	9%	30%
Percentage of listed conferences cited in ADC	21%	37%	27%	1%	13%	–
DEST publication rates of Australian authors in each of the CORE ranks	6%	27%	31%	6%	29%	

Table 2: Number of citations in the ADC proceedings to conferences in each CORE tier, and to conferences not listed by CORE.

	Tier					
	A+	**A**	**A+ & A**	**B**	**C**	**Not Listed**
Percentage of conferences cited in ADC (as in Table 2)	15%	26%	41%	19%	9%	30%
Percentage of conferences cited in ACSC	14%	22%	36%	14%	8%	42%
Percentage of conferences cited in AUIC	12%	19%	31%	14%	4%	50%
Percentage of conferences cited in ACE	5%	11%	16%	21%	9%	55%

Table 2a: A comparison of citations to conferences in each CORE tier in the ADC proceedings with three other ACSW conferences.

	Number of Citations																		
	1	2	3	4	5	6	7	8	9	10	11	13	16	19	30	48	81	85	**Total**
Number of Conferences	128	28	17	6	3	2	2	3	2	1	2	3	1	2	1 SIGIR	1 ICDE	1 VLDB	1 SIGMOD	204
Cumulative Percentage	62.7	76.5	84.8	87.7	89.2	90.2	91.2	92.6	93.6	94.1	95.1	96.6	97.1	98.0	98.5	99.0	99.5	100%	

Table 3: Number of conferences receiving various numbers of citations from the ADC2006–2008 proceedings

	Number of Different Papers that Cite a Particular Conference													
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**22**	**29**	**Total**
Total No. Conferences	120	27	8	4	4	2	3	2	1	1	1	1	2	176
Cumulative Percentage	68.2	83.5	88.1	90.3	92.6	93.8	95.5	96.6	97.2	97.7	98.3	98.9	100%	
No. Tier A+	9	5	FOCS CHI	JCDL SODA ICML	ICDM IJCAI AAAI	STOC		SIGKDD WWW	SIGIR		PODS	ICDE	SIGMOD VLDB	
No. Tier A	37	5	4	TREC		DEXA	EDBT ICDT			CIKM				
No. Tier B	20	7			ER									
No. Tier C	11	4					WebDB							
No. unlisted	43	6	2											

Table 4: Number of conferences receiving citations, excluding self-citations, from various numbers of papers in the ADC2006–2008 proceedings. For an explanation of the conference acronyms, see Table 5.

Conference	Tier	Cites	CitesXSelf	Papers	PapersXSelf
SIGMOD: ACM Special Interest Group on Management of Data Conference	A+	85	83	30	29
VLDB: International Conference on Very Large Databases	A+	81	80	30	29
ICDE: IEEE International Conference on Data Engineering	A+	48	44	22	22
PODS: ACM SIGMOD-SIGACT-SIGART Conference on Principles of Database Systems	A+	19	19	11	11
CIKM: ACM International Conference on Information and Knowledge Management	A	19	16	10	10
SIGIR: ACM International Conference on Research and Development in Information Retrieval	A+	30	30	9	9
SIGKDD: ACM International Conference on Knowledge Discovery and Data Mining	A+	16	16	8	8
WWW: International World Wide Web Conference (ACM)	A+	13	12	9	8
EDBT: Extending Database Technology	A	8	8	7	7
ICDT: International Conference on Database Theory	A	7	7	7	7
WebDB: International Workshop on the Web and Databases	C	8	8	7	7
STOC: ACM Symposium on Theory of Computing	A+	8	8	6	6
DEXA: International Conference on Database and Expert Systems Applications	A	9	6	9	6
ICDM: IEEE International Conference on Data Mining	A+	13	11	5	5
IJCAI: International Joint Conference on Artificial Intelligence	A+	10	10	5	5
AAAI: National Conference of the American Association for Artificial Intelligence	A+	7	7	5	5
ER: International Conference on Conceptual Modelling	B	5	5	5	5
JCDL: ACM/IEEE-CS Joint Conference on Digital Libraries	A+	6	6	4	4
SODA: ACM/SIAM Symposium on Discrete Algorithms	A+	9	9	4	4
ICML: International Conference on Machine Learning	A+	5	5	4	4
TREC: Text Retrieval Conference	A	13	13	4	4

Table 5: All conference proceedings cited by four or more papers (excluding self-citations) in the ADC2006, 2007 and 2008 proceedings. The columns show the CORE tier ("Tier"), total number of citations to the conference ("Cites"), total number of citations to the conference, excluding self-citations ("CitesXSelf"), total number of papers that cited that conference ("Papers"), and total number of papers that cited that conference, excluding self-citations ("PapersXSelf"). The list is ordered (descending) on the last column, then by tier.

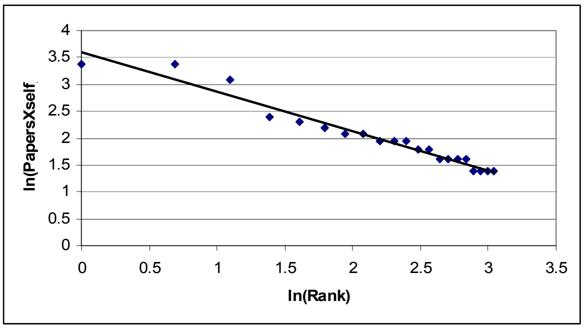


Figure 1: A plot of the natural logarithm of PapersXSelf versus the natural logarithm of the rank of the 21 conferences from Table 5, based on PapersXSelf.

Conference	Tier	Cites	CitesXSelf	Papers	PapersXSelf
DASFAA: DB Systems for Advanced Apps	A	3	3	3	3
SSDBM: International Conference on Scientific and Statistical Data Base Management	A	2	2	2	2
DOOD: Deductive and Object-Oriented Databases	A	1	1	1	1
PKDD: European Conf. on Principles and Practice of Knowledge Discovery in Databases	A	3	1	3	1
SSTD: International Symposium on Spatial Databases	A	0	0	0	0

Table 5a: Highly ranked database conferences with a low number of citations in ADC 2006–2008 proceedings.

	Tier				
	A*	A	B	C	Not Listed
Number of journals cited in ADC	19	32	28	12	44
Percentage of journals cited in ADC	14%	24%	21%	9%	33%
Percentage of ranked journals cited in ADC	21%	35%	31%	13%	–
CORE Overall Percentages (n=834)	6%	18%	28%	49%	

Table 6: The distribution of ADC citations across the CORE journal tiers.

	Tier					
	A*	A	A* & A	B	C	Not Listed
Percentage of journals cited in ADC (as in Table 6)	14%	24%	38%	21%	9%	33%
Percentage of journals cited in ACSC	13%	16%	29%	9%	4%	58%
Percentage of journals cited in AUIC	7%	7%	14%	16%	8%	61%
Percentage of journals cited in ACE	5%	4%	9%	5%	1%	85%

Table 6a: A comparison of the ADC journal citation distribution with three other ACSW conferences.

	Number of Citations														
	1	2	3	4	5	6	7	8	10	11	12	14	19	22	Total
Number of Journals	80	21	12	6	3	4	1	1	1	2	1	1	1	1	135
Cumulative Percentage	59	75	84	88	90	93	94	95	96	97	98	99	99	100	

Table 7: Number of journals receiving various numbers of citations from the ADC2006–2008 proceedings

	Number of Different Papers that Cite a Particular Journal										
	1	2	3	4	5	6	8	10	12	15	17	Total
Total No. of Journals	81	22	8	3	3	1	1	1	1	1	1	123
Cumulative Percentage	66	84	90	93	95	96	97	98	98	99	100	
No. Tier A*	21	6	ML	PAMI	TOIS	VLDB	CSUR		TODS			
No. Tier A	6	2	4	DPD CN	DMKD IS			TKDE				
No. Tier B	18	4								CACM		
No. Tier C	4	5	IPM								MOD	
No. unlisted	32	5	2									

Table 8: Distribution of citations to journals, excluding self-citations, from ADC2006–2008 papers.

Journal	Tier	Cites	CitesXSelf	Papers	PapersXSelf
MOD: SIGMOD Record	C	22	22	17	17
CACM: Communications of the ACM	B	19	19	15	15
TODS: ACM Transactions on Database Systems	A*	14	14	12	12
TKDE: IEEE Transactions on Knowledge and Data Engineering	A	11	11	10	10
CSUR: ACM Computing Surveys	A*	11	11	8	8
VLDB: The VLDB Journal	A*	8	8	6	6
TOIS: ACM Transactions on Information Systems	A*	10	10	5	5
DMKD: Data Mining and Knowledge Discovery	A	5	5	5	5
IS: Information Systems	A	7	7	5	5
PAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence	A*	6	6	4	4
DPD: Distributed and Parallel Databases	A	6	6	4	4
CN: Computer Networks	A	4	4	4	4
ML: Machine Learning	A*	5	5	3	3
MS: Multimedia Systems	A	3	3	3	3
CJ: Computer Journal	A	3	3	3	3
TCS: Theoretical Computer Science	A	4	3	4	3
TOIT: ACM Transactions on Internet Technology	A	3	3	3	3
IPM: Information Processing and Management	C	5	5	3	3
SIGIR: ACM SIGIR Forum	—	3	3	3	3
CSVT: IEEE Transactions on Circuits and Systems for Video Technology	—	12	11	3	3

Table 9: All journals cited by three or more papers (excluding self-citations) in the ADC2006, 2007 and 2008 proceedings. The columns show the CORE tier ("Tier"), total number of citations to the journal ("Cites"), total number of citations to the journal, excluding self-citations ("CitesXSelf"), total number of papers that cited that journal ("Papers"), and total number of papers that cited that journal, excluding self-citations ("PapersXSelf"). The list is ordered (descending) on the last column, then by tier, then by journal name.

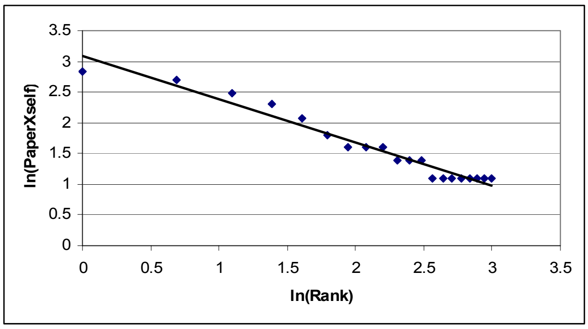


Figure 2: A plot of the natural logarithm of PapersXself versus the natural logarithm of the rank of the 20 journals from Table 9, based on PapersXself.

ActiveTags: Making Tags More Useful Anywhere on the Web

Stephan Hagemann **Gottfried Vossen**

European Research Center for Information Systems (ERCIS)
University of Münster
Leonardo-Campus 3, 48149 Münster, Germany,
Email: {shageman,vossen}@uni-muenster.de

Abstract

Tags in social tagging systems store meaning for the taggers who have entered them, and other users often share this understanding. The result of this, a *folksonomy*, is typically used in several ways, including information retrieval and clustering, serendipitous information access, or visualization of folksonomic characteristics. For these uses tags work pretty well; however, the ambiguity of tags makes it difficult to use them for more than searching and browsing. This paper introduces examples of current programmatic support in the form of mashups and highlights its shortcomings. It identifies several types of tags based on their structure and language, and discusses how these types support programmatic uses. The main part is the presentation of the ActiveTags system, a browser extension with supporting server infrastructure. Using it the same community process that creates a folksonomy can be used to enhance tags with programmatic meaning. Users are enabled to create reliable mashups based on tags. Effectively, this leads to customized views of Web pages with tagged content. ActiveTags naturally increases the usability of social tagging systems and further extends the notion of user-generated content.

Keywords: Social tagging; mashups; data as a service; information retrieval, filtering, and dissemination.

1 Introduction

Social tagging systems are part of numerous sites all over the Internet (Hammond et al. 2005, Lund et al. 2005, Marlow et al. 2006) and they are a central component in current Web 2.0 developments of the social Web (Hinchcliffe 2006). The objects that are tagged are quite diverse, ranging from bookmarks, photos and videos to products. The systems all share the fact that users can freely choose tags to be attached to objects, they can typically see tags that others have attached, and they can subsequently use all these to search and browse through the resources on a site. While this works quite well, the ambiguity of tags generally prevents more specific uses – the meaning of a tag is obvious to its creator but not necessarily to others. There are some examples of tags being used as a basis for mashups, which then specifically and effectively support linkages between information source and operations on their contents. Yet currently, these examples are few, have a very limited scope, and are implemented and enforced by the site operator. The ActiveTags system we propose in this paper allows users to implement their own mashups based on tags and use mashups of other users; tags can be used regardless of their Web location, thereby effectively allowing Web pages to be customized and enhanced by their users.

Two examples of support for specific tags are present on Flickr, a photo sharing site which allows the owners of photos to tag pictures, and to share them with friends depending on settings. So-called *machine tags* were introduced in January 2007 (Straup Cope 2007), which marked the beginning of special treatment of tags of the form "namespace:predicate=value." More functions are supported for machine tags when the Flickr API is used (flickr.com/services/api/flickr.photos.search.html). Earlier, the site had begun to support two sets of specific tags: geotags and event tags: (1) Geotags are a combination of three tags that together indicate the presence of a geotag and encode a geographic location. Two of the tags are machine tags, which encode the longitude and the latitude and can thus be used to position objects on a map. The site allows these two tags to be exported into structured data fields for geographic information, so that the pictures are automatically shown on maps also. (2) Tags with the prefix "upcoming:event=" are interpreted as links to events on Upcoming.org, with the effect that a link to the corresponding event is included in the page.

These two tag types essentially create mashups of Flickr content with related external content, and they are great features of tagging on the Flickr site. Similar examples are present on other sites, e.g., Delicious.com uses a special tag to send bookmarks from one user to another. While these specific forms of tag support are reasonable steps towards a use of tags for accessing relevant related information, they have several shortcomings:

Nature of tagging. The meaning of the tags is effectively enforced by the functions connected. An essential aspect of folksonomic tagging is that the vocabulary is free. In general, it should be possible for different understandings to coexist.

Flexibility. The tags that were chosen to be supported have been chosen by the site; extensions are only possible if the site operator implements them. Yet new understandings continuously arise in social tagging systems, and they should not depend on site operators to be effectively supported.

Note that for Flickr and Delicious this does not hold for uses through their APIs, where new functions can be implemented by users, although not within the confines of the original site and not without certain programming skills.

Scope. The tags we have discussed above only work on individual sites, although the outlined forms of tagging occur on other Web sites as well. It should be left to the users to decide what the scope of their tag usage and subsequent mashups is.

Users. As it is the users who generate tags, define their meanings, and make use of them, they have to be taken into account when it comes to programmatic support.

It is these shortcomings that the ActiveTags system is designed to overcome; corresponding support is the guideline for our development. Users should have a wide range of possibilities when it comes to selecting which interpretations they subscribe to. This includes which mashups they want to use, the support should be flexible so that new interpretations (mashups) can be included as the use of tags evolves, and it should be up to the users to define in which contexts interpretations should lead to actions (and mashups be shown). These contexts may, for example, include settings based on Web pages, time, or a user's social network. We have implemented ActiveTags as a browser extension to enhance a user's browsing experience based on the above principles.

The contribution of this paper is twofold: First, we will show how the ActiveTags system is designed in order to fulfill the principles mentioned above, and second we will show how the ActiveTags browser extension technically works, explaining how mashups can be created based on tags. To this end, the organization of this paper is as follows: Section 2 analyzes the current practice of tagging. Section 3 outlines the ActiveTags system design and introduces our prototype. Section 4 discusses which research findings and which practical developments are relevant for our work. We give our conclusions as well as an outlook in Section 5.

We are currently conducting an evaluation of our approach and the prototype implementation. Since that work is still ongoing, we have opted for presenting our concept independently here. We will report on the results of the evaluation in an upcoming paper.

2 On the Practice of Tagging

We have analyzed random tags from Flickr to evaluate what types of tags occur and now discuss whether these can be used to support mashups effectively. As opposed to other analyses such as Golder & Huberman (2006) we have not looked at the roles tags play, but at their syntax and whether they are from natural language dictionaries.

Flickr has several ways of storing tags. We have accessed the so-called *raw* version of a tag, which can be easily extracted via the Flickr API. Its disadvantage is that whitespaces contained in tags are not present in this form; thus, if tags are comprised of multiple words, these cannot be detected. 45,924 photos with at least one tag were randomly extracted. This has returned the tags of photos from 24,375 distinct users, with 490,561 tags in total or 4.43 tags per photo on average. We have analyzed whether tags belong to one of the four languages English, French, Spanish, or German, which we believe are currently the most popular languages on Flickr. This has been done by checking them against the respective aspell dictionaries (see `aspell.net`). We have divided all remaining tags into "machine tags" and "other tags." The results are summarized in Table 1.

Type	Absolute	Percentage
Language tags	244,692	49.88
English	222,145	45.28
French	68,236	13.91
German	61,069	12.45
Spanish	48,172	9.82
Machine tags	7,978	1.63
Other tags	237,891	48.49
All Tags	490,561	100.00

Table 1: Types of tags (based on all occurrences).

The largest group, almost 50% of all tags, contains tags from natural languages. This percentage is likely to be even higher (as we will see below) when one takes into account more languages and treats multiple-word tags differently. Machine tags make up for only 1.63% of the tags found. Together with the number for tags per photo, we can assume that machine tags are present roughly for 1 out of every 14 photos.

Table 2 contains the five most popular tags from each of the three categories. Due to their structure, machine tags, as we have seen above, can be used effectively for storing structured data in tags. In Table 2 we have based our frequency calculation on the name space and property parts of the tag only. As can be seen, geotags are by far the most popular machine tags, which is most likely due to the fact that these tags are supported by the Flickr site. We take this as evidence that programmatic support increases the incentive to use tags.

The question now is: Are we dealing with less than two percent of all tags, or can any of the other tags be used for building mashups as well?

An example of a mashup that uses ordinary tags to achieve something similar to geotagging is fotoland.us. It uses tags as the basis for a mashup of photos from Flickr and the maps provided by Google (`maps.google.com`). Instead of geotags it uses the names of countries and cities, but can also handle additional tags added by its users. Although the site often achieves good results, there tend to be photos in result sets where the tags have been interpreted the wrong way: For example, for the German city of *Essen*, which is also the German word for *food*, it is quite common to find photos that were actually taken in other cities, but that show food, people having food, or something similar.

Although this may be rare, problems with cities that have the same name are very common. As anecdotal evidence, take Athens, Texas. If this town were only tagged with "Athens" most pictures that get connected will actually be from Greece. If one tries to disambiguate the tag by tagging "Athens, US", the results get better, leading to some pictures from the city in Texas, but also pictures from Greece with people on them. Here the meaning of "US" was obviously not the country but *us* as in "we are in the picture." Finally, tagging "Athens, Texas" yields very good results. So, with proper disambiguation even common terms can be used as a reliable basis for mashups.

The five most popular tags in our sample which belong into the "other" category show what types of tags this category contains: Obviously there are abbreviations (bw - short for black & white photography), portmanteau words (geotagged - see above), tags composed of multiple words ("a big fave" – with "fave" being a short form of favorite, and "black and white"), and numbers (2006 - most probably denoting the year). Portmanteau words are commonly used to

#	Dictionary	Machine	Other
1	portrait (1333)	geo:lat (2455)	bw (3089)
2	canon (1250)	geo:lon (2396)	geotagged (1852)
3	me (1238)	flickr:user (422)	abigfave (1476)
4	nikon (1218)	dc:identifier (418)	blackandwhite (1110)
5	bravo (1120)	xmlns:dc (418)	2006 (1070)

Table 2: Five most popular tags by category.

uniquely identify events. Often used for conferences, examples include "barcampcologne", "barcampsydney07", and "drupalconbarcelona2007." For example, searching for the last term yields photos, slides, blog entries etc. related to the Drupal Conference 2007 in Barcelona.

We conclude that there are several groups of tags based on syntax which can reliably be used as metadata. The quality of this data is in the hands of the users. It is our hope that with the support of ActiveTags the awareness for good metadata will increase, as will the quality of the metadata.

3 The ActiveTags System

3.1 Tag-mashup lifecycle

The introduction of tag support for maps and events on Flickr was preceded by discussions in the communities of Flickr and other sites (Straup Cope 2007), and by external developments of services supporting the tags (Andrews 2006). As such, the implementation of terms depended on the process of the term development from inhibition to common understanding. The mashup development was then taken care of by the developers behind the site. In order to support a community in going through similar processes and the system to react flexibly, users have to be empowered to do so. They have to be able to define their understandings of tags, must be able to implement mashups, and to control the connection of the two.

The goal of ActiveTags is to support what could be called a "tag-mashup lifecycle" as depicted in Figure 1: (1) The starting point is an individual interest in expressing information which is not explicitly available. In this stage there is no common understanding of pragmatics. (2) The ActiveTags system allows the understanding to be expatiated through the implementation of a mashup. (3) The understanding spreads from individual to common, i.e., over the social network of a user. (4) This stimulates new terms to arise, which in turn spawns the creation of new mashups by restarting the cycle for other terms. This cycle is not to be made mandatory, but transitions between the steps are to be made as efficient as possible.

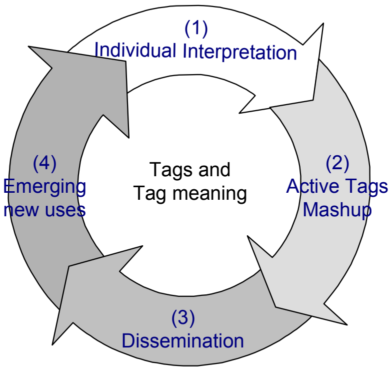


Figure 1: ActiveTags dissemination of mashups.

3.2 Building a tag-based mashup system

The four principles we have laid out above guide the design and implementation of the ActiveTags system; these were accordance to the nature of tagging, flexibility in control, user-defined scope, and explicit user modeling. The scope requirement has two aspects: to technically allow for a maximal scope, while enabling the user to restrict it. At this point we want to enable users to employ tags over the entire Web. Focusing on the Web does leave out programs that also use tagging (such as file tagging in Mac OS X and Microsoft Vista), but are not part of the Web. However, in doing so, we can rely on HTML as the dominant language for transporting content, and on the browser as the dominant way of access to the Web. Because of this, we have implemented ActiveTags as a Firefox extension that interacts with the Web pages a user visits to enhance them with tag-based mashup support. As a consequence, the design that is presented below is concerned with the Web context, but could be easily formulated for other contexts as well. Let us first look at an example of the intended user experience before we discuss the design of the ActiveTags system and its browser extension.

3.2.1 A sample page enhanced by ActiveTags

To exemplify the feasibility of our approach we have re-implemented the two functions which, as we have mentioned earlier, are available on Flickr as ActiveTags mashups. Figure 2 shows the screenshot of a Web page viewed in Firefox with the ActiveTags extension enabled and its results highlighted.

The Web page is a typical detailed photo page on the Flickr site. The dashed red lines indicate what the ActiveTags extension is working on. Tags are present on this page in the right column (1). They have been highlighted, indicating that ActiveTags has recognized them. As can be seen, the photo has been geotagged and tagged with an Upcoming.org Id. Flickr neither shows a map nor a link to one because geotags have to be specifically imported first; only thereafter a link to a map containing the photo will be shown. The link to Upcoming.org that is provided by Flickr is visible under "Additional Information" in the right column (2).

The mashups of ActiveTags that are available for this page are visible in the large highlighted section titled "ActiveTags MergeSpace" in the lower half of the image (3). First, a map mashup shows where the photo was taken, the map itself being a fully navigable Google Maps object included as an IFrame (4). Second, there is also a link to the event at Upcoming (5).

In effect, there is not much difference between the functionalities provided by Flickr's native methods and ActiveTags' mashups on this page. However, scope and implementation are very different: Indeed, the ActiveTags mashups were created by users and they not only work on this sample page, but on arbitrary Web pages as well.

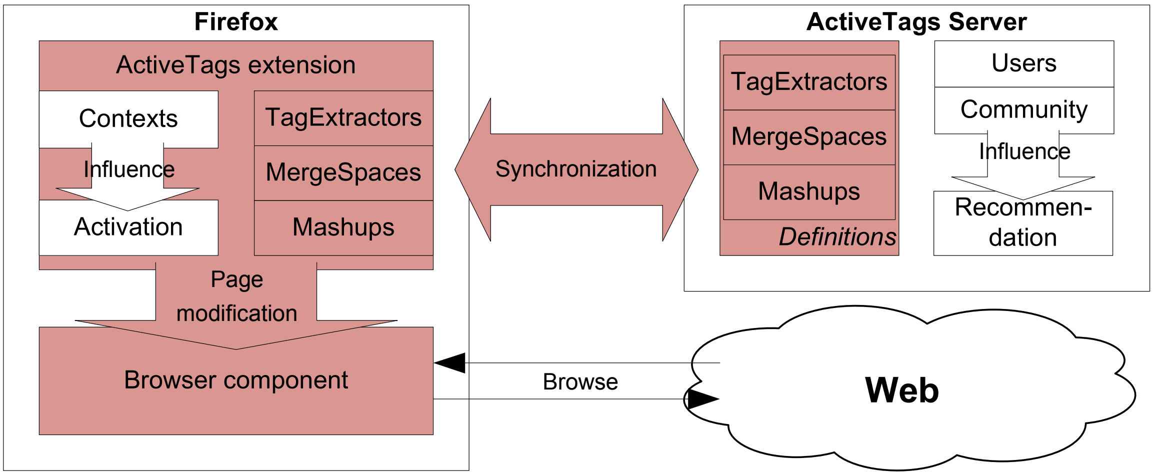
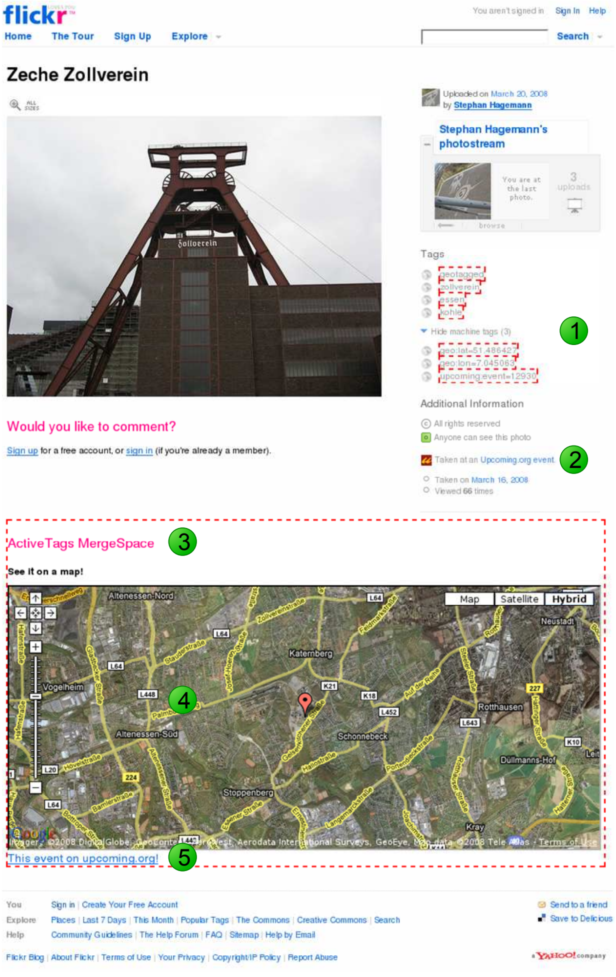


Figure 3: ActiveTags components.


flickr
You aren't signed in | Sign In | Help
Home | The Tour | Sign Up | Explore
Search
ALL SIZES
Uploaded on March 20, 2008
by Stephan Hagemann
Stephan Hagemann's photostream
You are at the last photo.
3 uploads
browse
Tags
geotagged
zollverein
essen
kohle
Hide machine tags (3)
geo:lat=51.486427
geo:lon=7.045063
upcoming:event=12930
1
Additional Information
All rights reserved
Anyone can see this photo
Taken at an Upcoming.org event
2
Taken on March 16, 2008
Viewed 66 times
3
Map | Satellite | Hybrid
Altenessen-Nord
Katernberg
Vogelheim
Altenessen-Süd
Schonnebeck
Rotthausen
Düllmanns-Hof
Stoppenberg
Kray
Neustadt
4
5
You | Sign in | Create Your Free Account
Explore | Places | Last 7 Days | This Month | Popular Tags | The Commons | Creative Commons | Search
Help | Community Guidelines | The Help Forum | FAQ | Sitemap | Help by Email
Send to a friend
Save to Delicious
Flickr Blog | About Flickr | Terms of Use | Your Privacy | Copyright/IP Policy | Report Abuse
a YAHOO! company

Zeche Zollverein

Would you like to comment?

Sign up for a free account, or sign in (if you're already a member).

ActiveTags MergeSpace

See it on a map!

This event on upcoming.org!

Figure 2: Sample page with ActiveTags mashups.

3.2.2 Design

Figure 3 contains an overview of the main components of the ActiveTags system. Depicted on the left is the user-side component of the ActiveTags system: a browser with the ActiveTags extension. The ActiveTags server (depicted on the right) stores the global database of definitions and social aspects, that is, users and community.

The *ActiveTags extension* is the hub for client interaction and it fulfills several functions. In executing the *ActiveTags routine*, it handles the visible part of ActiveTags' operations. The routine consists of the following steps:

1. Monitoring page loads,
2. extracting tags from the Web page,
3. checking for applicable mashups based on tags, URL, and settings,
4. finding a place on the Web page to merge mashups, and
5. including mashups into the Web page.

The process is aborted after steps 2, 3, or 4 if the previous step yields an empty result, resp. If, however, the process continues all the way to step 5, it leads to a situation similar to the one depicted in Figure 2, with tags detected, mashup integration space found, and mashups shown.

The extension regularly synchronizes its database of definitions with the server to support sharing among users. This encompasses TagExtractors (used in step 2), MergeSpaces (used in step 4), and mashups (used in steps 3 and 5). Finally, the extension offers users the possibility of creating new instances of these definitions. Contexts, the last portion of the extension, allow a user to specify his or her browsing context, which can be used to control the activation of different mashups in varying circumstances.

The *ActiveTags server* stores the global database of definitions. Through this database the extension instances exchange their definitions. Users can form links among each other that are equal to "friend" connections in social networks. Together with activation statistics the communities thus created influence the suggestion and ranking of definitions when they are proposed to the user.

3.2.3 Client implementation

The shaded parts of Figure 3 are those that have been implemented so far. We have opted to not implement contexts at this time, as a similar effect can be achieved through profiles in the Firefox browser. Also, the community features have not yet been released, because we first want to grow the user base and the number of definitions. Even with good ranking algorithms, a small user base that has only a few definitions to work on may lead to arbitrary and useless results, which is why we currently take care of quality control by closely monitoring all newly created definitions.

The details of our implementation of the **ActiveTags extension** will follow the steps from the five-step procedure outlined above:

Monitoring page loads. The extension registers an event listener that waits for `DOMContentLoaded` events and thus implements step 1. These events are a Mozilla-specific and they fire after the content of a page, but before images are fully loaded. It is checked whether such an event is from an IFrame that was generated by ActiveTags. If this is not the case, the process continues.

TagExtractors are definitions that perform step 2 of the ActiveTags procedure. They consist of a regular expression for URLs, an XPath expression for element extraction, and an optional separator string.

The following example is an extractor for tags attached to photos on a Facebook photo page:

URL pattern:	`^.*www\.facebook\.com\/photo\.php.*$`
XPath:	`//*[@id=’phototags’]/span`
Separator:	–

A TagExtractor’s XPath is executed on any browsed page that matches the URL pattern, where, if present, the separator string is used to split up the contents of elements into tags (otherwise the entire element is considered a single tag). The results of all applicable TagExtractors are combined and fed into the next step of the process.

There are certain extractors that are not site specific, since they make use of standardized formats, which can encode tags. Therefore they are applied to all Web pages. Relevant formats include microformats, eRDF, and RDFa. In particular, we use the rel-tag microformat (Celik & Marks 2005). It specifies tags by annotating links with a *rel* attribute whose value is “tag”. The XPath to extract tags with this microformat is `//a[contains(concat(’ ’,@rel,’ ’),’ tag ’)]`. Microformats are widely used: e.g., blogs created using major blogging tools such as those from `Blogger.com` or `Wordpress.com` encode tags for blog posts using this microformat.

The other formats are listed for completeness; they are currently not activated, since they are hardly used in practice. eRDF and RDFa allow RDF to be embedded into Web pages. GRDLL (W3C 2007) is a specification detailing how semantic data can be extracted from XML content. In the semantic data we would look for tag definitions as specified by the Newman tag ontology (Newman 2004).

If the tags on a page are not recognized by ActiveTags, users can create a new TagExtractor for it. TagExtractors are built as CSS selectors (W3C 2005) and converted into equivalent XPath expressions prior to storage and execution. Patterns are often easy to create, as tags are commonly presented on Web pages using lists or tables of links, or plain text versions of tags. Therefore, ActiveTags offers simple visual extractor creation with which users can point and click to create TagExtractors. Although this process can create only a very limited subset of selectors it is sufficient for many Web pages. Currently, only about 12% of TagExtractors in use have been manually corrected to cope with (partial) false positives.

The main part of the TagExtractor creation dialog is depicted in Figure 4: after a user opens the dialog, he or she starts creating a selector by moving the pointer over the copy of the Web page, which is displayed in the center (1). While hovering selectable elements under the pointer are highlighted. Once the user finds the tags and clicks on one of them a selector is created. Now, all elements that are covered by this selector are highlighted. The selector is also shown in the field *Selector* (2). Using the buttons that make up the selector, the selector expression can be generalized, i.e., parts can be generalized, or left out completely. This allows the user to widen the selector. Should multiple tags be captured by one HTML element, a separator can optionally be specified to separate them (3). Finally, the URL pattern on which the extractor should be working can be modified like the selector by clicking on the boxes (4). Submitting the extractor makes it available for immediate use and uploads it to the ActiveTags server so that it is available to other users as well.

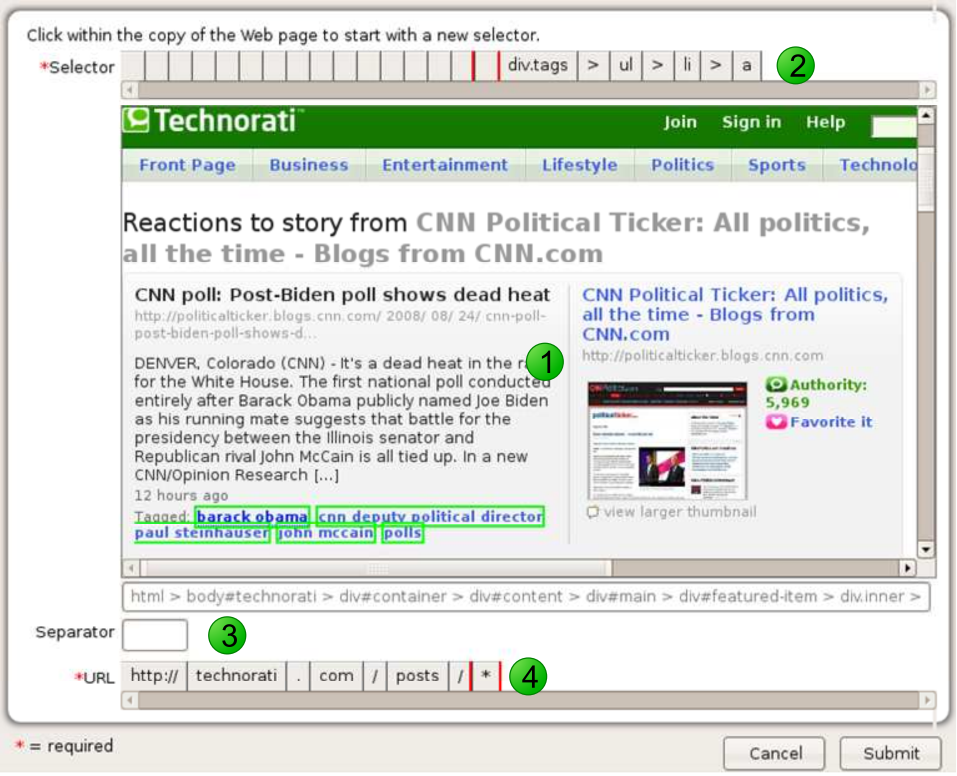
Mashup definitions check for the applicability of mashups (step 3) in two ways: via tags and via URLs. The latter is an optional list of regular expressions on URLs that, if present, needs to contain at least one item that matches the current URL. The former is a list of patterns for tags, where each item specifies what tag pattern is looked for and what part of this pattern (if any) is to be used as a mashup parameter.

The following example, which checks for the latitude component of geotags, is such an item. It defines the expected tag pattern as a regular expression. Part of the tag is to be used as a parameter. By default, it is the outermost parentheses in the regular expression that define which part of the tag is to be used as a parameter: in this case it is the numerical component of the tag. Finally, the extracted part can be referred to using the parameter name, which here is “lat.”

Tag pattern:	`^geo:lat=(-?\d*[0-9](\.\d*[0-9])?)$`
Param?	yes
Param name:	lat

The other parts of a mashup definition are relevant for its execution and will be explained below.

Finding a place on Web page. Step 4 is performed with the help of MergeSpace definitions. These are very similar to TagExtractors, the difference being that MergeSpaces do not have separators. The rule to go from selected elements to the space where to include mashups is the following: Mashups are incorporated into a page after the last element that is selected by the XPath of the definition. If multiple MergeSpaces are applicable on a page, the first MergeSpace found will be used.



Click within the copy of the Web page to start with a new selector.

*Selector: div.tags > ul > li > a

Reactions to story from CNN Political Ticker: All politics, all the time - Blogs from CNN.com

CNN poll: Post-Biden poll shows dead heat
http://politicalticker.blogs.cnn.com/ 2008/ 08/ 24/ cnn-poll-post-biden-poll-shows-d...

DENVER, Colorado (CNN) - It's a dead heat in the r... for the White House. The first national poll conducted entirely after Barack Obama publicly named Joe Biden as his running mate suggests that battle for the presidency between the Illinois senator and Republican rival John McCain is all tied up. In a new CNN/Opinion Research [...]
12 hours ago
Tagged: barack obama cnn deputy political director paul steinhauser john mccain polls

CNN Political Ticker: All politics, all the time - Blogs from CNN.com
http://politicalticker.blogs.cnn.com
Authority: 5,969

html > body#technorati > div#container > div#content > div#main > div#featured-item > div.inner >

Separator:

*URL: http:// technorati . com / posts / *

* = required

Figure 4: TagExtractor creation dialog.

Mashup execution. As seen above, mashups extract parameters from tags. In order to be able to execute, they also define where the Web procedure call (WPC, see (Vossen & Hagemann 2007)) endpoint implementing the mashup is located. For the example introduced above the URL is `www.bananathinking.net/ActiveTags/webServices/google-map/map.php?lat=%lat%&lon=%lon%`. At the end of this URL, enclosed in percent signs, the parameter names for "lat" and analogously for "lon" can be seen. For an execution of this WPC, these parameter names are replaced with the current values. Mashups are executed by a piece of activating code that is embedded into the Web page.

There are currently two types of mashups, which differ in their embed code: link and IFrame mashups. The former creates a link pointing to the URL discussed above, while the latter creates an IFrame for which the source is that URL.

3.2.4 Server implementation

The **ActiveTags server** has been implemented using CakePHP (`cakephp.org`). It currently resides at `bananathinking.net/ActiveTags/server`. It offers a management console for enabling, disabling, and modifying definitions. While access to most parts of the server is restricted, definitions can be up- and downloaded by ActiveTags extensions. A running ActiveTags extension will usually download new definitions from the server once after every browser start. There are options to change this default behavior and users can trigger this manually. The server delivers definitions as a JSON string, which is cached by the extension so that it is operable even should the server be unreachable. A newly created definition is uploaded to the server and will spread to other users when they restart their browsers or manually update their definitions.

The current implementation is running proof that the outlined system is functional; however, there are aspects of large, distributed systems which need to be addressed. The server implementation is performing all storage and distribution functions of the system. This centralized component will eventually become a bottleneck when the number of users grows. Therefore, the next step in the development will include an extension of the server towards a peer-to-peer architecture. We are currently investigating the use of the BitTorrent protocol. In this scenario, the server will continue to handle definition uploads, but distribute them not only directly, but also via the BitTorrent protocol. To this end, the server will provide tracker information for the definition files and be the seed of these files, while the Firefox extensions will act as BitTorrent peers, helping with the distribution of definitions. An approach of combined short-term individual updates and regular cumulative ones will balance timeliness and generated network traffic.

While TagExtractors and MergeSpaces are needed by every extension, the utility of mashups is very much dependent on what a user wants the functionality to be. Therefore, a second way to distribute mashups will be implemented in the future. Similar to the way Ubiquity distributes actions (see Section 4), it will be possible to include mashup definitions into arbitrary Web pages, highlighted in such a way that ActiveTags extensions recognize them and allow installation. This approach would simultaneously tackle conflicts between different requirements towards mashups and scalability of mashup distribution. At the same time it allows mashups to remain private, e.g., a business mashup using company internal sources.

Conflicts between definitions are also an issue with TagExtractors and MergeSpaces. They are tackled by a fine-granular recommendation system. Based on the numbers of users activating certain definitions, and explicitly stated inconsistencies, activations will be proposed to other users.

4 Related Work

The projects closest to ActiveTags are Piggy Bank (Huynh et al. 2007) and Intel Mash Maker (`www.mashmaker.intel.com/`), which both are also Firefox extensions and quite similar in their features. Piggy Bank allows Web pages to be connected after the pages have been scraped. Scrapers are available for nine Web sites and three generic formats. The scraped data is transformed into Semantic Web languages, and then used for mashups. Intel's project covers features similar to Piggy Bank, yet without focusing on Semantic Web languages. The scraped data has to be transferred to Intel servers before it can be used. Both projects do not have an integrated notion of communities. In order for mashups to be used they have to be manually installed. This gives users control over which mashups they want to use. We aim at retaining control while increasing the ease of sharing mashups. An essential part of ActiveTags, namely supporting the development of meaning in folksonomies, is not within the scope of these two projects.

Ubiquity (Raskin 2008), an extension adding a command line interface to Firefox, has a complementary approach. Instead of focusing on the resources (of Web sites), its main concept are *commands*, which can be invoked on any Web page. If text is selected on a Web page this text functions as the parameter to commands automatically. The mapping example mentioned above is available in Ubiquity. Here, the workflow is as follows: The text of an address is selected, the Ubiquity shortcut is pressed (which opens the "command line"), the command "map" is typed in, which brings up a small map immediately. New commands can be developed by anyone. The code implementing a command is embedded into a Web page using special markup so that a browser with Ubiquity installed will inform the user of the command's presence and allow installation.

The online mail application Zimbra implements a feature called "Zimlets" (Zimbra 2006). These are programs that are activated on pre-specified patterns found in certain objects (mails, event entries, contacts, etc.). They are used to show data from other sources, thereby effectively mashing-up these pieces of content. The Operator extension (`https://addons.mozilla.org/de/firefox/addon/4106`) for the Firefox browser scans Web pages for microformat data, which it can extract and send to other programs that make use of this data. The extension Sxipper (`www.sxipper.com`) extends the Firefox password store and is able to automatically fill out forms by learning forms on Web pages through user interaction. The information on how a form is to be used is stored centrally, so of all Sxipper users using a site only the first one has to train a particular form. The ActiveTags system uses similar information extraction mechanisms as these two extensions. It employs a mashup execution mechanism similar to Zimlets, but allows more complex invocation criteria.

There is a lot of research on folksonomies and tagging, as well as on bringing these two together with Semantic Web concepts; see Voss (2007) for an overview. Ontologies are a central aspect of the Semantic Web. They provide machine-interpretable semantics and intend to allow machines to reason about the meaning of objects. In order to create an ontology, a design process is needed, which means upfront investment of knowledge engineers that will structure the respective field (Hüsemann & Vossen 2005).

There are several different attempts to bridge social tagging systems and the resulting folksonomies with ontologies. It has been argued by Christiaens (2006) that these two approaches are complementary and differ by their respective degree of freedom. The relationship between folksonomies and machine-extracted keywords was evaluated in Al-Khalifa & Davis (2007) in order to study the former's potential for being used to generate semantic metadata. How semantic knowledge can be used to enhance tags in folksonomies explicitly has been evaluated by Angeletou et al. (2007). Tags have been processed by Specia & Motta (2007) to infer meaning by using other Web resources. Their results are "faceted ontologies ... partial ontologies conceptualizing specific facets of knowledge." In contrast, statistical methods have been used by Zhang et al. (2006) to infer semantics from folksonomies.

Plug-in modules have been proposed by Wu et al. (2006) for social tagging systems that support the extraction of knowledge from those systems. They thus aim at systematizing the capabilities of the technical infrastructure to allow for further use in this direction. The possibility of defining relationships between tags into a social bookmarking system has been introduced by Hotho et al. (2006). All these contributions relate folksonomies and the Semantic Web in one way or another, but do not attempt to directly support the use of tags programmatically, which is the aim of our contribution.

5 Conclusions and Future Work

Social tagging systems obtain their capabilities from massive amounts of users who participate. The open structure and low specificity of tagging allows for many uses, without forcing a particular one or favoring another. The downside of this is that, although the tags allow it, only a few forms of use are efficiently supported in typical tagging systems. The ActiveTags system brings direct programmatic support for tags in the form of mashups right alongside the objects where the tagging takes place.

This paper has shown how the ActiveTags system has been designed and how it works. The system is able to copy popular tag-based mashups that are in use today. Moreover, it has been shown that right from the start the scope is greatly improved when compared to these examples. The evaluation period that is currently running has seen the creation of several new mashups. Other initial results from our evaluation are also promising: For example, Yahoo! Pipes has been used to connect related articles of big German news portals. The rel-tag microformat TagExtractor detects tags on many pages that have not been specifically trained for ActiveTags. Still, the largest growth in numbers has been observed for TagExtractors – most of which worked properly right away. As already mentioned, we are working on a detailed evaluation of the ActiveTags system. Nevertheless, we take the observations mentioned above as positive indicators from the first weeks of our evaluation. Although not all features of the system have been activated, the desired developments are already observable. While continuing the evaluation, we will complete the implementation of the system as outlined in Section 3.2.4.

The design of the ActiveTags system is inherently distributed: The more Web sites use tags (and these becoming detectable), the more common tag interpretations and the more ubiquitous mashups can become. ActiveTags leverages several dimensions of user-generated content and we expect it to influence the way in which tags are used.

References

Al-Khalifa, H. S. & Davis, H. C. (2007), 'Exploring The Value Of Folksonomies For Creating Semantic Metadata.', *International Journal on Semantic Web and Information Systems (IJSWIS)* **3**(1), 13–39.

Andrews, P. (2006), 'Flickr Upcoming Event Userscript'. http://userscripts.org/scripts/show/5464.

Angeletou, S., Sabou, M., Specia, L. & Motta, E. (2007), Bridging the Gap Between Folksonomies and the Semantic Web: An Experience Report., *in* 'Workshop: Bridging the Gap between Semantic Web and Web 2.0, European Semantic Web Conference'.

Celik, T. & Marks, K. (2005), 'Rel-tag microformat'. http://microformats.org/wiki/rel-tag.

Christiaens, S. (2006), Metadata Mechanisms: From Ontology to Folksonomy ... and Back, *in* R. Meersman, Z. Tari & P. Herrero, eds, 'OTM Workshops (1)', Vol. 4277 of *Lecture Notes in Computer Science*, Springer, pp. 199–207.

Golder, S. & Huberman, B. A. (2006), 'Usage patterns of collaborative tagging systems', *Journal of Information Science* **32**(2), 198–208.

Hammond, T., Hannay, T., Lund, B. & Scott, J. (2005), 'Social Bookmarking Tools (I)', *D-Lib Magazine* **11**(4). http://www.dlib.org//dlib/april05/hammond/04hammond.html.

Hinchcliffe, D. (2006), 'Continuing an Industry Discussion: The Co-Evolution of SOA and Web 2.0', Dion Hinchcliffe's Web 2.0 Blog. http://web2.wsj2.com/continuing_an_industry_discussion_the_coevolution_of_soa_and.htm.

Hotho, A., Jäschke, R., Schmitz, C. & Stumme, G. (2006), Emergent Semantics in BibSonomy, *in* C. Hochberger & R. Liskowsky, eds, 'GI Jahrestagung (2)', Vol. 94 of *LNI*, GI, pp. 305–312.

Hüsemann, B. & Vossen, G. (2005), Ontology Engineering from a Database Perspective, *in* S. Grumbach, L. Sui & V. Vianu, eds, 'ASIAN', Vol. 3818 of *Lecture Notes in Computer Science*, Springer, pp. 49–63.

Huynh, D., Mazzocchi, S. & Karger, D. (2007), 'Piggy bank: Experience the semantic web inside your web browser', *Web Semantics: Science, Services and Agents on the World Wide Web* **5**(1), 16–27. http://www.sciencedirect.com/science/article/B758F-4MVF4YB-1/2/38957cbb8d2a861463d6a77e35637bf6.

Lund, B., Hammond, T., Flack, M. & Hannay, T. (2005), 'Social Bookmarking Tools (II): A Case Study - Connotea', *D-Lib Magazine* **11**(4). http://www.dlib.org/dlib/april05/lund/04lund.html.

Marlow, C., Naaman, M., boyd, d. & Davis, M. (2006), HT06, tagging paper, taxonomy, Flickr, academic article, to read, *in* (Wiil et al. 2006), pp. 31–40.

Newman, R. (2004), 'Newman Tag Ontology'. http://www.holygoat.co.uk/projects/tags/.

Raskin, A. (2008), 'Introducing ubiquity', Mozilla Labs Blog. http://labs.mozilla.com/2008/08/introducing-ubiquity/.

Specia, L. & Motta, E. (2007), Integrating Folksonomies with the Semantic Web, *in* E. Franconi, M. Kifer & W. May, eds, 'ESWC', Vol. 4519 of *Lecture Notes in Computer Science*, Springer, pp. 624–639.

Straup Cope, A. (2007), 'Machine tags', Flickr API / Discuss. http://www.flickr.com/groups/api/discuss/72157594497877875/.

Voss, J. (2007), 'Tagging, Folksonomy & Co - Renaissance of Manual Indexing?'. http://arxiv.org/abs/cs/0701072.

Vossen, G. & Hagemann, S. (2007), *Unleashing Web 2.0 - From Concepts to Creativity*, Morgan Kaufmann, Burlington, MA.

W3C (2005), 'Selectors, W3C Working Draft'. http://www.w3.org/TR/css3-selectors/.

W3C (2007), 'W3C GRDLL Specification'. http://www.w3.org/2001/sw/grddl-wg/.

Wiil, U. K., Nürnberg, P. J. & Rubart, J., eds (2006), *HYPERTEXT 2006, Proceedings of the 17th ACM Conference on Hypertext and Hypermedia, August 22-25, 2006, Odense, Denmark*, ACM.

Wu, H., Zubair, M. & Maly, K. (2006), Harvesting social knowledge from folksonomies, *in* (Wiil et al. 2006), pp. 111–114.

Zhang, L., Wu, X. & Yu, Y. (2006), 'Emergent Semantics from Folksonomies: A Quantitative Study', *J. Data Semantics* **VI**, 168–186.

Zimbra (2006), *Zimlets - A Mechanism for Integrating Disparate Information Systems and Content with the Zimbra Collaboration Suite (ZCS)*, version 0.97 edn.

Unified Q-ary Tree for RFID Tag Anti-Collision Resolution

Prapassara Pupunwiwat **Bela Stantic**

Institute of Integrated and Intelligent Systems
Griffith University, Gold Coast Campus
Queensland 4222, Australia
Email: {p.pupunwiwat, b.stantic}@griffith.edu.au

Abstract

Radio Frequency Identification (RFID) technology uses radio-frequency waves to automatically identify people or objects. A large volume of data, resulting from the fast capturing RFID readers and a huge number of tags, poses challenges for data management. This is particularly the case when a reader simultaneously reads multiple tags and Radio Frequency (RF) collisions occur, causing RF signals to interfere with each other and therefore preventing the reader from identifying all tags. This problem is known as *Missed reads*, which can be solved by using *anti-collision* techniques to prevent two or more tags from responding to a reader at the same time. The current *probabilistic* anti-collision methods are suffering from *Tag starvation problems* so not all tags can be identified, while the *deterministic* methods suffer from too long *Identification delay.* In this paper, a "Unified Q-ary Tree Protocols" based on *Query tree* is presented. In empirical study compared with the Query tree and *4-ary tree*, we show that the proposed method performs better, it requires less number of queries per complete identification, which results in less total identification time.

1 Introduction

RFID technology has gained significant momentum in the past few years. It has promised to improve the efficiency of business processes by providing the automatic identification and data capture. The core RFID technology is not new, and it can be traced back to World War II where it was used to distinguish between friendly and enemy aircrafts or known as friend-or-foe (Landt 2001). Currently RFID technology is used in different systems such as: transportation, distribution, retail and consumer packaging, security and access control, monitoring and sensing, library system, defence and military, health care, and baggage and passenger tracing at the airports.

In warehouse distribution environment where RFID systems are deployed, the Ultra High Frequency (UHF) range of radio-frequency waves are used for long distance identification. UHF includes frequencies from 300 to 1000MHz, but only two frequency ranges, 433MHz and 860-960MHz, are used for UHF RFID systems. The 433MHz frequency is used for *active tags*, while the 860-960MHz range is used for *passive tags* or *semi-passive tags.* All protocols in the UHF range have some type of anti-collision capability, which allow multiple tags to be read simultaneously within the *interrogator zone* (Brown et al. 2007).

Despite significant improvement with respect to the quality of readers and tags, a significant percentage of captured data still has errors, which are particularly due to the *Missed reads.* To prevent these Missed reads, mostly caused by RF Collision, several techniques for the *Edge anti-collision* have been proposed in the literature. However, these approaches still suffer from either *Tag starvation problem*, or produced too many *Collision cycles* and *Idle cycles*, which causes *Identification delay.*

In this study, we propose a new anti-collision algorithm called "Unified Q-ary Tree Protocols", which is a combination of *Q-ary trees*, particularly a binary tree, 4-ary tree, 8-ary tree, and 16-ary tree, to optimise the anti-collision in reading RFID tags. We focus on *deterministic* anti-collision protocols since they can achieve 100 percent identification. We also concentrate on the impact of *similarity of EPC* data, especially in warehouse environment where most items have *bulky movement.* These items are usually manufactured from the same company which evidently used the same *Encoding Schemes* and have the same *Company Prefixes.* In simulated experimental study, we show that our method reduces *Collision cycles* and *Idle cycles*, which resulted in less *Identification delay* and improve a quality of captured data.

The remainder of this paper is organised as follows: In section 2, some general background on RFID and information related to *Missed reads* including their causes is provided. In section 3, we discuss the related works to our proposed method and their limitations, which include *probabilistic* and *deterministic* anti-collision methods, and *Query tree* protocols. In section 4, we are presenting a new technique, the "Unified Q-ary Tree Protocols" including methodology and scenarios. In section 5, we present experimental results, analysis and discussions, and finally in section 6 we provide our conclusion and future work.

2 RFID Background

RFID may only consist of a tag and a reader but a complete RFID system involves many other components, such as computer, network, Internet, and software such as middleware and user applications. A typical RFID system is divided into two layers: the physical layer and Information Technology (IT) layer (Brown et al. 2007).

The physical layer consists of: one or more reader antennas, one or more readers (Interrogator), one or more tags (Transponder), and deployment environment.

The IT layer consists of one or more host computers connected to readers (directly or through a network), and appropriate software such as device drivers, filters, middleware, databases, and user applications.

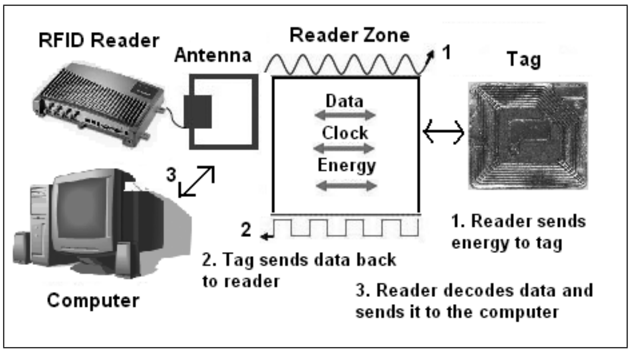

RFID Reader
Antenna
Reader Zone
1
Tag
Data
Clock
Energy
3
2
1. Reader sends energy to tag
2. Tag sends data back to reader
3. Reader decodes data and sends it to the computer
Computer

Figure 1: An example of how RFID tag, reader, middleware and application operate.

Figure 1 shows that RFID reader retrieves information from tag and sends that information back to host computer via middleware. Middleware first needs to convert *raw data* retrieved by the reader to a meaningful data before sending them to an application layer assigned on a host computer.

2.1 Electronic Product Code (EPC)

EPC Class 1 Generation 2 is widely used in UHF range for communications at 860-960MHz. This passive tag is also referred to as EPC Gen-2 tag, where the standards have been created by EPCGlobal (EPCGlobal 2006). The most common encoding scheme currently widely used includes: General Identifier (GID-96), Serialised Global Trade Item Number (SGTIN-96), Serialised Shipping Container Code (SSCC-96), Serialised Global Location Number (SGLN-96), Global Returnable Asset Identifier (GRAI-96), Global Individual Asset Identifier (GIAI-96), and DoD Identifier (DoD-96).

In this paper, we only focus on **General Identifier 96-bits** type due to page limitation. The implementation and experiment will be determined by the impact of EPC *Encoding Scheme* and *bulky movement* of items, therefore at present, only one type of encoding is necessary.

GID-96	Bit	Max.Decimal/Binary
Header	8	0011 0101
GMN*	28	268,435,455
Object Class	24	16,777,215
Serial Number	36	68,719,476,735

Table 1: The General Identifier (GID-96) includes three fields in addition to the 'Header' with a total of 96-bits binary value (*GMN = 'General Manager Number').

The general structure of EPC tag encodings is a string of bits, consisting of a fixed length (8-bits) 'Header' followed by a series of numeric fields whose overall length, structure, and function are completely determined by the Header value. Table 1 shows an example of GID-96 EPC generation 2 encoding scheme. Only Header is shown in binary, the rest are shown in decimal number.

2.2 Warehouse Distribution Justification

Items tend to move and stay together through different locations especially in a large warehouse (Gonzalez, Han & Li 2006), (Gonzalez, Han, Li & Klabjan 2006). For example, 10 pallets with 60 cases of crystal glasses each may be ready to leave the warehouse and deploy to different retailer. At this point, 10 pallets move along a conveyor belt through dock doors mounted with RFID readers. We can, for example, use the assumption that many RFID objects stay or move together, especially at the early stage of distribution, and these EPC data will be very similar since the first few bits of encoding will determine the type of *Encoding Scheme* (Header), *Company Prefixes* (GMN), and *Object Class*.

2.3 Errors in RFID Data Streams

Due to the characteristics of RFID streaming data, which does not contain much information and can be captured very fast, some of these data need to be filtered before being stored into the database. Such filtering is called 'Edge cleaning/filtering'. There are four typical errors: *Unreliable reads*, *Noise*, *Missed reads*, and *Duplication*. Several techniques for filtering RFID data have been proposed in literatures (Bai et al. 2006), (Jeffery, Garofalakis & Franklin 2006), (Carbunar et al. 2005), (Fishkin et al. 2004); however, these techniques only filter specific kind of errors generated. A research on Noise and Duplication data filtering has been done very well previously; however, the Unreliable reads can be prevented only at some point. This depends on the deployment of readers, tags, and an environment.

2.4 Problems with "Missed reads" and their solutions

Missed Reads are very common in RFID applications and often happened in a situation of low-cost and low-power hardware, which leads to a frequently *Dropped Reading* referred to in other work (Derakhshan et al. 2007). Another cause of Missed reads is usually when multiple tags are to be detected by a reader but RF collisions occur causing RF signals to interfere with each other preventing the reader from identifying any tags. Dropped reading can be easily filtered using "Smoothing" technique proposed by Jeffery, Alonso, Franklin, Hong & Widom (2006), where missing data from specific time can be filled. However, preventing data resulting from RF collisions can be harder and in order to solve this problem, anti-collision can be performed at the edge to prevent two or more tags from responding to a reader at the same time.

3 Related Works

The various types of *anti-collision* methods for multi-access/tag collision can be reduced to two basic types: *probabilistic* method and *deterministic* method.

3.1 Probabilistic Methods

In a *probabilistic* method, tags respond at randomly generated times. If a collision occurs, colliding tags will have to identify themselves again after waiting at a random period of time. This technique is faster than *deterministic* but suffers from *Tag starvation problem* where not all tags can be identified due to the random nature of chosen time.

The *probabilistic* methods are based on "Slotted-ALOHA" protocol (Quan et al. 2006), which introduces discrete time-slots for tags to be identified by reader at the specific time. To improve the performance, a "Frame Slotted ALOHA" (Shin et al. 2007) based anti-collision algorithm has been suggested, where each frame is formed of specific number of slots that is used for the communication between the readers and the tags. Each tag in the interrogation zone arbitrarily selects a slot for transmitting the tag's information. However, the probabilistic method can only be improved to a very high throughput rate but they still cannot achieve 100 percent tag identification.

3.2 Deterministic Methods

The *deterministic* method starts by asking for the first number of the tag (Query Tree algorithm) until it matches the tags; then it continues to ask for additional characters until all tags within the region are found. This method is slow and introduces a long *Identification Delay* but leads to fewer collisions, and have 100 percent successful identification rate.

Such *deterministic* methods can be classified into a *Memory* based algorithm and a *Memoryless* based algorithm. In the Memory based algorithm, which can be grouped into a splitting tree algorithm such as an "Adaptive Splitting Algorithm" and a "Bit Arbitration Algorithm", the reader's inquiries and the responses of the tags are stored and managed in the tag memory, resulting in an equipment cost increase especially for RFID tags. In contrast, in the Memoryless based algorithm, the responses of the tags are not determined by the reader's previous inquiries. The tags' responses and the reader's present inquiries are determined only by the present reader's inquiries so that the cost for the tags can be minimised. Memoryless based algorithms include a "Query Tree Algorithm", a "Collision Tracking Tree Algorithm", and a "Tree Walking Algorithm".

In this paper, we will focus on *Memoryless Query Tree* based protocols since it is the most popular and is an effective *anti-collision* technique for passive UHF tags. However, there are other improved anti-collision methods based on Query Tree such as an "Adaptive Query Splitting" (AQS) proposed by Myung & Lee (2006*b*), Myung & Lee (2006*a*) and a "Hybrid Query Tree" (HQT) proposed by Ryu et al. (2007). AQS keeps information which is acquired during the last identification process in order to shorten the collision period. This technique requires tags to support both the transmission and reception at the same time, thereby making it difficult to apply to low-cost passive RFID systems. HQT uses a 4-ary query tree instead of a binary query tree, which increased too many *Idle cycles* despite reducing *Collision cycles*, while extra memory needed also increases as an identification process gets longer, since each query increase the prefixes by 2-bits instead of 1-bit. Accordingly, the *Query Tree algorithm*, adopted at present as the *anti-collision* protocol in EPC Class 1, may be limited to the tree based anti-collision protocol, which can be implemented (Choi et al. 2008).

3.3 Query Tree Based Protocols

The *Query tree* is a data structure for representing prefixes which is sent by the reader in the Query tree protocols. A reader identifies tags through an uninterrupted communication with tags. The Query tree protocols consist of loops, and in each loop, the reader transmits a query with specific prefixes, and the tags respond with their IDs. Only tags with IDs that match the prefixes, respond. When only one tag responds to reader, the reader successfully recognises the tag. When more than one tag tries to respond to reader's query, *tag collision* occurs and the reader cannot get any information about the tags. The reader, however, can recognise the existence of tags to have ID which match the query. For identifying tags that lead to the collision, the reader tries to query with 1-bit longer prefixes in next loops. By extending the prefixes, the reader can recognise all the tags.

Depending on the number of tags that respond to the interrogator, there are three cycles of communication between tag and reader.

- **Collision cycle:** Number of tags that respond to the reader is more than one. The reader cannot identify the ID of tags.
- **Idle cycle:** No response from any tag. It is a waste that should be reduced.
- **Success cycle:** Exactly one tag responds to the reader. The reader can identify the ID of the tag.

The delay of identification of tags is mostly affected by the *Collision cycles*, *Idle cycles*, and *similarity of IDs*. Therefore, reducing the number of Collision cycles and eliminating Idle cycles, can improve the identification ability of the reader.

In order to overcome shortcomings of existing methods for collision resolution, we propose a "Unified Q-ary Tree Protocols" based on *Memoryless QT*. We focus on analysing the impact of EPC Gen-2 encoding scheme and the fact of *bulky items movement* within warehouse, therefore, we only considered static tags where tags have no mobility. We investigated different combination of *Q-ary trees*, to reduce *Collision cycles* and *Idle cycles*, which lead to shorter identification time.

4 Unified Q-ary Tree Methodology

In order to reduce *Collision cycles* and *Idle cycles*, and minimise total *Identification delay*, a "Unified Q-ary Tree Protocols" or a combination of two *Q-ary trees* are employed. The Unified Q-ary tree is a *Memoryless anti-collision protocols* based on QT. This section will describe the Q-ary tree, the scenarios where EPC data are similar, and what can be improved by combining two Q-ary trees together.

4.1 Q-ary Tree

Instead of using a *Query Tree*, which uses each bit of tag ID to split a tag set, *Q-ary tree* uses every 2-, 3-, or 4-bits of tag ID to split a tag set. Q-ary tree increases the child node of tree from '2' to '4', '8' or '16' nodes and so on. This way, we can reduce more collision but at the same time, *Idle cycles* will also increase. In the literature (Ryu et al. 2007), the author used a 4-ary tree (HQT) to optimise the anti-collision performance, which increases a lot of Idle cycles despite reducing number of *Collision cycles*, and requires extra memory and time to avoid them. Therefore, the best way to solve the problem is to produce both Collision cycles and Idle cycle as low as possible in order to improve identification time.

4.2 Warehouse Distribution Scenarios

In this paper, we are examining a specific scenario based on the assumption that items tend to move and stay together through different locations especially in a large warehouse. We are focusing on Crystal warehouse scenario which can be classified into four different scenarios as follows:

Scenario One: Two collided tags are captured and they have the same *Encoding Scheme* (Header), same *General Manager Number* (Company Prefixes), same *Object Class*, and different *Serial Number*. We can assume that all items are from the same warehouse that uses the same Encoding Scheme throughout the warehouse, and the warehouse also keeps different kind of product from different companies.

For example, the company warehouse produces different kind of crystal wine glasses, and all glasses that have the same sculpture will be packed in the same case and pallet. Therefore, crystal Red-wine glasses and crystal White-wine glasses should be packed in different case and pallet since they are different type of wine glasses. Within this scenario, each case of wine glasses will have a unique *Serial Number* attached to it with different *Object Class* for each pallet of White-wine or Red-wine.

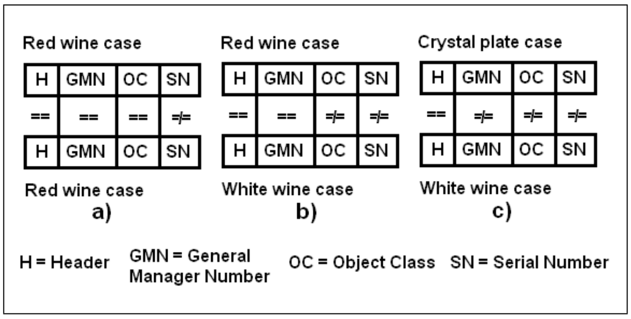


Figure 2: Crystal Warehouse Scenario: a) Two cases of Red-wine have the same Object Class but different Serial Number, b) White-wine case and Red-wine case have different Object Class and Serial Number, and c) White-wine case and Crystal plate case have different General Manager Number, Object Class, and Serial Number.

As for scenario one, by using the crystal warehouse example from Figure 2 a), when two collided tags are captured and they have the same *Encoding Scheme*, same *General Manager Number*, same *Object Class*, and unique *Serial Number*; we believe that both tags are each attached to two different cases of Red-wine.

Scenario Two: Two collided tags are captured and they have the same *Encoding Scheme*, same *General Manager Number*, different *Object Class*, and different *Serial Number*.

As for scenario two, by using the crystal warehouse example from Figure 2 b), when two collided tags are captured and they have the same *Encoding Scheme*, same *General Manager Number*, unique *Object Class*, and unique *Serial Number*; we believe that one tag is attached to Red-wine case, while the other tag is attached to White-wine case.

Scenario Three: Two collided tags are captured and they have the same *Encoding Scheme*, different *General Manager Number*, different *Object Class*, and different *Serial Number*.

As for scenario three, by using the crystal warehouse example from Figure 2 c), when two collided tags are captured and they have the same *Encoding Scheme*, unique *General Manager Number*, unique *Object Class*, and unique *Serial Number*; we believe that one tag is attached to Crystal plate case, while the other tag is attached to White-wine case. We can also make the assumption that there are two different companies producing separate crystal ware; and the wine glasses and plates are from different company but share the same warehouse since they are both crystal.

Scenario Four: Two collided tags are captured and they have the different *Encoding Scheme*, different *General Manager Number*, different *Object Class*, and different *Serial Number*. We can assume that all items are from different company that uses different encoding schemes. For example, two wine glasses with different sculpture, one made from crystal and one made from plastic, are allocated in the same warehouse. This scenario will not be discussed any further in this paper since we are only looking at a large warehouse distribution where most items move together as a group. Therefore, most items from the same type of manufacturing will stick together until deployed to smaller retailer.

4.3 Unified Q-ary Tree

Instead of using a plain *Q-ary tree*, which uses every 2-, 3-, or 4-bits of tag ID to split a tag set, we propose a "Unified Q-ary tree" or a combination of two Q-ary trees (12 combinations), which can reduce more collision and at the same time, *Idle cycles* can be minimised. For example, we can combine 4-ary tree with 8-ary tree and apply this *anti-collision* to 96-bits EPC; however, we need to configure the right partition so that 4-ary tree can be applied to the first half bits of EPC and 8-ary tree can be applied to the remaining bits. The remaining of this section will focus on two approaches: 1) a *Naive* approach, where Q-ary tree is non-unified and only a single Q-ary tree is used as an anti-collision; and 2) a *Unified* approach, where two Q-ary trees are combined as an anti-collision with 12 possible combinations.

Naive Approach - Non-Unified Q-ary tree: The Naive approach is a *non-unified Q-ary tree* that does not have a combination between two different Q-ary trees. There are four non-unified Q-ary trees investigated in this paper: binary QT, 4-ary tree, 8-ary tree, and 16-ary tree. Table 2 represents a *Number of bits* needed for each query using different Q-ary tree.

	Binary	**4-ary**	**8-ary**	**16-ary**
No. of bits	1	2	3	4

Table 2: The non-unified Q-ary Tree is where no combination between two Q-ary trees is necessary. The Table represents 4 non-unified Q-ary tree: binary, 4-ary, 8-ary, and 16-ary tree with 1, 2, 3, and 4 bits needed for each query respectively.

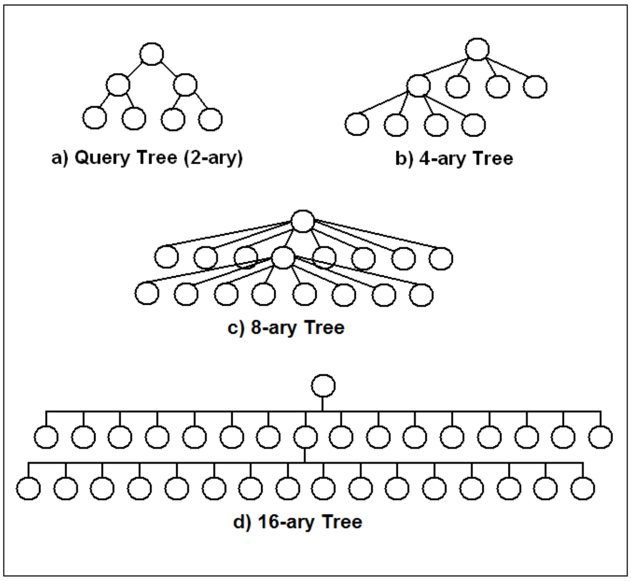
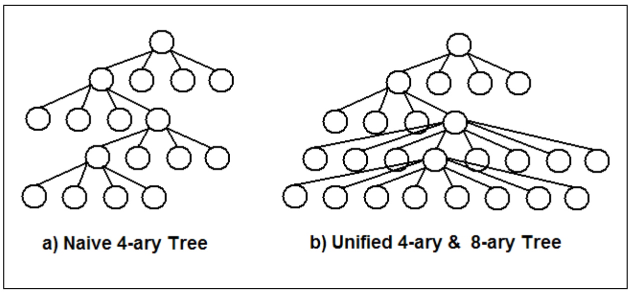


Figure 3: Naive Q-ary Tree: a) Query Tree (2-ary), b) 4-ary Tree, c) 8-ary Tree, and d) 16-ary Tree.

Figure 3 shows the example of the *Naive Q-ary tree*. There are four Naive Q-ary trees shown in the figure: a) Query/Binary/2-ary Tree with 2 child nodes, b) 4-ary Tree with 4 child nodes, c) 8-ary Tree with 8 child nodes, and d) 16-ary Tree with 16 child nodes.

The *Highest Level Tree* for each *Naive Q-ary tree* is calculated as shown in Table 4. The first four rows represent the Highest Level Tree for a binary QT, 4-ary tree, 8-ary tree, and 16-ary tree. Calculation for this Highest Level Tree will be explained in detail under heading 'Highest Level Tree for each combination'.

Unified Approach - Unified Q-ary tree: The Unified approach is a *unified Q-ary tree* with 12 possible combinations. This approach will be applied on each collided tags EPC which will be split using every 1, 2, 3, or 4-bits of tag ID for the first few queries; and then at one point every 1, 2, 3, or 4-bits will be queried. With the fact that most items from warehouse have *bulky movement*, first few bits of EPC will be identical. For example, first 8-bits of EPC are 'Header', which will be the same for all items using the same encoding and they usually came from the same company and in the same pallet. These 8-bits of EPC can be bypassed faster using 4-ary tree instead of binary tree but by doing so, too many *Idle cycles* will be produced. By using 4-ary tree instead of binary tree, the *Number of bits* needed for each query also accumulates faster. Thus, we need to optimise the performance of "Unified Q-ary tree" by configuring the right separating point between the two Q-ary trees. The objective of Unified Q-ary tree is to minimise the Number of Bits used for querying all tags within an interrogation zone. Figure 4 shows the example of the Naive 4-ary Tree (4a) and the Unified 4-ary & 8-ary Tree (4b).


a) Naive 4-ary Tree

b) Unified 4-ary & 8-ary Tree

Figure 4: a) Naive 4-ary Tree, and b) Unified 4-ary & 8-ary Tree.

	binary		4-ary		8-ary		16-ary	
	F	S	F	S	F	S	F	S
binary	-		2	1	3	1	4	1
4-ary	1	2	-		3	2	4	2
8-ary	1	3	2	3	-		4	3
16-ary	1	4	2	4	3	4	-	

Table 3: The Unified Q-ary Tree can be merged into 12 different combinations. Each EPC can be divided into two parts: First half (F) of EPC where all bits are identical, and Second half (S) of EPC where most bits are unique for all EPC in the reader zone. 1, 2, 3, and 4 represent the Number of bits queries each time for splitting tags when collision occurred.

For *Highest Level Tree*, *Idle cycles*, *Collision cycles*, and *Number of bits* estimation calculation purposes; let 'F' be the first half of EPC where bits are identical, and let 'S' be the second half of EPC where bits are unique. Table 3 shows possible combinations between four of the Q-ary trees; binary, 4-ary, 8-ary, and 16-ary.

Highest Level Tree for each combination: We now present a *Highest Level Tree* for all combinations of both *Naive Q-ary tree* and *Unified Q-ary tree*. Table 4 shows the Highest Level Tree where x_l is the maximum level tree of 'x' (first partition variable); y_l is a maximum level tree of 'y' (second partition variable); and T_l is a maximum level tree of $x_l + y_l$.

From the Table, for the first four *Naive Q-ary tree*, we can see that T_l is equal to 96 divided by the number of bit/bits needed for each query. For example: for the Naive 4-ary tree, $T_l = 48$ which is 96 divided by 2. In the case of a Naive Q-ary tree, 'F', 'S', and variable 'x' and 'y', does not play any major role.

Sample calculation of 4-ary tree for x_l, y_l, and T_l where x = 36, y = 60:

$$\log_F(2^x) + \log_S(2^y) = \log_4(2^{36}) + \log_4(2^{60})$$

$$\log_4(2^{36}) + \log_4(2^{60}) = \frac{\log_{10}(2^{36})}{\log_{10}(4)} + \frac{\log_{10}(2^{60})}{\log_{10}(4)}$$

$$\log_4(2^{36}) + \log_4(2^{60}) = 4 + 44 = 48$$

OR

$$\log_4(2^{96}) = \frac{\log_{10}(2^{96})}{\log_{10}(4)} = 48$$

Therefore, $x_l = 4$, $y_l = 44$, and $T_l = 48$

For the remaining 12 combinations of *Unified Q-ary tree* in Table 4, we can see that T_l is equal to the sum of x_l and y_l, where the sum of x and y equal to 96. For example: for 4-ary tree combining with 8-ary tree (F = 4, S = 8), $T_l = 34$ which is 4 plus 30 ($x_l + y_l$). In the case of a Unified Q-ary tree, 'F', 'S', and variable 'x' and 'y', play any major role.

Sample calculation of a Unified 4-ary & 8-ary Tree, for x_l, y_l, and T_l where F = 4, S = 8, x = 8, y = 88:

$$\log_F(2^x) + \log_S(2^y) = \log_4(2^8) + \log_8(2^{88})$$

$$\log_4(2^8) + \log_8(2^{88}) = \frac{\log_{10}(2^8)}{\log_{10}(4)} + \frac{\log_{10}(2^{88})}{\log_{10}(8)}$$

$$\log_4(2^8) + \log_8(2^{88}) = 4 + 30 = 34$$

Therefore, $x_l = 4$, $y_l = 30$, and $T_l = 34$. Note that the outcome in decimal is rounded up to the nearest whole number since level of tree cannot be fractioned.

From Table 4, by combining two *Q-ary trees*, for example binary tree and 4-ary tree, T_l for this combination is different to T_l of the Naive 4-ary tree and T_l of the Naive binary tree. The question is, would the difference between levels of tree have any impact on the total number of *Idle cycles* and number of *Collision cycles* for all tags identification within one interrogation zone? *Identification delay* can be reduced by reducing number of queries, which is a summation of Idle cycles, Collision cycles and Success cycles. *Success cycles* will always be the same using any combination techniques, which leaves Idle cycles and Collision cycles. Furthermore, *Number of bits* per query need to be taken into consideration since different Q-ary tree uses different Number of bits for each query. Thus, the following paragraph shows the difference between performances of the *Naive* approach versus *Unified* approach, and the impact from number of Idle cycles, number of Collision cycles, and total Number of bits.

$\log_F(2^x)$ $\log_S(2^y)$		x = 8 y = 88			x = 36 y = 60			x = 60 y = 36		
F	**S**	x_l	y_l	T_l	x_l	y_l	T_l	x_l	y_l	T_l
2	**2**	8	88	96	36	60	96	60	36	96
4	**4**	4	44	48	18	30	48	30	18	48
8	**8**	2	30	32	12	20	32	20	12	32
16	**16**	2	22	24	9	15	24	15	9	24
2	**4**	8	44	52	36	30	66	60	18	78
4	**2**	4	88	92	18	60	78	30	36	66
2	**8**	8	30	38	36	20	56	60	12	72
8	**2**	3	88	91	12	60	72	20	36	56
4	**8**	4	30	34	18	20	38	30	12	42
8	**4**	3	44	47	12	30	42	20	18	38
2	**16**	8	22	30	36	15	51	60	9	69
16	**2**	2	88	90	9	60	69	15	36	51
4	**16**	4	22	26	18	15	33	30	9	39
16	**4**	2	44	46	9	30	39	15	18	33
8	**16**	3	22	25	12	15	27	20	9	29
16	**8**	4	30	34	9	20	29	15	12	27

Table 4: Each combination 1 to 16 are applied with three different variables of 'x' and 'y', where x_l = number of highest tree level for 'F'; and y_l = number of highest tree level for 'S'. T_l, which is the summation of x_l and y_l, also represents the number of queries needed for the worst case of identification where two collided tags have all identical bits except for the last bit (bit 96).

Sample comparison between a performance of Naive approach versus Unified approach: We are now initiating a comparison between the performance of Naive binary tree, Naive 4-ary tree, Unified binary & 4-ary tree, and Unified 4-ary & binary.

Figure 5 shows a comparison between *Unified* approach (binary & 4-ary, 4-ary & binary) and *Naive* approach (4-ary, binary) on the five EPC data. We can see that the Naive 4-ary tree have the shortest level of tree, however, by examining Table 6, 4-ary tree does not have the lowest *Total number of bits*. This proves that *levels of tree* have an impact on the Total number of bits and *Overall cycles*, but does not necessarily result in the best performance of tree.

In order to calculate a *Total number of bits* required for the whole identification process, information on *Number of Child Nodes* (NCN) for each level of tree and *Number of Bits per Query* (NBQ) for that specific level, is needed. *Number of bits per Level* (NBL) can be calculate as follows:

$$NBL \;=\; NCN \times NBQ$$

Level	**2**	**2, 4**	**4, 2**	**4**
1	2	2	8	8
2	4	4	16	16
3	6	6	24	24
4	8	8	32	32
5	10	10	18	40
6	12	12	40	48
7	14	14	22	-
8	16	16	24	-
9	18	40	-	-
10	40	48	-	-
11	22	-	-	-
12	24	-	-	-
NBLs	**176**	**160**	**184**	**168**

Table 5: Calculation of Total memory bits required for two Naive Q-ary trees and two Unified Q-ary trees. NBLs shows the Total number of bits required for the specific Naive/Unified Q-ary tree.

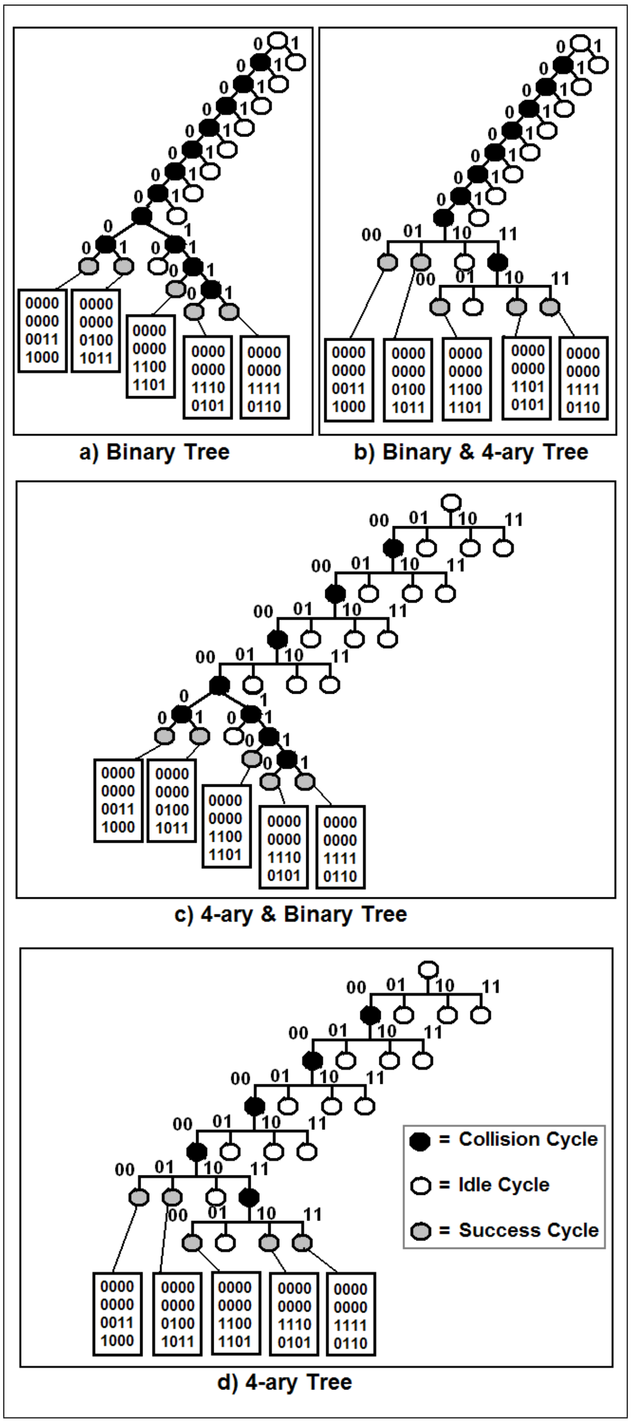


Figure 5: Identification processes of: a) Naive Binary tree, b) Unified Binary & 4-ary tree, c) Unified 4-ary & Binary tree, and d) Naive 4-ary tree.

After calculating the NBL for each level of tree, the *Total number of bits* (NBLs) required can be found by doing the summation of all NBL. For example, in Figure 5 a) it can be seen that the tree has 12 levels where all levels except Level 10 have 2 child nodes each. For each Level, NBQ increased by 1-bit since this is a Naive binary tree. Thus, NBL for each level are (NCN x NBQ): 2 or (2x1), 4 or (2x2), 6 or (2x3), 8 or (2x4), 10 or (2x5), 12 or (2x6), 14 or (2x7), 16 or (2x8), 18 or (2x9), 40 or (4x10), 22 or (2x11), 24 or (2x12) respectively. After adding all NBL together, the NBLs of 176-bits is as shown in Table 5.

Table 6 shows that both Naive binary tree and Unified 4-ary & binary tree, has the same number of *Overall cycles*. However, the *Total number of bits* for the two approaches is different. The same goes with Naive 4-ary tree and Unified binary & 4-ary tree where Overall cycles are the same but have a different Total number of bits. As for the impact of EPC data, we can see that when EPC IDs are identical (bit 1-8), a binary tree works better since it uses less *Number of bits* than 4-ary tree. This difference cannot be seen without calculating a proper Total number of bits since for the 'F', both binary and 4-ary tree have the same number of *Collision cycles* and *Idle cycles*. However, for each of these cycles, different Number of bits are used for querying, thus 4-ary tree uses more bits than binary tree. For 'S', 4-ary tree uses less Number of bits than binary tree since the number of Collision cycles happened more in binary tree. Although a 4-ary tree produces more Idle cycles than binary tree in the second half, it still produces less total number of Collision cycles and Idle cycles. We can now conclude that for *identical bits* of EPC, lower level tree can perform better than higher level one and for *unique bits* of EPC, a higher level tree is more suitable.

Combination	**2**	**2, 4**	**4, 2**	**4**
Collision Cycles (F)	8	8	4	4
Collision Cycles (S)	4	1	4	1
Total Collision Cycles	**12**	**9**	**8**	**5**
Idle Cycles (F)	8	8	12	12
Idle Cycles (S)	1	2	1	2
Total Idle Cycles	**9**	**10**	**13**	**14**
Success Cycles (F)	0	0	0	0
Success Cycles (S)	5	5	5	5
Total Success Cycles	**5**	**5**	**5**	**5**
Overall Cycles	**26**	**24**	**26**	**24**
Number of bits (F)	72	72	80	80
Number of bits (S)	104	88	104	88
Total number of bits	**176**	**160**	**184**	**168**

Table 6: Results of Collision cycles, Idle cycles, Success cycles, Overall cycles, Number of bits, and Total number of bits for 5 tags identification using Naive approach and Unified approach.

Number of tags in the interrogation zone also has an impact on *Collision cycles* and *Idle cycles*. When number of tags increases, number of collision also increases, thus higher level trees are more suitable since they provide more unique queries at each level of tree. In this paper, we will only test the combination of binary and 4-ary tree and analyse a performance to see how much improvement can be formed. In the future, further study will be done with higher level trees and a larger number of tags will be used for an experiment.

5 Experiment and Results

In this section, we present experiment conducted to evaluate the performance of "Unified Q-ary tree". As a result, analysis discussions are evaluate between the performance of *Naive* approach versus *Unified* approach.

5.1 Environment

To study the proposed "Unified Q-ary Tree" and compare with the performance of *Naive* approach, experiments are performed according to a Crystal warehouse scenario. The experiment is set up in a well controlled environment where there is no metal or water nearby. A *UHF RFID reader* is used and mounted on a dock door at the end of a conveyor belt. *Passive RFID tags* are attached to each case of crystal ware. Each pallet of wine glasses, plates, bowls are moved along this conveyor belt. At this stage, we assume that all three pallets move-in and move-out at the same time to an interrogation zone and no *arriving tag* or *leaving tags* are present during each identification round.

Specification: An Intel Pentium 4 CPU with 2.80GHz processor and 2GB RAM is used for testing. A Microsoft Window XP professional with Service Pack 3 is installed on the computer. Algorithms are implemented using Java JCreator.

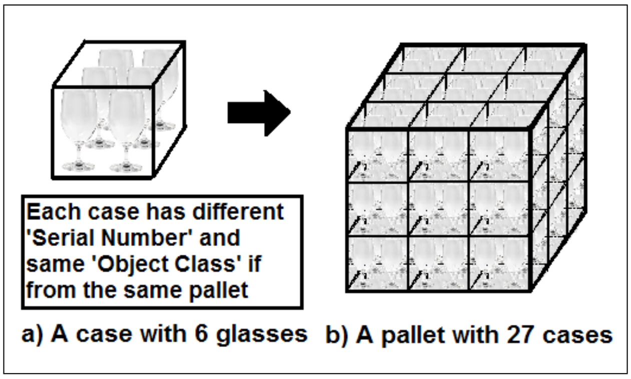

Each case has different 'Serial Number' and same 'Object Class' if from the same pallet
a) A case with 6 glasses
b) A pallet with 27 cases

Figure 6: Level-Packaging.

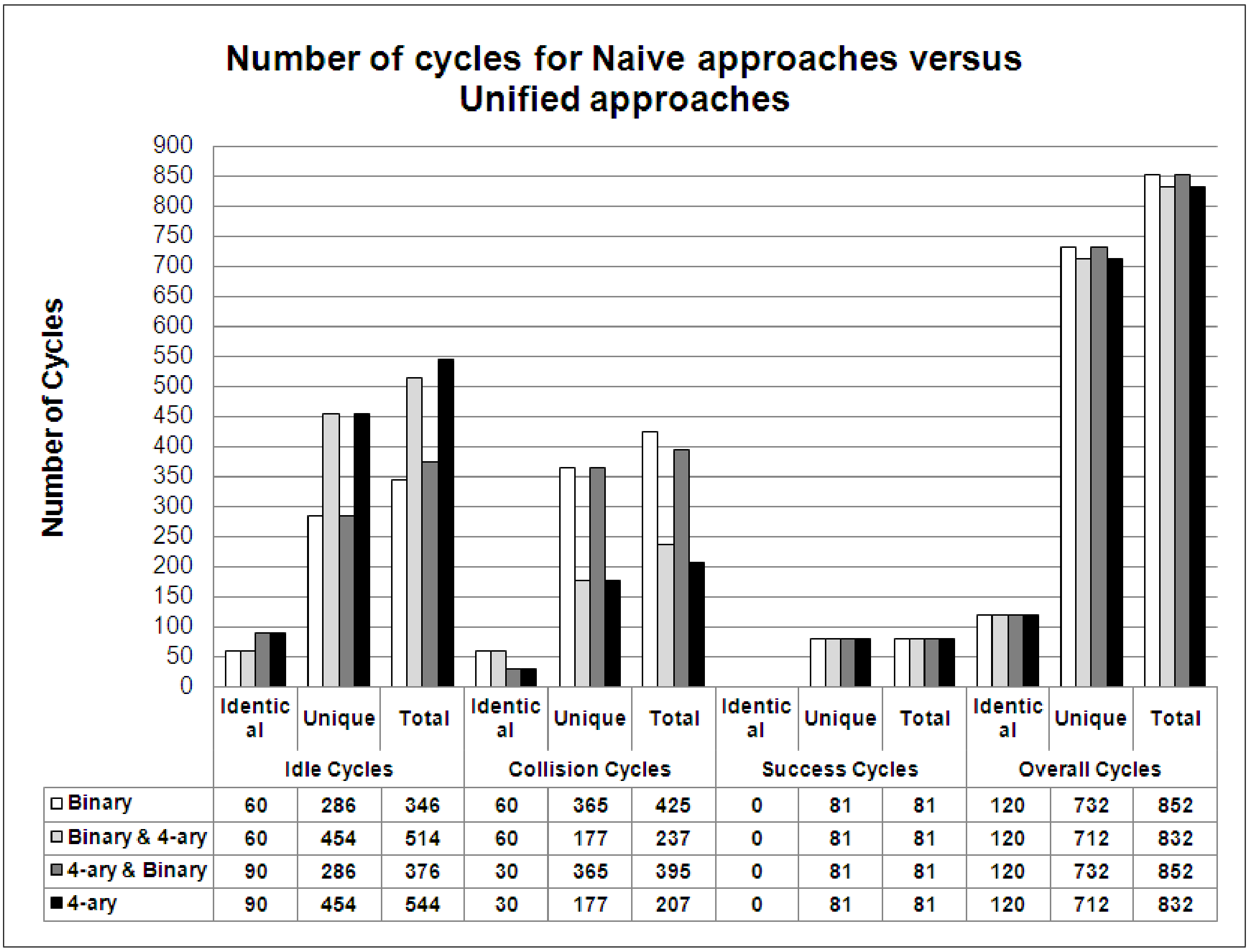
Figure 6 displays a level-packaging, where each case contains 6 glasses and each pallet contains 27 cases. For our experiment, 3 of these pallets will be visible to the reader attached to the dock's door next to the conveyor belt.

5.2 Data Set

Results presented are related to the first scenario mentioned earlier in Section 4.2. We performed ten runs on the data set and present the average results. For the data set, there are 81 tags/EPC used in the experiment. Each tag contains 60 *identical bits* for 'F' and 36 *unique bits* for 'S'. Each pallet contains 27 tags (See Figure 6) and 3 pallets are assumed to be visible to the reader each time. We applied the *Naive* approach, binary and 4-ary tree, to the data set with no partition. On the other hand, we applied *Unified* approach to the data set using 'x' = 60 and 'y' = 36 based on the nature of scenario one where the first 60-bits are identical.

5.3 Result, Analysis and Discussion

Based on the experiment simulation, Figure 7 shows the average results, from ten runs, on all four combinations: Naive binary, Unified binary & 4-ary, Unified 4-ary & binary, and Naive 4-ary tree. From Figure 7, we can see that the Naive 4-ary tree produced the most *Idle cycles* while the Naive binary tree produced the least. In contrast, the Naive binary tree produced the most *Collision cycles* while the Naive 4-ary tree produced the least. Both Naive binary and Unified 4-ary & binary have the same total number of cycles, which corroborate our methodology. In addition, the total number of cycles for Naive 4-ary tree and Unified binary & 4-ary tree are also equal. The total number of cycles can, at one point, clarify the performance of all four methods. We notice that both Naive 4-ary tree and Unified binary & 4-ary tree have less total cycles than binary and 4-ary & binary. This means that these first two methods will use less *Number of bits* in querying for all 81 tags than the other two. However, without looking into the actual results of



	Idle Cycles			Collision Cycles			Success Cycles			Overall Cycles		
	Identical	Unique	Total	Identical	Unique	Total	Identical	Unique	Total	Identical	Unique	Total
□ Binary	60	286	346	60	365	425	0	81	81	120	732	852
□ Binary & 4-ary	60	454	514	60	177	237	0	81	81	120	712	832
▨ 4-ary & Binary	90	286	376	30	365	395	0	81	81	120	732	852
■ 4-ary	90	454	544	30	177	207	0	81	81	120	712	832

Figure 7: Results of two Naive approaches (Binary, 4-ary) and two Unified approaches (Binary & 4-ary, 4-ary & Binary) for number of Idle cycles, Collision cycles, Success cycles, and Overall cycles.

Number of bits, we still cannot conclude which of the two methods will achieve less *identification time* for querying.

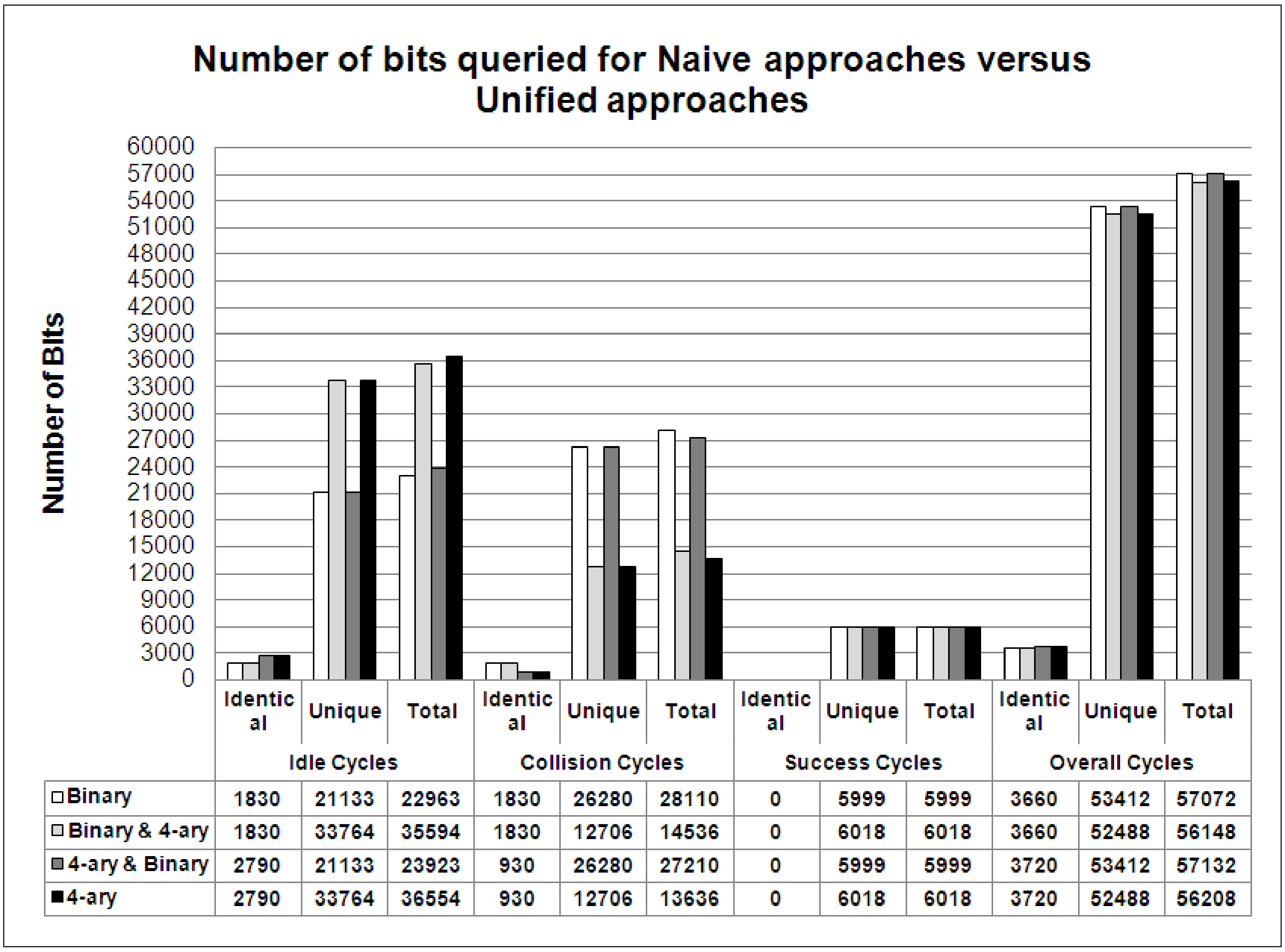
Based on Figure 7 we are now aware of *Success cycles* of all four methods are all equal to 81, which means that all tags in the interrogation zone are 100 percent identified. We can also see that all 81 tags were recognised at the later stages, where all bits (bit no. 61-82) are unique. As for *identical bits* of *Idle cycles* and *Collision cycles*, the sum of Idle cycles and Collision cycles have an outcome of 120 cycles, which means that both methods of binary or 4-ary tree have no impact in the sense of cycles count but as mentioned earlier, we need to calculate the actual *Number of bits* in order to clarify the difference of the performance of both methods. The next Figure (Figure 8) shows the Number of Bits for Idle cycles, Collision cycles, Success cycles, and *Overall cycles*, of each method.

Figure 8 shows all the actual bits for all queries that occur during tags identification. We now notice that the Unified binary & 4-ary tree have the lowest *Number of bits* queried for entire identification process. This verify our theory that by using a lower level tree for *identical bits* of EPC and higher level tree for *unique bits* of EPC, Number of bits queried can be minimised and identification process can be accelerated. There is not much difference in results but we can assume that as the number of tags in an interrogation zone increases, and other combinations of *Q-ary tree* are used, we will be able to see more differences in the outcome.

For *identical bits* of EPC, there is a slight difference between the *Number of bits* queried by the four methods. While Figure 7 shows that there is no difference between total number of cycles for identical bits for all four methods, we can see clearly that *Total number of bits* is different for each case in Figure 8. This is because each query inquired each time issues different Number of bits. For example, 4-ary tree issues 2 extra bits from the last query (from the parent node), while binary tree only append 1 extra bit to the last query. The Unified binary & 4-ary tree performed the best overall and required 60 bits less than the Naive 4-ary tree, and 924 bits less than the Naive binary tree. In contrast, the Unified 4-ary & binary tree performed the worst out of all four methods. This is because a higher level tree was used at the earlier stages of identification where all bits are identical. This means that more than 75 percent of the queries were *Idle cycles* which are waste of resources (See Figure 8 - 4-ary & Binary; Idle cycles:Collision cycles = Ratio of 3:1 or 75%:25%). By using binary tree instead of 4-ary tree for identical bits, 60 bits of queries were reduced (3720 minus 3660).

For *unique bits* of EPC, Number of bits query rises rapidly compared to *identical bits*. Figure 8 shows that, by using 4-ary tree for unique bits of EPC, number of queries and bits were slightly reduced (see Total bits queried for unique bits). The performance of each method on unique bits of EPC will be specified in detail in Figure 9.

Figure 9 shows the number of *Idle cycles*, *Collision cycles*, *Success cycles* and *Overall cycles* produced in each query loop. We can see that at bit 63-64 to bit 65-66, the difference between Overall cycles of binary



	Idle Cycles			Collision Cycles			Success Cycles			Overall Cycles		
	Identical	Unique	Total	Identical	Unique	Total	Identical	Unique	Total	Identical	Unique	Total
□Binary	1830	21133	22963	1830	26280	28110	0	5999	5999	3660	53412	57072
□Binary & 4-ary	1830	33764	35594	1830	12706	14536	0	6018	6018	3660	52488	56148
▨4-ary & Binary	2790	21133	23923	930	26280	27210	0	5999	5999	3720	53412	57132
■4-ary	2790	33764	36554	930	12706	13636	0	6018	6018	3720	52488	56208

Figure 8: Results of two Naive approaches (Binary, 4-ary) and two Unified approaches (Binary & 4-ary, 4-ary & Binary) for total number of bit queried for Idle cycles, Collision cycles, Success cycles, and Overall cycles.

and 4-ary tree grows. After bit 67-68, there is not much difference between the two. From bit 73-74 to bit 79-80, there are no Success cycles for both methods; therefore, there are no differences for their Overall cycles. We can now assume that at bit 61-62 to bit 71-72, the EPC are similar but not identical, which results in the unstable change in number of Overall cycles. On the other hand, at bit 73-74 to bit 79-80, we can assume that all bits become identical again resulting in no change in Overall cycles. The number of collided tags at bit 73-74 to bit 79-80 are exactly two since the ratio of Idle cycles to Collision cycles is 1:1 for binary tree and 3:1 for 4-ary tree respectively. At last, all tags were identified at bit 81-82 resulting in the same number of Overall cycles for both binary tree and 4-ary tree.

We can now summarise that by using a lower level tree for *identical bits* of EPC, and by using a higher level tree for *unique bits* of EPC, the *Total number of bits* for querying can be decreased. By reducing the Total number of bits, *identification time* for each round can be minimised.

6 Conclusion

In this study, we identified the significance of RFID *tags anti-collisions* and developed efficient method to minimise the use of memory bits; and at the same time to ensure that all RFID tags are 100 percent identified, which is essential to provide correct RFID data before they can be further processed, transformed, and integrated for RFID-enabled applications. We proposed a "Unified Q-ary tree", which combines two *Q-ary trees* together in order to reduce *Collision cycles* and *Idle cycles*; and to minimise *identification time*. In the experimental evaluation, we showed that our method performs better, ensures 100 percent tags identification and reduces *Overall cycles*, and *Total number of bits* queried, which leads to faster identification time.

As of future work, we intend to test other combinations of *Q-ary trees*. Different pallet sizes will also be inspected to determine the impact of packaging and density on quality of captures data. Different *number of tags* will be tested for the impact of number of tags within one interrogator zone. Also, different *Encoding Scheme* other than GID-96 will be observed.

Acknowledgements

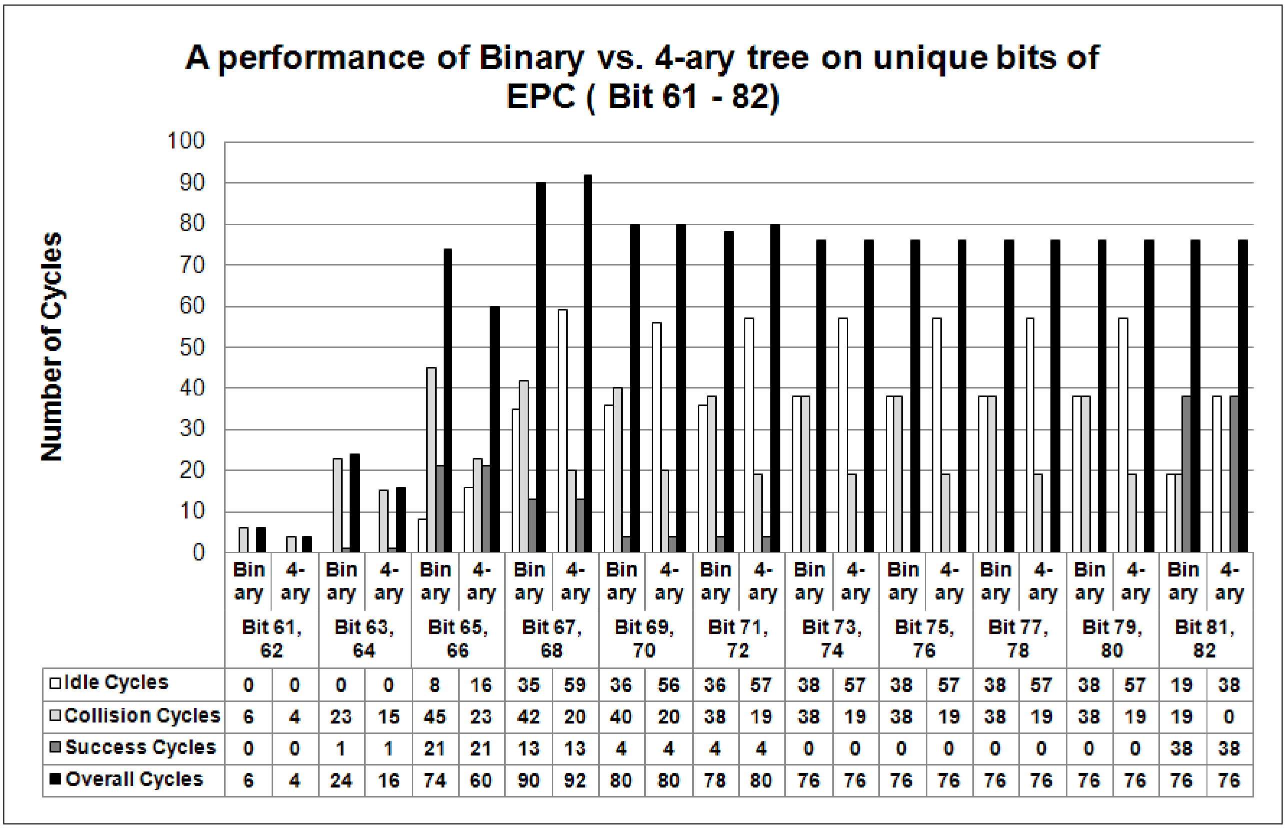
This research is partly sponsored by ARC (Australian Research Council) grant no DP0557303.

References

Bai, Y., Wang, F. & Liu, P. (2006), Efficiently Filtering RFID Data Streams, *in* 'CleanDB Workshop', pp. 50–57.

Brown, M., Patadia, S. & Dua, S. (2007), *Mike Meyers'Certification Passport: CompTIA RFID+ Certification*, McGraw-Hill.

Carbunar, B., Ramanathan, M. K., Koyuturk, M., Hoffmann, C. & Grama, A. (2005), Redundant Reader Elimination in RFID Systems, *in* '2nd Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'05)', pp. 176–184.



	Bit 61, 62		Bit 63, 64		Bit 65, 66		Bit 67, 68		Bit 69, 70		Bit 71, 72		Bit 73, 74		Bit 75, 76		Bit 77, 78		Bit 79, 80		Bit 81, 82	
	Binary	4-ary	Binary	4-ary	Binary	4-ary	Binary	4-ary	Binary	4-ary	Binary	4-ary	Binary	4-ary	Binary	4-ary	Binary	4-ary	Binary	4-ary	Binary	4-ary
□Idle Cycles	0	0	0	0	8	16	35	59	36	56	36	57	38	57	38	57	38	57	38	57	19	38
□Collision Cycles	6	4	23	15	45	23	42	20	40	20	38	19	38	19	38	19	38	19	38	19	19	0
▨Success Cycles	0	0	1	1	21	21	13	13	4	4	4	4	0	0	0	0	0	0	0	0	38	38
■Overall Cycles	6	4	24	16	74	60	90	92	80	80	78	80	76	76	76	76	76	76	76	76	76	76

Figure 9: Performance Analysis of Binary tree vs. 4-ary tree on unique bits of EPC, Bit 61 - 68, until all tags are identified. Results of Idle cycles, Collision cycles, Success cycles, and Overall cycles are displayed.

Choi, J. H., Lee, H. J., Lee, D., Lee, H. S., Youn, Y. & Kim, J. (2008), 'Query Tree Based Tag Identification Method in RFID Systems'. www.freshpatents.com/Query-tree-based-tag-identification-method-in-rfid-systems-dt20080508ptan20080106383.php.

Derakhshan, R., Orlowska, M. E. & Li, X. (2007), RFID Data Management: Challenges and Opportunities, *in* 'IEEE International Conference on RFID 2007', Texas, USA, pp. 175–182.

EPCGlobal (2006), 'EPCGlobal Tag Data Standards Version 1.3: Ratified Specification'. http://www.epcglobalinc.org/standards/tds/.

Fishkin, K. P., Jiang, B., Philipose, M. & Roy, S. (2004), I Sense a Disturbance in the Force: Unobtrusive Detection of Interactions with RFID-tagged Objects, *in* 'UbiComp 2004: Ubiquitous Computing', Seattle, Washington, USA, pp. 268–282.

Gonzalez, H., Han, J. & Li, X. (2006), Mining Compressed Commodity Workflows from Massive RFID Data Sets, *in* 'CIKM '06: Proceedings of the 15th ACM international conference on Information and knowledge management', ACM Press, New York, NY, USA, pp. 162–171.

Gonzalez, H., Han, J., Li, X. & Klabjan, D. (2006), Warehousing and Analyzing Massive RFID Data Sets, *in* 'ICDE '06: Proceedings of the 22nd International Conference on Data Engineering', IEEE Computer Society, Washington, DC, USA, p. 83.

Jeffery, S. R., Alonso, G., Franklin, M. J., Hong, W. & Widom, J. (2006), Declarative Support for Sensor Data Cleaning, *in* 'Pervasive Computing', Springer Berlin/Heidelberg, pp. 83–100.

Jeffery, S. R., Garofalakis, M. & Franklin, M. J. (2006), Adaptive cleaning for RFID data streams, *in* 'VLDB'2006: Proceedings of the 32nd international conference on Very large data bases', VLDB Endowment, Seoul, Korea, pp. 163–174.

Landt, J. (2001), Shrouds of Time The history of RFID, An Aim Publication, Pittsburg, PA.

Myung, J. & Lee, W. (2006*a*), 'Adaptive binary splitting: a RFID tag collision arbitration protocol for tag identification', *Mob. Netw. Appl.* **11**(5), 711–722.

Myung, J. & Lee, W. (2006*b*), Adaptive splitting protocols for RFID tag collision arbitration, *in* 'MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing', ACM, New York, NY, USA, pp. 202–213.

Quan, C. H., Hong, W. K. & Kim, H. C. (2006), Performance Analysis of Tag Anti-collision Algorithms for RFID Systems, *in* 'Emerging Directions in Embedded and Ubiquitous Computing', Vol. 4097, Springer Berlin/Heidelberg, pp. 382–391.

Ryu, J., Lee, H., Seok, Y., Kwon, T. & Choi, Y. (2007), A Hybrid Query Tree Protocol for Tag Collision Arbitration in RFID systems, *in* 'MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing', IEEE Computer Society, Glasgow, UK, pp. 5981–5986.

Shin, J. D., Yeo, S. S., Kim, T. H. & Kim, S. K. (2007), Hybrid Tag Anti-collision Algorithms in RFID Systems, *in* 'Computational Science ICCS 2007', Vol. 4490, Springer Berlin/Heidelberg, pp. 693–700.

Score Aggregation Techniques in Retrieval Experimentation

Sri Devi Ravana **Alistair Moffat**

Department of Computer Science and Software Engineering
The University of Melbourne
Victoria 3010, Australia
{sravana, alistair}@csse.unimelb.edu.au

Abstract

Comparative evaluations of information retrieval systems are based on a number of key premises, including that representative topic sets can be created, that suitable relevance judgements can be generated, and that systems can be sensibly compared based on their aggregate performance over the selected topic set. This paper considers the role of the third of these assumptions – that the performance of a system on a set of topics can be represented by a single overall performance score such as the average, or some other central statistic. In particular, we experiment with score aggregation techniques including the arithmetic mean, the geometric mean, the harmonic mean, and the median. Using past TREC runs we show that an adjusted geometric mean provides more consistent system rankings than the arithmetic mean when a significant fraction of the individual topic scores are close to zero, and that score standardization (Webber et al., SIGIR 2008) achieves the same outcome in a more consistent manner.

Keywords: Retrieval system evaluation, average precision, geometric mean, MAP, GMAP.

1 Introduction

Measurement is an essential precursor to all attempts to improve information retrieval (IR) system effectiveness. But to experimentally measure the effectiveness of an IR system is a non-trivial exercise, and requires that a complex sequence of tasks and computations be carried out. These tasks typically involve:

1. Selecting a representative corpus and a set of topics;
2. Creating appropriate relevance judgements that describe which documents are relevant to which topics;
3. For each of the systems being compared, building a set of runs, one for each of the topics;
4. Selecting one or more effectiveness metrics, and applying them to create a set of per-metric per-topic per-system scores;
5. For each effectiveness metric, comparing the scores obtained by one system against the scores obtained by the other systems, to determine if the difference in behavior between the systems can be assessed as being statistically significant; and then, finally,
6. Writing a paper that describes what the new idea was, and summarizing the measured improvement (or not) that was obtained relative to other previous techniques.

A variety of mechanisms have evolved over the years to perform these tasks. For example, because of the high cost of undertaking relevance judgements, steps 1 and 2 have tended to be carried out via large whole-of-community exercises such as TREC and CLEF. To perform step 3, we might build our own software system, or, to again make use of shared resources, we might choose to modify a public system such as Lemur, Terrier, or Zettair[1]. In the area of effectiveness metrics (step 4), the IR community has converged on a few that are routinely reported and can be evaluated via public software such as `trec_eval`. These include precision@10 (denoted here as P@10), R-precision, average precision (denoted as AP), normalized discounted cumulative gain (nDCG), rank-biased precision (RBP), and so on. Common tools and agreed techniques for step 5 are also emerging – there is now a clear community expectation that researchers must indicate whether any claimed improvements are statistically significant using an appropriate test (Cormack & Lynam 2007).

Now consider the last stage, denoted step 6 above. We wish to describe the new system in a context that allows the reader to appreciate the aspects of it that will lead to superior performance; and then need to provide empirical evidence that the hypothesized level of performance is attained. And, inevitably, we seek to do all of that within an eight or ten page limit. In support of our new system we describe the (established) corpus and topics that we have used in any training that we did to set parameters for our system and a baseline or reference system; and we describe the (different) corpus and/or topics that we then used to test the two systems with those parameters embedded. We can also succinctly summarize any statistically significant relationships between the two systems (the step 5 output): the sentence "System *New* was significantly better than System *Old* at the 0.05 level for all of X, Y, and Z" (where X, Y, and Z are effectiveness metrics) takes up hardly any space at all in our paper. Yet such a claim is the holy grail of IR research – provided, of course that System *Old* is a state-of-the-art reference point; that we had implemented it correctly; and that the experiments do indeed lead to the desired level of significance.

We would then like to provide evidence of the magnitude of the improvement we have obtained. To do that, we add a table of numbers to our paper. And here is the question that is at the heart of this work: what numbers? Two systems (at least) have been compared, over (probably) 50 or more topics, using (say) three effectiveness metrics. The experiments thus give rise to at least 300 per-system, per-topic, per-metric scores, and while we have all seen IR papers with that many numbers in them, including that

[1] See `http://www.lemurproject.org/`, `http://ir.dcs.gla.ac.uk/terrier/`, and `http://www.seg.rmit.edu.au/zettair/` respectively.

volume of data is rather less than ideal. Even worse, the effectiveness scores are all derived from the fifty underlying runs, each containing (typically) $1{,}000$ document identifiers. So if we wish to provide sufficient data that follow-up researchers can apply new effectiveness measures to the data, we need to publish $2 \times 50 \times 1{,}000 = 10^5$ "things" as the output of even the simplest two-system evaluation. (For example, TREC has accumulated the actual system runs lodged by the participating groups in over a decade of experimentation, and these now form an invaluable resource in their own right, and have been used in the experiments leading to this paper.)

Given that it is impossible for the data that was used as the input to our statistical testing to be included in our paper, but desirous of including at least some element of numeric output, we inevitably *average* the per-topic per-system per-metric effectiveness scores, to obtain a greatly reduced volume of per-system per-metric scores. We then populate a table in our paper with these numbers (perhaps as few as six of them in a one-collection, two-system, three-metric evaluation); use superscript daggers or bold font to indicate the relationships that are statistically significant, and submit our work for refereeing. This approach is now so prevalent that the phrase "mean average precision" has taken on a life of its own, and a table of "MAP" scores is an unavoidable part of every experimental IR paper, as if MAP was an axiomatic measure in its own right.

In this paper we take a pace back from this process, and ask two very simple questions that arise from the discussion above: what does it mean to "average" effectiveness scores? And, if it is indeed a plausible operation, is the arithmetic mean the most sensible way to do it, or are there other methods that should be considered?

2 Numeric aggregation

If a set of observations describes some phenomenon, it is natural to seek some kind of gross, or *aggregate* statistic that summarizes those observations. The simplest of these central tendencies is the *arithmetic mean*, which, for a set of t observations $\{x_i \mid i \in 1 \dots t\}$ is computed as:

$$\mathsf{AM} = \left(\sum_{i=1}^{t} x_i \right) / t \, ,$$

As examples, consider the four possible sets of $t = 5$ observations that are shown in Table 1. The arithmetic mean of example set S_1 is 0.28.

One point worth noting in connection with the arithmetic mean is that all of the values should be on the same scale – it is not possible to compute the average of five inches, ten centimeters, and 0.001 miles without first converting them to a common framework. Similarly, it is impossible to average three liters, five centimeters, and four kilograms, because they cannot be converted to common units.

Another key aggregation mechanism is the *geometric mean*, defined as the t th root of the product of the t numbers,

$$\mathsf{GM} = \left(\prod_{i=1}^{t} x_i \right)^{1/t} .$$

The geometric mean is more stable than the arithmetic mean, in the sense of being less affected by outlying values. However, when any of the values in a set is zero, the geometric mean over that set is also zero. For IR score aggregation purposes, this introduces the problem that a single "no answers" topic in the topic set might force all of the system scores to be zero. To sidestep this difficulty, Robertson (2006) thus defined an *ϵ-adjusted geometric mean*,

$$\epsilon\mathsf{GM} = \exp \left(\frac{\sum_{i=1}^{t} \log(x_i + \epsilon)}{t} \right) - \epsilon \, ,$$

where ϵ is a small positive constant, and the summation, exp, and log functions calculate the product and t th root of the set of t ϵ-adjusted values x_i in a non-underflowing manner. Because it is based on multiplication, in which there is no requirement that the quantities have the same units, it is permissible (although somewhat confusing) to take the geometric mean of, for example, three liters, five centimeters, and four kilograms. In this example, the GM is 3.91 (liters $\cdot$ centimeters $\cdot$ kilograms)$^{1/3}$. Further examples of GM and ϵGM are shown in Table 1.

A variant of ϵGM-AP in which ϵ is applied to AP scores in a thresholding sense rather than in an additive/subtractive sense is now one of the aggregate scores routinely reported by the `trec_eval` program (see `http://trec.nist.gov/trec_eval`) using $\epsilon = 10^{-5}$ (Voorhees 2005):

$$\epsilon\mathsf{GM}_{\texttt{trec\_eval}} = \exp \left(\frac{\sum_{i=1}^{t} \log \max\{x_i, \epsilon\}}{t} \right) .$$

For example, the $\epsilon\mathsf{GM}_{\texttt{trec\_eval}}$ score for sequence S_2 in Table 1 is 0.039, and would be 0.157 with $\epsilon = 0.01$, the value used in the table. Note that when ϵ approaches ∞ the additive/subtractive ϵGM score for a set of numbers (but not the thresholded $\epsilon\mathsf{GM}_{\texttt{trec\_eval}}$ variant) approaches the AM score for the same set. For this continuity reason, we prefer the ϵGM-AP additive/subtractive version to the `trec_eval` version, and have primarily used the former in this paper.

The *harmonic mean* is another central tendency that is typically used to combine rates, and can also be used as a method for score aggregation. It is defined as the reciprocal of the average of the reciprocals,

$$\mathsf{HM} = \frac{t}{\sum_{i=1}^{t} (1/x_i)} .$$

The harmonic mean is undefined if any of the set values are zero, meaning that it is again convenient to make use of an ϵ-adjusted version:

$$\epsilon\mathsf{HM} = \frac{t}{\sum_{i=1}^{t} 1/(x_i + \epsilon)} - \epsilon \, .$$

When ϵ is large, the ϵHM score again converges to the AM score.

The fourth and final central tendency explored in this paper is the *median*, denoted here as MD. The median of a set is the middle value of the set when they are sorted into numeric order: $x_{(t+1)/2}$ when t is odd, and $(x_{t/2} + x_{t/2+1})/2$ when t is even. The median has the benefit of being relatively unaffected by outliers, but the flip side of this is that it is completely insensitive to changes of value that do not affect the set ordering except when it is the middle values that change.

Table 1 shows the application of these four aggregation methods, plus two ϵ-adjusted variants, to four example data sets, with the largest value in each column picked out in bold. It is apparent that the aggregation techniques have different properties, since the four sequences are placed into different "overall" orderings by the various aggregation techniques. There is, of course, no sense in which any of systems S_1 to S_4 in Table 1 is superior to the others (presuming that the five elements in each sequence can

System	Scores					AM	GM	ϵGM	HM	ϵHM	MD
S_1	0.1	0.1	0.3	0.8	0.1	**0.280**	0.189	0.192	0.145	0.148	0.100
S_2	0.0	0.4	0.2	0.4	0.3	0.260	0.000	0.151	–	0.034	**0.300**
S_3	0.1	0.5	0.3	0.2	0.2	0.260	**0.227**	**0.228**	0.197	0.200	0.200
S_4	0.2	0.2	0.3	0.2	0.2	0.220	0.217	0.217	**0.214**	**0.214**	0.200

Table 1: Example sets of values corresponding to different systems applied to $t = 5$ topics, and their calculated central tendencies. In the ϵGM and ϵHM methods, $\epsilon = 0.01$. The values in bold are the largest in each column.

be regarded as paired observations and then a statistical significance test applied), so no answer is possible to the question "which system is better in the sense of having the highest score?" Nevertheless, and despite that patent lack of demonstrable differentiation, as soon as an aggregate score has been computed for the observations generated by some system, it is immediately tempting to then "order" the systems by their aggregate scores – exactly as we have in Table 1 by presenting the sequences in decreasing AM order. In Table 1, system S_1 at face value "outperforms" the other three systems by a quite wide margin.

Robertson (2006) provides an insightful discussion of measures and how they apply to AP and other effectiveness scores, including the relationship between AM-AP and GM-AP. Our discussion here can be seen as extending Robertson's evaluation, through the use of experiments in which aggregation methods are used to represent the overall performance of retrieval systems. In order to understand the effects of using these aggregation methods and their ability to produce consistent system rankings, evaluations were conducted using various TREC collections, and different types of effectiveness measure. Our experiments indicate that ϵGM handles variability in topic difficulty more consistently than does the usual AM aggregation method, and also better than the median MD and harmonic mean HM methods, when a significant fraction of the individual topic scores are close to zero. Also of considerable interest is that the standardized average precision scores of Webber et al. (2008a) achieve the same outcomes, even when coupled with the standard AM aggregation.

3 Topic hardness

The effectiveness of a retrieval system is gauged as a function of its ability to find relevant documents (Sanderson & Zobel 2005). One of the aims of the recent TREC Robust Track is to improve the consistency of retrieval technology by focusing on poorly performing topics – ones for which most of the participating systems score poorly (Voorhees 2003). The GM-AP aggregation method was introduced as part of this effort, in order to de-emphasize the role of high-scoring topics in system comparisons, and to enhance the relative differences amongst low-scoring topics (Voorhees 2005, Robertson 2006). Note, however, that changing to GM-AP has no effect on the significance or otherwise of any pairwise system comparison, since significance is a function of the elemental effectiveness scores, prior to any summary value being computed. The aggregation mechanism relates purely to the gross statistic that is presented as being the overall score for the system.

Mizzaro (2008) makes further observations in this regard. The importance of good relative performance over all topics, and not just excellent performance on one (which is how system S_1 obtains its high AM score in Table 1), and the fact that users remember any delivery of poor results by a system for a topic was also discussed by Mandl et al. (2008). Similarly, Buckley (2004*a*) points out that the topic variability is the main problem when designing an IR system for all user needs, and that a universal IR system should perform well across a range of topics with varying levels of difficulty.

It has been noted that a key expectation (or rather, hope) arising from any form of IR experimentation is that the system performance results based on one topic or one collection should be able to predict system performance on other topics and other collections (Buckley 2004*a*, Webber et al. 2008b). It has also been noted (see Buckley (2004*b*) and Webber et al. (2008a)) that topic variability is at least as great as system variability, and that in the nominal matrix of per-system per-topic scores, there is more commonality of scores across any particular topic than there is across any individual system. Restating this observation another way, the score achieved by a particular system on a given topic tends to be more a function of the topic than of the system.

In a similar vein, Mizzaro & Robertson (2007) argue that GM-AP is a more balanced measure than AM-AP for TREC effectiveness evaluations. This is due to the way in which the arithmetic mean can be influenced by easy topics, for which the system-topic scores are generally high, and bad systems might still get scores that are numerically large. Mizzaro & Robertson also asserted that GM-AP is not overly biased towards the low end of the scale, where the system-topic scores are low, and, equivalently, the topics are hard. Indeed, as was noted by O'Brien & Keane (2007), users prefer strategies and technologies that maximize the amount of information they gain as a function of the interaction cost that they invest.

Observations such as these then raise the question as to how best to measure topic "hardness". In the TREC 2003 Robust Retrieval Track, difficult topics were defined as being those with a low median AP score and at least one high outlier score (Voorhees 2003). Other definitions include computing (Mizzaro 2008)

$$D_t = 1 - \mathit{mean}_t \,, \qquad (1)$$

where mean_t is the average of the system-topic scores for topic t; or computing

$$D_t = 1 - \mathit{max}_t \,,$$

where max_t is the maximum score for topic t. For the purposes of the experiment, we took another approach, and defined the difficulty D_t of a topic t to be

$$D_t = \frac{\mathit{max}_t - \mathit{mean}_t}{\mathit{sd}_t} \,, \qquad (2)$$

in which standardized z-scores are calculated in the system-topic matrix (Webber et al. 2008a), and the most difficult topic is deemed to be the one for which the most "surprisingly good" score is obtained by one of the systems, with surprise defined in terms of standard deviations above the mean.

4 Methodology

Our purpose in this investigation was to examine the effect that the choice of score aggregation technique had on the outcomes of experiments, and whether the proposed use of the ϵGM adjusted geometric mean (Robertson 2006) could be experimentally supported in any way. To carry out this study, we devised the following experimental methodology. While we have no basis for proving that what we have computed has "real" meaning, we trust that the reader will find the experiment plausible (and interesting), and will agree that the results we have computed are grounded in practice.

We made use of standard TREC resources – collections, matching topics, and corresponding relevance judgements (see `http://trec.nist.gov`). We also used the official TREC runs, as lodged each year by the participating research groups, and were able to compute retrieval effectiveness scores for each of the submitted systems based on any subset of the topics that we wished to use. Al-Maskari et al. (2008) consider these test collections and support their use in IR experimentation, arguing that they can be used to predict users' effectiveness successfully.

Each of our main experiments proceeded by:

1. Choosing a random subset containing half of the set of t topics.
2. Extracting the rankings for those $t/2$ topics from the s available runs.
3. Using the chosen effectiveness metric and the relevance judgements to calculate a set of $st/2$ per-system per-topic scores.
4. Using the chosen score aggregation technique to compute a set of s per-system scores.
5. Sorting the s systems into decreasing score order, based on the per-system scores.
6. Repeating this process, using the other $t/2$ topics.
7. Taking the two s-item system orderings, and calculating the similarity between them using a mechanism such as Kendall's τ (Kendall & Gibbons 1990).
8. Then repeating this entire sequence $10{,}000$ times, so that $10{,}000$ Kendall's τ scores could be used to represent the self-consistency of the score aggregation technique.

We note that researchers have also applied this methodology in investigations examining other aspects of retrieval performance (Zobel 1998, Sanderson & Zobel 2005).

Part of the purpose of Table 1 was to illustrate the inconsistencies that can arise out of system "orderings" based on aggregate scores, and our experiments in this paper are intended to uncover the extent to which such inconsistencies are an issue in real IR experimentation. If an aggregation computation was "perfect", and if subsets of topics could be equally balanced (whatever that means), the two system orderings would be the same, and the Kendall's τ would be 1.0. Variation in aggregation technique, and variations in subset balance, mean that it is unlikely that Kendall's τ scores of 1.0 can be achieved. But, if the same (large number of) topic subsets are used for all aggregation methods, any consistently-observed difference in Kendall's τ can be attributed to the aggregation method.

5 Testing aggregation

For the initial experiments, the AP metric was coupled with a range of aggregation techniques. Use of AP in IR experimentation is widespread, and while it is normally regarded as being a "system" metric rather than a "user" one, it does still correspond to a (somewhat contrived) user model (Robertson 2008).

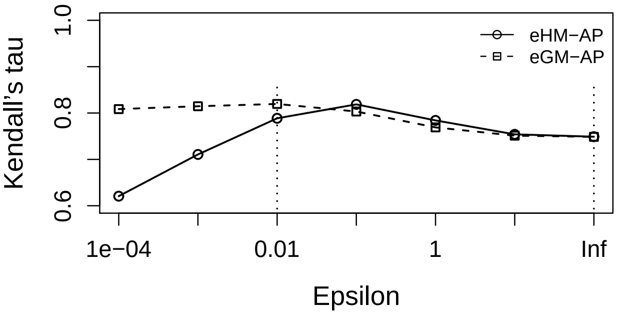


Figure 1: Average system ordering correlations when GM-AP and HM-AP are used as the score aggregation method across topics. In this experiment, the 50 TREC9 Web Track topics were randomly divided into equal-sized subset pairs, and the system rankings generated on those two subsets were compared using Kendall's τ. When the GM-AP and HM-AP parameter ϵ approaches infinity, the resultant system ordering approaches the ordering generated by AM-AP.

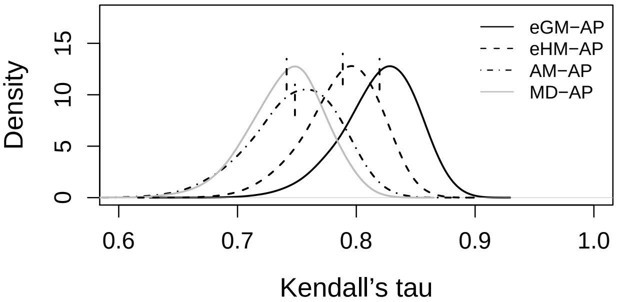


Figure 2: Density distribution of $10{,}000$ Kendall's τ values for ϵGM-AP, ϵHM-AP (both with $\epsilon = 0.01$), AM-AP, and MD-AP. In each experiment the TREC9 Web Track topics are randomly split into two halves, each system is scored using the topic subsets, and then the two resultant system orderings are compared. Three of the four curves represent density cross-sections corresponding to points in Figure 1. The MD-AP cross-section is additional. The vertical dotted lines on the curves indicate the means of the four density distributions.

5.1 Results using average precision

Figure 1 provides a first illustration of the data collected using the experimental methodology described in the previous section. In this graph ϵGM-AP and ϵHM-AP induced system orderings are compared for different values of ϵ, with the average value of Kendall's τ for random pairs of query subset-induced system orderings plotted as a function of ϵ. The line shows the average τ value over $10{,}000$ random splittings of the $t = 50$ TREC9 topics and $s = 105$ TREC9 systems, in an experiment designed to answer the very simple question as to whether either ϵGM-AP or ϵHM-AP should be preferred to AM-AP, and if so, what range of values of ϵ is appropriate.

The shape of the curve in Figure 1 makes it clear that when ϵ is small the average system ordering correlations are higher – that is, that the ϵGM-AP aggregate score (towards the left end of the graph) places the systems into rankings that are more self-consistent than does the conventional AM-AP summarization at the righthand end of the graph (when $\epsilon \to \infty$). The ϵHM-AP method is also better at ordering the systems than is AM-AP provided that mid-range values are chosen for ϵ. For small ϵ, the quality of the induced system rankings for ϵHM-AP drops markedly.

A major theme of this paper is that plain averages should be treated with caution, and we should heed our

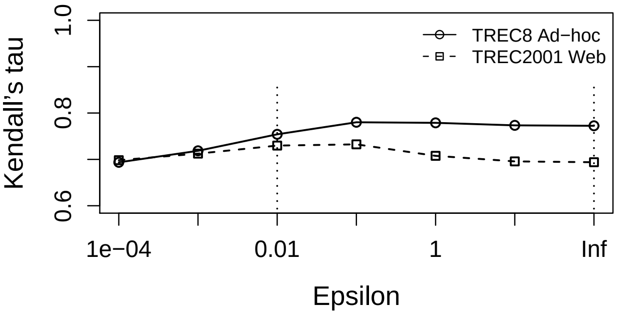


Figure 3: Average system ordering correlations when GM-AP and AM-AP (as $\epsilon \to \infty$) are used as the score aggregation method across topics. The two curves for the TREC8 Ad-Hoc Track ($t = 50$ topics, $s = 129$ systems) and the TREC2001 Web Track ($t = 50$ topics, $s = 97$ systems) can be directly compared with the ϵGM-AP curve in Figure 1.

own advice in this regard. Figure 2 shows the density distribution of the $10{,}000$ τ values at three of the points plotted in Figure 1, showing the variability of the system orderings when ϵGM-AP and ϵHM-AP (both with $\epsilon = 0.01$) and AM-AP are used to aggregate the per-system per-topic and generate the system orderings. Also shown as a fourth line is the τ density curve for the system similarities generated using the median AP score as the gross system statistic. The density curves are derived from $10{,}000$ random splittings of the 50 TREC9 topics into two 25-topic subsets. The system orderings produced by GM-AP aggregation are consistently more similar than the system orderings induced by AM-AP for the same topic splits, and suggest that GM-AP is the more stable aggregation technique for this data set. Analysis of the paired differences shows that all of the relationships shown are significant at the $p = 0.01$ level, that is, that ϵGM-AP $>$ ϵHM-AP $>$ AM-AP $>$ MD-AP with high confidence.

5.2 Other collections

Having used the TREC9 Web Track data to confirm that ϵGM-AP with $\epsilon = 0.01$ yields more consistent system rankings than does AM-AP, we turned to other TREC data sets in order to determine the extent to which that relationship is a general one. Figure 3 was created via the same experimental methodology as was used for Figure 1, but using the TREC8 Ad-Hoc Track queries and systems ($t = 50$, $s = 129$); and the TREC2001 Web Track queries and systems ($t = 50$, $s = 97$). Neither of these experiments favor ϵGM-AP compared to AM-AP, and for the TREC8 data set, ϵGM-AP with $\epsilon = 0.01$ is significantly less consistent that AM-AP ($p = 0.05$).

Figure 4 helps understand why this difference in behavior arises. It shows the distribution of the per-topic per-system AP scores for the three TREC data sets used in our experiments. The TREC9 collection, topics, and judgements combination generates a high number of low scores compared to the other two data sets, and it is these low scores that the ϵGM method is handling better. For example, in the TREC9 results, 10% of the system-topic scores are zero, and a further 46% are below 0.1, making a total of 56% low scores, shown in the first row of Table 2. For TREC2001 the corresponding rates were 4% and 45%, totalling 49%; and for TREC8 the rates were 3% and 33%, totalling 36%. That is, in the TREC2001 and TREC8 experiments there were fewer low AP values in the score matrix, AM-AP suffers less vulnerability, and there is thus less scope for ϵGM-AP to be superior.

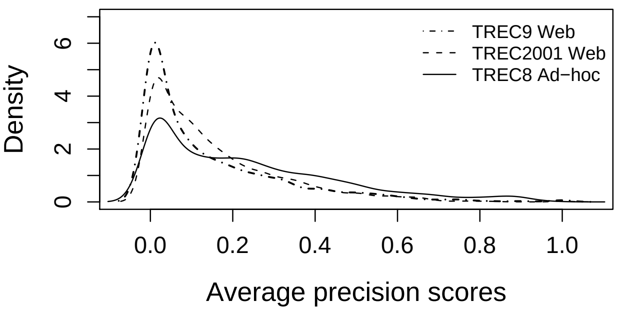


Figure 4: Density distribution of AP scores for TREC9 Web Track data ($t = 50$ topics and $s = 105$ systems); TREC8 Ad-Hoc Track data ($t = 50$ topics and $s = 129$ systems); and TREC2001 Web Track data ($t = 50$ topics and $s = 97$ systems).

Metric	$\% = 0$	$\% \leq 0.1$
AP	10	56
P@10	37	37
nDCG	10	23
RBP0.95	17	52
SP	0	4

Table 2: Proportion of low scores among the TREC9 system-topic combinations when assessed using different effectiveness metrics.

5.3 Effectiveness measures

Figure 5 and Figure 6 show what happens in the TREC9 environment when AP is replaced by P@10, RBP0.95 (see Moffat & Zobel (2009) for a definition of this metric), and nDCG (see Järvelin & Kekäläinen (2002)) as the underlying similarity measure. The score density plot in Figure 6 suggests that ϵGM-nDCG should be stable as ϵ is changed, because of the low density of nDCG near-zero scores, and that is what is observed in Figure 5. On the other hand, RBP0.95 has a relatively high density of near-zero scores, and ϵGM-RBP0.95 is accordingly sensitive to ϵ, with more consistent system rankings being generated with $\epsilon = 0.1$ than when ϵ tends to ∞ and the arithmetic mean is being used. Note also the comparatively poor performance of P@10 – it is relatively immune to the choice of ϵGM or AM aggregation, but nor is it a terribly good basis for ordering systems.

Webber et al. (2008a) recently introduced a *standardized* version of AP that we denote here as SP. The critical difference between AP and SP is that a set of t topic means and standard deviations are computed across the st per-system per-topic scores, and each of the st scores is then converted into a z score with regard to that topic's statistics:

$$e'_{s,t} = \frac{e_{s,t} - mean_t}{sd_t} ,$$

where $e'_{s,t}$ is the z-score corresponding to the average precision effectiveness $e_{s,t}$. Because z-scores are centered on zero, and can be negative as well as positive, Webber et al. further transformed the $e'_{s,t}$ values through the use of the cumulative normal probability distribution. The result is a set of $e''_{s,t}$ scores that lie strictly between zero and one, and which, for each topic, has a mean of 0.5 and a uniform standard deviation. The last row of Table 2 shows the rather unique properties of the set of effectiveness scores that result from this two-stage standardization of AP scores to generate SP scores.

Given the intention of the standardization process, it is unsurprising (Figure 7) that there is no change in sys-

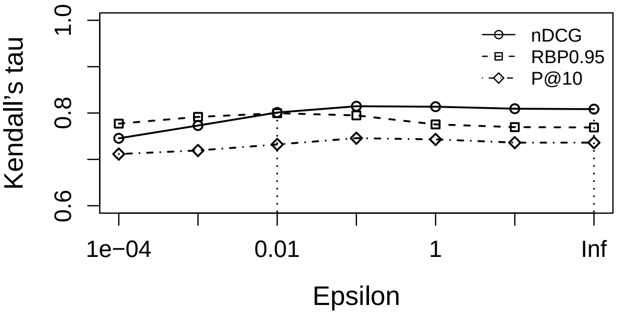


Figure 5: Average system ordering correlations when ϵGM-P@10, ϵGM-RBP0.95, and ϵGM-nDCG are used to score the TREC9 systems. When ϵ approaches infinity the methods converge to AM-P@10, AM-RBP0.95, and AM-nDCG respectively.

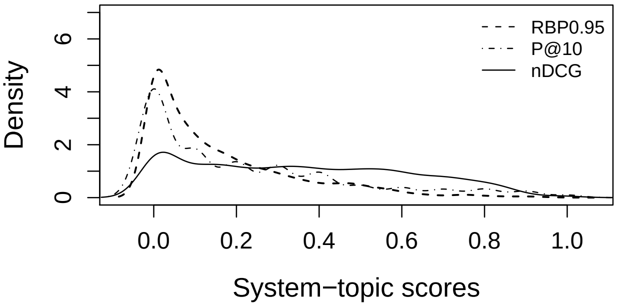


Figure 6: Density distribution of TREC9 system-topic scores using RBP0.95, P@10, and nDCG ($t = 50$ topics and $s = 105$ systems).

tem ranking consistency as ϵ is changed; and also unsurprising (Figure 8, Table 2) that the fraction of SP scores that are near zero is small. That is, the condition that led to geometric mean average precision being proposed as an alternative to arithmetic mean average precision – the presence of important low-scoring topics in amongst the high-scoring ones – is removed by the standardization process, and there is no benefit in shifting to ϵGM aggregation. Instead, AM aggregation, with the elimination of the need to select ϵ, is the preferred approach, because standardization means that all topics are already contributing equally to any assessment in regard to differences in system effectiveness, and no further benefit arises from emphasizing low scores.

Note that standardization is only possible when a set of retrieval systems are being mutually compared, and topic means and standard deviations can be calculated; or when pooled relevance judgements based on a set of previous systems are available, together with the runs that led to those judgements.

Figure 9 draws together the observations we have made. A total of fifteen collection and effectiveness metric couplings are shown, with the percentage of low effectiveness scores for that coupling plotted horizontally, and the extent of the superiority (or inferiority) of ϵGM as an aggregation method relative to AM plotted vertically. The behavior of the SP metric does not depend on a specific aggregation technique to perform well, and for all three collections used is insensitive to the score averaging mechanism. In the case of the other four metrics the trend is as we have observed already: if there are many low scores, ϵGM gives more consistent system orderings than does AM.

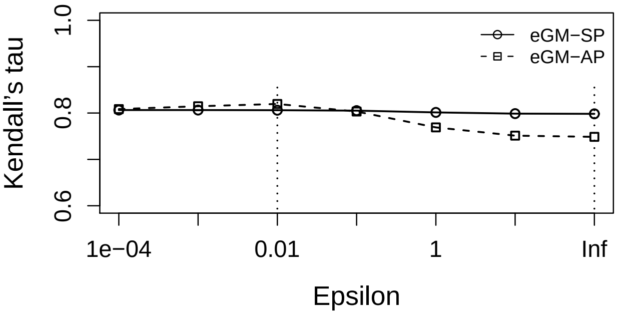


Figure 7: Average system ordering correlations when ϵGM-SP (based on standardized average precision) is used as the evaluation metric, compared to the ϵGM-AP combination. The methodology and data set used in this experiment are identical to those used in Figure 1.

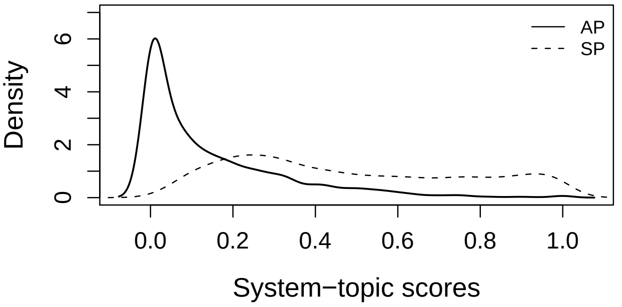


Figure 8: Density distribution of AP and SP scores for the TREC9 Web Track data ($t = 50$ topics and $s = 105$ systems).

5.4 Query "hardness"

All of the results presented thus far have been based on the average performance of the aggregation methods over large numbers of random splittings of the topics in the query set. As an alternative, we also constructed two particular topic subsets based on the nominal topic difficulty scores defined by Equation 2, to evaluate the extent to which the various aggregation mechanisms were affected by topic difficulty. In undertaking this part of the experimentation, we additionally sought to determine whether the additive/subtractive version of the adjusted geometric mean gave rise to consistent rankings, or whether the trec_eval ϵ-thresholding version was more consistent.

The two query partitionings were constructed to illustrate extreme behavior. In the first partitioning, the Hard/Easy split, Equation 2 was used to assign a difficulty rating to each of the $t = 50$ TREC9 topics, and then the highest-scoring 25 were taken into one set, and the lowest 25 topics were taken into the other. The second Middle/Rest partitioning again ranked the topics by difficulty, but then extracted the mid-scoring (most internally consistent) group into one subset, and left the remaining 12 hardest topics and 13 easiest ones in the other set. System orderings based on aggregate score over AP and SP for each pair of sets were then computed, and compared using Kendall's τ. The results of these experiments are shown in Table 3. The columns headed " Random" relate to the previous methodology of taking the average τ value over 10,000 random splittings of the 50 topics, and represent the numeric values of some of the points already plotted in the various graphs.

There are a number of trends that can be distilled from Table 3. Looking at the AP halves of the three subtables it is clear that the additive version of ϵGM is preferable to the ϵGM$_{\texttt{trec\_eval}}$ version – in eight of the nine cases, ϵGM detects more similarity between the system orderings,

Aggregation method	AP			SP		
	Random	Hard/Easy	Middle/Rest	Random	Hard/Easy	Middle/Rest
AM	0.748	0.694	0.722	0.798	0.704	0.820
ϵGM, $\epsilon = 0.01$	0.819	0.779	0.826	0.805	0.712	0.824
ϵGM$_{\texttt{trec\_eval}}$, $\epsilon = 0.00001$	0.798	0.818	0.800	0.806	0.802	0.826

(a) TREC9 data set, $t = 50$ and $s = 105$.

Aggregation method	AP			SP		
	Random	Hard/Easy	Middle/Rest	Random	Hard/Easy	Middle/Rest
AM	0.693	0.584	0.611	0.741	0.622	0.798
ϵGM, $\epsilon = 0.01$	0.729	0.650	0.674	0.740	0.622	0.798
ϵGM$_{\texttt{trec\_eval}}$, $\epsilon = 0.00001$	0.682	0.639	0.567	0.737	0.657	0.816

(b) TREC2001 data set, $t = 50$ and $s = 97$.

Aggregation method	AP			SP		
	Random	Hard/Easy	Middle/Rest	Random	Hard/Easy	Middle/Rest
AM	0.773	0.719	0.753	0.781	0.739	0.790
ϵGM, $\epsilon = 0.01$	0.755	0.732	0.760	0.769	0.714	0.785
ϵGM$_{\texttt{trec\_eval}}$, $\epsilon = 0.00001$	0.681	0.672	0.676	0.768	0.680	0.785

(c) TREC8 data set, $t = 50$ and $s = 129$.

Table 3: System ranking correlation coefficients using Kendall's τ, for two different effectiveness metrics, three different score aggregation methods, three different collections, and three different ways of splitting the topics into two halves. The three subtables are ordered by decreasing percentage of system-topic AP scores that are below 0.1. Note that the Hard/Easy and Middle/Rest splits are both one-off arrangements in each of the three data sets.

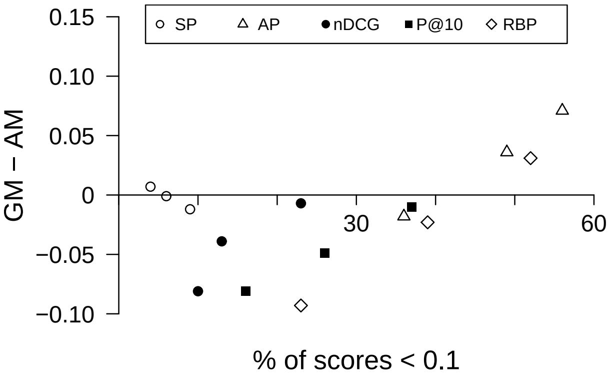


Figure 9: Comparing the ϵGM and AM score aggregation methods across three collections and across five effectiveness metrics. The horizontal axis shows the percentage of low effectiveness scores generated by that collection/metric combination; the vertical axis plots the difference in the Kendall's τ score obtained using ϵGM $-$ AM and $\epsilon = 0.01$.

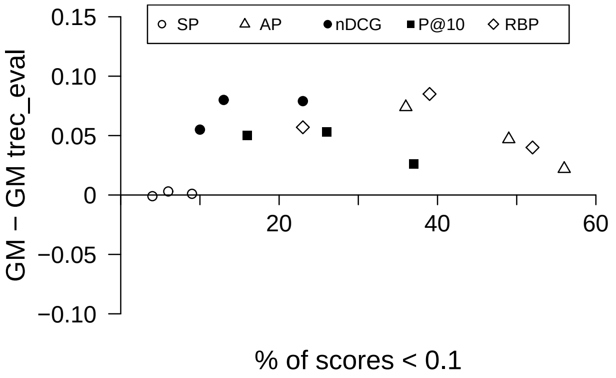


Figure 10: Comparing the ϵGM aggregation method and the thresholded GM variant used in the trec_eval program, with other experimental details as for Figure 9. The ϵGM approach is more self-consistent on all three collections, and for four of the five effectiveness metrics used.

including in all three of the Random evaluations.

Second, looking at the top row of each subtable, the AM-SP combination of effectiveness metric and aggregation method is uniformly better than the AM-AP combination that has dominated IR reporting for more than a decade. The third effect to be noted is that the Hard/Easy pairing, with just two exceptions in connection with the ϵGM$_{\texttt{trec\_eval}}$ aggregation, is more likely to lead to different system orderings than is the Middle/Rest topics split. And fourth, it also appears that the Middle/Rest is no less likely to generate different system orderings than a Random split, and hence there is no sense in which it provides a problematic arrangement for the aggregation metrics to handle. This is a slightly surprising outcome, since the Rest group contains topics of widely varying difficulty, at least in terms of Equation 2.

We also explored the use of Equation 1 as a topic difficulty rating, and in results not included here, obtained correlation patterns similar to those shown in Table 3.

Finally in this section, to reinforce our contention that trec_eval's thresholding ϵGM$_{\texttt{trec\_eval}}$ aggregation method is less reliable than the additive/subtractive ϵGM version, Figure 10 repeats the "differences between the average Kendall's τ score" experiment of Figure 9. The three points representing SP on the three different collections are again unaffected by the aggregation method used. But in all of the other twelve cases, ϵGM generates more consistent system rankings than does ϵGM$_{\texttt{trec\_eval}}$. We can see no basis for persisting with the thresholding version of GM-AP provided by trec_eval, and suggest that other authors be similarly vary of using the scores it generates.

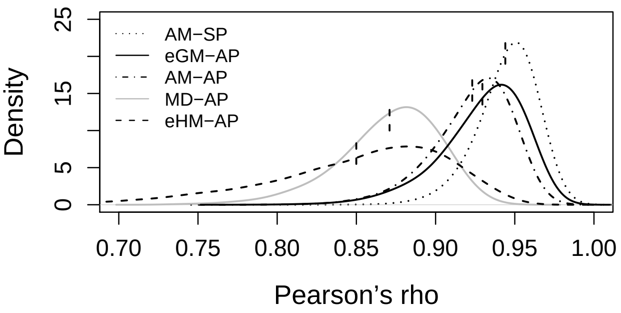


Figure 11: Density distribution of $10{,}000$ Pearson's ρ values for AM-AP, GM-AP ($\epsilon = 0.01$), HM-AP ($\epsilon = 0.01$), MD-AP, and AM-SP, based on $10{,}000$ random splittings of the TREC9 data set ($t = 50$ and $s = 105$). The experimental methodology was as for Figure 2.

5.5 Correlation coefficient methods

We have made extensive use of Kendall's τ in our analysis, starting from the presumption outlined in Section 1 that whether or not significance of any particular pairwise relationships had been established, it was likely that overall system scores would be used to derive system rankings, and thus, that study of score aggregation mechanisms was of merit. Use of Kendall's τ allowed the "closeness" of pairs of system rankings to then be quantified.

Kendall's τ takes into account the system ordering that is generated, but not the scores that led to that ordering, meaning that when there are clusters of near-similar scores, modest changes in the scores can lead to more dramatic changes in the correlation coefficient. An alternative metric that is based on scores rather than rankings is the Pearson product-moment coefficient. To verify that the relationships between rankings that have been noted above are not specific to the use of Kendall's τ, we repeated the experiments that led to Figure 2, using Pearson's ρ to compare pairs of lists of "system, score" pairs. The results are shown in Figure 11, again using the TREC9 resource.

Broadly speaking, Figure 11 shows the same trends as had been identified using Kendall's τ. The best method for obtaining a per-system score is the SP metric coupled with the AM averaging technique. Earlier in this paper we noted that because AM relied on addition, it could technically only be applied to values that were on the same scale, a restriction that did not apply to the geometric mean. The superior performance of AM-SP compared to AM-AP can be interpreted as a verification of this observation, since the process of standardizing the AP scores (subtracting the mean, and then dividing by the standard deviation for that topic) renders them into unitless values on a common scale. On the other hand, the pre-standardization AP values have different scales (in the sense of millimeters versus inches), because what is good performance on one topic might be substandard performance on another. In this sense, the process of standardization makes it "right" to then compute the arithmetic mean as the gross statistic for a system. And, even though the fully standardized scores are constrained to the $(0, 1)$ interval, the fact that equality is not possible at either end of the scale means that no matter how good (or bad) a system is on a particular topic, it is possible – at least in theory – for a different system to be better (or worse). That is, there are no bookends in SP that force systems to be considered to be "equal" on very easy or very hard topics.

Note also in Figure 11 the good performance of ϵGM – by ϵ-adjusting the scores, and computing a geometric mean, small effectiveness values are allowed to contribute to the final outcome. However use of the harmonic mean is not appropriate, and the HM-AP method is inferior to the baseline AM-AP approach. Nor – predictably – is the median an especially good score aggregation technique.

6 Conclusion

Our exploration of AM-AP and ϵGM-AP has confirmed that for the TREC9 Web data the ϵ-adjusted geometric mean is a more appropriate score aggregation mechanism than is the arithmetic mean. This appears to be a consequence of the large number of low AP scores (more than 50% are less than 0.1) across the TREC9 systems and topics. Experiments conducted with other TREC data resources confirm that when a collection has a majority of system-topic AP scores that are zero or close to zero, the ϵ-adjusted geometric mean is a more appropriate score aggregation method than is the arithmetic mean. On the other hand, when there are only a minority of low system-topic scores, the arithmetic mean is resilient, and tends to perform well, where "performs well" is in the sense of system orderings derived from one subset of the topics being similar to the system orderings derived from a different set.

We also experimented with other effectiveness metrics, including P@10, nDCG, RBP0.95, and SP, and observed the same overall outcome – when a large fraction of small effectiveness scores are generated by the metric, it is better to use the ϵGM aggregation approach.

On the other hand, experiments using the standardized precision (SP) metric showed that it anticipates the benefits brought about through the use of the geometric mean, and that the best aggregation rule for it was the standard arithmetic mean. The SP metric has the useful attribute of transforming the effectiveness scores so that, for every topic in the set, the mean score for that topic is 0.5, and the standard deviation is also fixed. Standardizing thus converts all system-topic scores to the same "units", and allows averaging as a logically correct operation.

To broaden the scope of our investigation we also plan to explore both further aggregation mechanisms, such as the MEDRANK approach of Fagin et al. (2003); and also other rank correlation approaches, including the τAP top-weighted approach of Yilmaz et al. (2008). Top-weighting of the rank correlation scoring mechanism is important if we are more interested in fidelity near the top of each ranking than in (say) the bottom half of the two system orderings being compared.

We conclude by reiterating that our experiments – in which we regard a metric and aggregation technique to be "good" if they yield similar overall system rankings from different topic sets – are founded in practice rather than in theory, and reflect common evaluation custom rather than an underlying principle. We also stress that to be plausible, experimentation should be accompanied by significance testing, and because significance tests are carried out over sets of values, the details of the aggregation technique used to obtain representative gross scores are of somewhat parenthetical interest if significance cannot be asserted. Nevertheless, and even given these caveats, we have found that ϵGM has a role to play when there are many small score values to be handled, and that AM-SP is the combined metric and aggregation technique that is most strongly self-consistent in terms of score-induced system rankings.

Acknowledgment: This work was supported by the Australian Research Council, by the Government of Malaysia, and by the University of Malaya. We thank the referees for their helpful comments.

References

Al-Maskari, A., Sanderson, M., Clough, P. & Airio, E. (2008), The good and the bad system: Does the test collection predict users' effectiveness?, *in* 'Proc. 31st Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval', Singapore, pp. 59–66.

Buckley, C. (2004*a*), Topic prediction based on comparative retrieval rankings, *in* 'Proc. 27th Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval', Sheffield, England, pp. 506–507.

Buckley, C. (2004*b*), Why current IR engines fail, *in* 'Proc. 27th Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval', Sheffield, England, pp. 584–585.

Cormack, G. V. & Lynam, T. R. (2007), Validity and power of t-test for comparing MAP and GMAP, *in* 'Proc. 30th Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval', Amsterdam, The Netherlands, pp. 753–754.

Fagin, R., Kumar, R. & Sivakumar, D. (2003), Efficient similarity search and classification via rank aggregation, *in* 'Proc. SIGMOD Int. Conf. on Management of Data', San Diego, CA, pp. 301–312.

Järvelin, K. & Kekäläinen, J. (2002), 'Cumulated gain-based evaluation of IR techniques', *ACM Transactions on Information Systems* **20**(4), 422–446.

Kendall, M. & Gibbons, J. D. (1990), *Rank Correlation Methods*, Oxford University Press, New York.

Mandl, T., Womser-Hacker, C., Nunzio, G. D. & Ferro, N. (2008), How robust are multilingual information retrieval systems?, *in* 'Proc. 23rd Ann. ACM Symp. on Applied Computing', Fortaleza, Cear, Brazil, pp. 16–20.

Mizzaro, S. (2008), The good, the bad, the difficult, and the easy: Something wrong with information retrieval evaluation?, *in* 'Proc. 30th European Conf. on Information Retrieval', Glasgow, Scotland, pp. 642–646.

Mizzaro, S. & Robertson, S. (2007), HITS hits TREC: Exploring IR evaluation results with network analysis, *in* 'Proc. 30th Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval', Amsterdam, The Netherlands, pp. 479–486.

Moffat, A. & Zobel, J. (2009), 'Rank-biased precision for measurement of retrieval effectiveness', *ACM Transactions on Information Systems*. To appear.

O'Brien, M. & Keane, M. T. (2007), Modeling user behavior using a search-engine, *in* 'Proc. 12th Int. Conf. on Intelligent User Interfaces', Honolulu, Hawaii, USA, pp. 357–360.

Robertson, S. (2006), On GMAP: And other transformations, *in* 'Proc. 15th ACM Int. Conf. on Information and Knowledge Management', Virginia, USA, pp. 78–83.

Robertson, S. (2008), A new interpretation of average precision, *in* 'Proc. 31st Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval', Singapore, pp. 689–690.

Sanderson, M. & Zobel, J. (2005), Information retrieval system evaluation: effort, sensitivity, and reliability, *in* 'Proc. 28th Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval', Salvador, Brazil, pp. 162–169.

Voorhees, E. M. (2003), Overview of the TREC 2003 Robust Retrieval Track, *in* 'Proc. 12th Text REtrieval Conference (TREC 2003)', Gaithersburg, Maryland.

Voorhees, E. M. (2005), Overview of the TREC 2005 Robust Retrieval Track, *in* 'Proc. 14th Text REtrieval Conference (TREC 2005)', Gaithersburg, Maryland.

Webber, W., Moffat, A. & Zobel, J. (2008a), Score standardization for inter-collection comparison of retrieval, *in* 'Proc. 31st Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval', Singapore, pp. 51–58.

Webber, W., Moffat, A., Zobel, J. & Sakai, T. (2008b), Precision-at-ten considered redundant, *in* 'Proc. 31st Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval', Singapore, pp. 695–696.

Yilmaz, E., Aslam, J. A. & Robertson, S. (2008), A new rank correlation coefficient for information retrieval, *in* 'Proc. 31st Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval', Singapore, pp. 587–594.

Zobel, J. (1998), How reliable are the results of large-scale information retrieval experiments?, *in* 'Proc. 21st Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval', Melbourne, Australia, pp. 307–314.

S.E.A.L. – A Query Language for Entity-Association Queries

Edward Stanley*, Pavle Mogin†, Peter Andreae†

* Kakapo Technologies
32a Salamanca Road, Wellington 6140, New Zealand
† School of Mathematics, Statistics, and Computer Science
Victoria University of Wellington
PO Box 600, Wellington 6140, New Zealand

eddiewould@paradise.net.nz, Pavle.Mogin/Peter.Andreae@mcs.vuw.ac.nz

Abstract

The paper presents the S.E.A.L. query language and interpreter for entity-association queries that allows such queries to be expressed in a much simpler way than in SQL. S.E.A.L (Simplified Entity Association Language) also supports Entity-Attribute-Value (EAV) data structures, enabling users to write queries without having to know whether a particular attribute is a regular attribute or an EAV attribute. The language allows parts of a query to be omitted if they are implied by the rest of the query, and the interpreter will infer the missing requirements, or ask the user to resolve the ambiguity. S.E.A.L. makes it easier for users of databases, particularly science and eCommerce databases, to make use of the valuable information in their databases.

Keywords: Databases, Query Language, EAV, Query Inference

1 Introduction

Many database users in fields such as Medicine, Biology, and Genetics have large collections of data which contain important information. This information would be valuable to their owners, if only it could be extracted from the data. A similar situation exists in the case of many eCommerce databases. However, many users of these databases have only modest database expertise and use very simple SQL or query-by-example (QBE) tools such as Microsoft Access. But QBE tools are limited in the kinds of queries they support, and therefore users are unable to exploit all the information hidden in their data. This paper presents S.E.A.L. (the Simplified Entity Association Language) – an extension to SQL to enable such users to extract information more easily.

A common class of queries involves finding all instances of an entity-type which satisfy some constraints involving participation in relationships with other entities. An example of this kind of query would be finding all the patients in a medical database with a set of heart related symptoms who were closely related to someone who had been diagnosed with a particular disease. These queries, called entity-association queries, are common but are difficult to express in SQL. The difficulty arises because users must know the implementation schema in detail. Particularly, users need to know table names, the location of attributes, and table primary and foreign keys; simply knowing the conceptual schema (problem domain) is insufficient. Additionally, users need advanced specialist knowledge of SQL in order to declare explicit joins and constraints on entity properties in terms of conditional expressions.

Entity-Association queries become even more complex if a query contains a conjunctive condition on two different values of the same attribute, or if some of the query attributes are stored in Entity-Attribute-Value (EAV) database structures. In both cases, the number of nesting levels may increase, and nesting makes queries difficult to construct and harder to debug. (Such queries may prove to be complex even for database professionals.)

If an entity-association query contains a conjunctive constraint on two different values of the same attribute, e.g. “retrieve students who passed both Database Systems and Operating Systems courses”, expressing it in SQL requires finding the intersection of entity sets satisfying each of the constraints alone. Such queries are nested by default. Increasing the number of entity and relationship types involved in the query may also increase the number of nesting levels in the SQL statement.

Entity-Attribute-Value database structures are used for efficiently storing sparse data and for allowing user-defined attributes. There are a number of variants of EAV database structures, but they all share a common characteristic: meta-data regarding attribute names is stored in database tables like other common data. Syntactically, EAV structures can be considered as entity-association structures, and therefore even conceptually simple queries involving EAV database structures share all the complexity of other entity-association structures, requiring detailed knowledge of the database schema, and often requiring complex nested query structures. The need for EAV structures is very common in Medicine, Biology, Genetics, Chemistry, and generic e-Commerce web database applications. For example, in the medical database above, most patient symptoms would need to be stored in an EAV structure because there would be too many sparsely populated possible symptoms to use regular attributes.

As a solution to these problems, S.E.A.L. offers a highly declarative syntax for expressing entity-association queries in a natural way that is closer to the conceptual than to the implementation level of the database abstraction. Additionally, S.E.A.L. possesses inference mechanisms that allow information about some of the database structural concepts to be omitted from the query. S.E.A.L.'s inference mechanism relies on a disciplined database schema approach. Thanks to this design approach, S.E.A.L. accepts queries having only one entity type name, a list of result attributes, and conditional expressions on attribute values, and produces a corresponding SQL expression or warns the user that the query specification is insufficient for an unambiguous translation of the query. In the course of the query translation, S.E.A.L. infers all necessary associated entity types, relationship types, and entity type roles, and generates a SQL query in terms of the underlying relational implementation structure.

The paper describes:

- A set of rules and guidelines for designing databases compatible with S.E.A.L.,
- The S.E.A.L. declarative language for specifying entity-association queries, which also supports EAV structures,
- A prototype interpreter for S.E.A.L. queries.

Section two of the paper reviews related work. Sections three and four introduce an example database schema, and briefly discuss EAV database structures. Section five analyses a simple entity-association query and justifies the claim that such queries are complex to define. Sections six and seven introduce the S.E.A.L. syntax and briefly describe the design of the S.E.A.L. interpreter. Section eight reports on the results of a limited number of performance measurements and the final section presents conclusions and ideas for future work.

2 Related Work

Because of its inference abilities, S.E.A.L belongs to the class of the database query languages with a Universal Relation Schema Interface. According to the Universal Relation Assumption (Ullman 1982), there exists a hypothetical relation schema (URS) which contains all attributes of a universe of discourse (problem domain). An actual database schema is produced by decomposing the URS. A URS query interface allows a user to define queries solely on attributes without having to care about real database objects like tables.

One of the first research projects on a query language with URS user interface was developed within an experimental database management system called System/U (Ullman 1982). Ullman describes a query interpretation algorithm and two ways to cope with ambiguities induced by database cyclic structures, but does not discuss interpretation of queries having conjunctive conditional expressions on different values of an attribute.

The Query-By-Example (QBE) language is a graphical query language developed by IBM Research and is available as a part of the Query Management Facility (Elmasri and Navathe 2006). It is also embedded into Microsoft Access. A query is formulated in QBE by filling in table templates that are displayed on the screen. Users drag and drop tables and set predicate conditions to construct a query. The QBE engine takes care of issues such as aliasing and generating join conditions. QBE relieves a user from having to know the structure of the underlying database, but it is still not a URS interface language, since it builds queries using tables. Also, QBE engines that we have tested have been unable to produce queries involving conjunctive conditional expressions on different values of an attribute (except by modifying the database structure).

Entity Attribute Value (EAV) database structures are important for this paper since queries against EAV database structures belong to the class of entity-association queries. EAV structures are described in Nadkarni and Brandt (1998), Dinu and Nadkarni (2006), and Corwin *et al* (2007).

We identified two systems which abstract EAV attribute representation from a database user. The first, ACT/DB (Nadkarni *et. al*. 1998), is a database tool for managing clinical trial data. It uses a client/server architecture with Oracle7 at the backend. Users construct queries with a GUI-based tool written in Microsoft Access. The tool uses Visual Basic code to handle the abstraction of EAV attributes in queries and translation into SQL to be executed at the backend. The tool relies upon the specific schema for which it was designed. The schema includes conventional tables as well as six general purpose EAV tables for the various data types supported. ACT/DB supports a number of comparison operators as well as aggregate functions such as average and standard deviation.

The second is QAV (Nadkarni 1996) which is a GUI-based tool which allows users to perform queries against the Columbia MED dataset, a large medical metadata repository. QAV uses a special schema in which all data is represented in EAV form. QAV is also based on client-server architecture.

Both of these systems are tied to a particular schema. A significant advantage of S.E.A.L is that it can be used with any schema which conforms to the rules and guidelines given in section 6.3 of this paper.

During the literature search, we found no previous attempts to classify entity-association queries or implement a general solution for allowing users to express these queries in an easier way.

3 Example Database Schema

To illustrate entity-association queries in a way that requires no specialized domain knowledge and to explain the proposed language and interpreter, the paper uses a running example of a fictional '*robbers*' database. The '*robbers*' database contains a variety of structures that require defining entity-association queries.

The *robbers* database describes robbers and banks they have robbed. Robbers have certain skills that are tested at special testing locations and some robbers may have special features such as haircuts, or like particular kinds of music. Figure 1 contains the conceptual '*robbers*' database schema in the form of an entity-relationship diagram, which, for simplicity, is presented with no attributes.

Figure 2 contains a relational schema that corresponds to the ER diagram in Figure 1, giving the schema name and the set of attributes, with the primary key underlined.

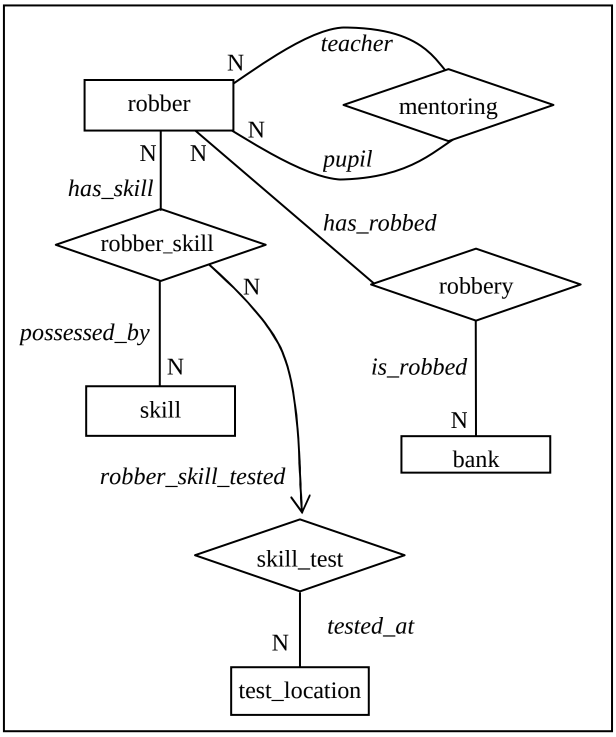


Figure 1: ER diagram of the *Robbers* database.

Relation Schemes:

robber{*robberid*, *nickname*, *age*},
robber_attributes{*attributeid*, *attribute*},
robber_eav{*robberid*, *attributeid*, *value*},
mentoring{*robberid1*, *robberid2*}
robber_skill{*robberid*, *skillid*, *skilllevel*},
skill{*skillid*, *skillname*},
skill_test{*robberid*, *skillid*, *locationid* },
test_location{*locationid*, *locationname*},
bank{*bankid*, *bankname*},
robbery{*robberid*, *bankid*, *date*, *amount*}

Referential Integrity Constraints:

robber_eav[*robberid*] ⊆
robber[*robberid*] (*eav_robber*),

robber_eav[*attributeid*] ⊆
robber_attributes [*attributeid*] (*eav_skill*),

mentoring[*robberid1*] ⊆ *robber* [*robberid*] (*teacher*),

mentoring[*robberid2*] ⊆ *robber* [*robberid*] (*pupil*),

robber_skill[*robberid*] ⊆
robber [*robberid*] (*has_skill*),

robber_skill[*skillid*] ⊆ *skill* [*skillid*] (*possessed_by*),

skill_test[(*robberid*, *skillid*)] ⊆
robber_skill[(*robberid*, *skillid*)] (*rob_skill_tested*),

skill_test[*locationid*] ⊆
test_location[*locationid*] (*tested_at*),

robbery[*robberid*] ⊆ *robber* [*robberid*] (*has_robbed*),

robbery[*bankid*] ⊆ *bank* [*bankid*] (*is_robbed*)

Figure 2: Implementation Schema

For simplicity, all relation schemes have only one key — the primary key — and a small number of attributes

The referential integrity constraints have the form $N_1[FK\ ] \subseteq N_2[PK\ ]$ (*role*), where *FK* is the foreign key corresponding to the primary key *PK* of N_2, and *role* is the role of the entity type N_2 in the relationship type N_1.

Sparse robber attributes like *haircut*, *music*, and *call-sign* are stored in EAV database structures, represented by *robber_attributes* and *robber_eav* relation schemes. EAV database structures are described in the next section.

4 EAV Database Structures

Under a conventional relational database design, each entity instance is represented by a single row in the table representing that entity type. This row has a fixed number of columns and each column stores an attribute value. Each attribute is described by the name of the column and the data type. As a consequence of this, all entity instances of the same entity-type contain values of the same set of attributes. Metadata on the number, names and types of attributes an entity-type stores information on is defined by the table structure.

An Entity Attribute Value (EAV) database structure is an alternative method for representing the attributes belonging to an entity type. EAV represents each entity instance as a set of (*entityid*, *attribute*, *value*) triples (Nadkarni and Brandt 1998). The *entity* describes the entity, the *attribute* value carries the information about the attribute name, while the *value* assigns data to the attribute name. Under EAV, metadata is represented as data: not only are the values of attributes data (as they were in the conventional relational model) but the attribute names are also data.

EAV tables are generally not shared between entity types: that is, each entity type making use of EAV storage is assigned a separate EAV table. A useful variation to the described structure is to also have a separate lookup table to record the names of attributes stored in EAV form. Often the attribute lookup table will have a surrogate integer primary key (*attributeid*) and the EAV table stores a reference to this field.

EAV database structures are used in applications that handle sparse data, or have a need for user defined attributes. They can also be used to store multivalued attributes.

4.1 Sparse Attributes

Conventional "one fact per column" table designs are unsuitable for extremely sparse data. A common example of sparse data is a patient record in a medical database. While there may be thousands of possible facts that can apply to a single patient record, the number that typically applies may be only a few dozen (Nadkarni and Brandt 1998). Unlike conventional relational tables which set aside space for each attribute whether it is null or not, EAV only represents facts which apply to the given entity instance. EAV can be used in combination with conventional storage: data which applies to every instance can be stored in a conventional table and the sparse data can be represented in an EAV table (Dinu and Nadkarni 2006).

4.2 User Defined Attributes

In certain applications, as in generic e-commerce databases, there is a need for users to be able to define (and destroy) attributes as part of a normal usage. Under a conventional schema this would involve acquiring an exclusive lock on the table and then issuing data definition language (DDL) instructions to alter the table structure. In a busy high-volume database, it may not be possible to acquire an exclusive lock or it may be too detrimental to performance. Allowing the user/application to perform DDL is also a significant security risk.

Using an EAV design, the attributes for an entity-type are described as regular data, and therefore only regular row INSERT/DELETE operations are required to add and remove attributes respectively. This design removes the security risk associated with user-defined attributes and also removes the need to acquire an exclusive lock on the whole table, making user-defined attributes feasible even in a high-volume database.

4.3 Multivalued Attributes

In the conventional normalized relational databases, multivalued attributes are stored using separate tables. For example, the multivalued attribute robber's *skills* is represented this way in the schema of Figure 2. An EAV database structure is another possible solution for representing multi-valued attributes, if the primary key of the EAV table is set to be (entityid +attributeid +value).

Example 1. In the example *robbers* database, dense robber attributes *nickname* and *age* are stored in the conventional table *robber*, and sparse attributes like *haircut*, *music*, and *call-sign* are stored in EAV database structures. Figure 3 presents instances of the *robber_attributes* and *robber_eav* EAV relation schemes. The *robber_attributes* instance lists the sparse attributes *haircut*, *music*, and *callsign*, and the *robber_eav* instance specifies the association between robbers and attribute values.

robber

robberid	*nickname*	*Age*
1	*Al Capone*	*31*
2	*Bugsy Malone*	*23*
3	*Lucky Luchiano*	*55*
4	*Anastazia*	*47*
5	*Dutch Schulz*	*63*

robber_attributes

attributeid	*attribute*
1	*haircut*
2	*music*
3	*callsign*

robber_eav

robberid	*attributeid*	*value*
1	*1*	*Mohawk*
4	*1*	*Mohawk*
4	*2*	*Latin*
4	*2*	*Classic*

Figure 3

Using the combination of the following two SQL statements, a user may insert a new sparse attribute *food* and assign a value '*Pizza*' to the *food* attribute of a robber:

```
INSERT INTO robber_attributes VALUES
(4, 'food');
INSERT INTO robber_eav VALUES (2, 4,
'Pizza');
```

The structure composed of tables *has_skill* and *skill* is another EAV structure that is used to represent the multivalued attribute *skill* in the *robbers* database. Here, the attribute *skilllevel* in the *robber_skill* table associates values of the attribute *skill* to robbers.

A significant problem with EAV structures is that the user has to know whether an attribute is stored as a regular column or in an EAV table (Nadkarni and Brandt 1998). This problem adds to the complexity of writing SQL queries against EAV database structures. As it will be shown in the following sections, thanks to its inference capabilities S.E.A.L. hides this complexity from users.

5 Analysis of an Entity Association Query

If an entity association query contains two or more conjunctively bound constraints on the same attribute, SQL does not have a declarative syntax for expressing this class of queries in a simple way. In principle, such queries require finding the intersection or set difference of entity sets satisfying each of the constraints alone. We are aware of four main strategies for expressing such entity-association queries in SQL. These are:

1. Multiple `SELECT` statements combined with the set operators,
2. Nested `SELECT` queries using the `IN` operator,
3. Correlated nested `SELECT` queries using `EXISTS` operators, and
4. Multiple nested `SELECT` queries where the intersection is performed using an equi-join.

Example 2. The following is a relatively simple entity-association query for the *robbers* database:

> "Find the robbers who have the skill '*Gun Shooting*' but do not have the skill '*Money Counting*', and the robbers who have the skill '*Explosives*'".

Using the first approach, this query could be broken down into three SELECT queries and then combined with the set operators UNION, INTERSECT and EXCEPT:

```
SELECT nickname FROM robber NATURAL
JOIN
((SELECT robberid FROM robber_ skill
  NATURAL JOIN skill WHERE
  skillname='Gun Shooting'
 EXCEPT
  SELECT robberid FROM robber_skill
  NATURAL JOIN skill WHERE
  skillname='Money Counting')
 UNION
 SELECT robberid FROM robber_skill
  NATURAL JOIN skill WHERE
  skillname='Explosives') AS foo;
```

Using the second approach, the query could be expressed using nested IN statements:

```
SELECT nickname FROM robber WHERE
((robberid IN (SELECT robberid FROM
  robber_skill NATURAL JOIN skill
  WHERE skillname='Gun Shooting'))
 AND
 (NOT (robberid IN (SELECT robberid
  FROM robber_skill NATURAL JOIN
  skill WHERE skillname='Money
  Counting'))))
OR
(robberid IN (SELECT robberid FROM
 robber_skill NATURAL JOIN skill
  WHERE skillname = 'Explosives'));
```

Using the third and fourth strategies results in similarly complex nested SQL expressions.

We argue however that a query constructed using either of these strategies poorly reflects the logical intent of the query and because of this is difficult to write. Given the intent of the query, it will not be clear to a user why they would need to issue multiple select statements. From the user's perspective, the relationships in which an entity participates can be viewed as a property of the entity itself. The following relation schema probably better represents the user's understanding of the domain:

```
robber{robberid, nickname,
       has_gun_shooting_skill,
       has_money_counting_skill,
       has_explosives_skill}
```

Example 3. With this schema, the query could be expressed straightforwardly as:

```
SELECT nickname FROM robber WHERE
 (has_gun_shooting_skill AND NOT
 has_money_counting_skill )
 OR has_explosives_skill;
```

While this database design is flawed (it will not scale to hundreds of possible skills), it is easy to see that this SQL reflects the user's intention more simply and more directly than the complex queries above.

Furthermore, all of the strategies described lead to queries which are unnecessarily difficult to write. Factors contributing to this difficulty include the following:

1. Users need to know where an attribute is stored. Is it stored in a regular column or as an EAV attribute?
2. Users need to construct join conditions explicitly. For this trivial example, `NATURAL JOIN` suffices but, in the case of joins involving a recursive relationship type, INNER JOIN has to be used.
3. When using INNER JOIN, a user needs to know the primary keys of tables representing the entities involved in the relationship as well as the primary and foreign keys of the tables which represent the relationship. In more complex examples, these keys may be composite, involving three or more attributes.
4. With nested associations, multiple table aliases must be used. These aliases must be uniquely named and knowing which alias to refer to is a source of confusion.
5. Finally, if the query contains a conjunctive constraint on the same attribute, the query has to find the intersection of entity sets satisfying each of the constraints alone. Such queries are complex: hard to define and even harder to debug.

This analysis applies equally to queries on EAV structures, since the *entity_eav* relation schema corresponds to a relationship type and the *entity_attributes* relation schema corresponds to an entity type.

Example 4. Consider the following query against the *robbers* database:

"Find robbers who like '*Latin*' and '*Classic*' music."

The query constraints ask for two values of the multivalued sparse attribute *music,* which is stored in the EAV structure. The following SQL query uses the `SELECT … IN …` approach:

```
SELECT nickname FROM robber WHERE
  (robberid IN (SELECT robberid FROM
robber_eav NATURAL JOIN
   robber_attributes WHERE attribute
= 'music' AND value = 'Latin'))
  AND
  (robberid IN (SELECT robberid FROM
robber_eav NATURAL JOIN
   robber_attributes WHERE attribute
= 'music' AND value = 'Classic'));
```

6 S.E.A.L. Design

This section describes the design of the S.E.A.L. system, including the syntax of the S.E.A.L. query language, the requirements on the design of the database, and the S.E.A.L. interpreter, particularly the inference algorithms S.E.A.L. uses to infer entity and relationship types left implicit in a query.

6.1 S.E.A.L. Syntax

The syntax of the S.E.A.L. query language is inspired by the SQL SELECT syntax, but is specialised for entity-association queries. A S.E.A.L. query specifies a set of attributes and a base entity type, along with two kinds of constraints: constraints on the attributes of the base entity (regular or EAV attributes), and constraints on entities associated with the base entity. A S.E.A.L. query has the following form:

```
Query ::=
   SELECT
      attribute [,...] | *
   FROM
      baseEntityType
      [ '['
       AttributeConstraintExpression
       ']' ]
      [ AssociationExpression ]
```

An `AttributeConstraintExpression` is a logical expression (which may contain conjunction, disjunction, negation, and parentheses) involving `AttributeConstraint`(s) which specify constraints on regular or EAV attributes of the base entity type:

```
AttributeConstraint ::=
   attribute ( '=' | '!=' | '<' | '>' |
   '>=' | '<=' ) value
```

An `AssociationExpression` is a logical expression involving `Association`(s), each of which specifies an associated entity type, the relationship type by which the entity type is associated with the base entity type, the roles in the relationship type of the base entity type (“`VIA`”) and associated entity type (“`AS`”), followed by constraints on the relationship and associated entity types. The entity type, relationship type and roles are all optional if they can be unambiguously inferred from the rest of the query.

```
Association ::=
 ‘ASSOCIATED_WITH’ ‘(’
  [ ‘VIA’ EntityRole ]
  [ AssociatedEntityType
   [ ‘AS’ AssociatedEntityRole ]
   [ ‘THROUGH’ AssociatingRelationship ]
  ‘,’ ]
  AssociationConstraintExpression
 ‘)’
```

An `AssociationConstraintExpression` is a logical expression involving `Association Constraints`, each of which specifies constraints on the associating relationship type and/or the associated entity type, possibly including nested `Association Expressions` to constrain the entity or the relationship type.

```
AssociationConstraint ::=
   ‘<’ AttributeConstraint [, . . .] ‘>’
  [AssociationExpression ]
```

The attribute constraints in an association constraint can refer to any attribute (regular or EAV) of either the associating relationship type or the associated entity type of the `Association`.

6.2 Example Queries in S.E.A.L. Syntax

The following examples use the *robbers* schema given in Section 3. All examples follow the same structure: the aim of the example is given first, it is followed by the query, then the S.E.A.L. syntax, and finally by an optional comment.

Example 5. The following demonstrates uniform access to attributes (whether EAV or regular ones):

“Find robbers who are *39* years old and have a ‘*Mohawk*’ haircut.”

```
SELECT nickname FROM robber [haircut
= 'Mohawk' AND age = 39]
```

The user does not need to know that *haircut* is represented in EAV form, while *age* is a regular column.

Example 6. The following demonstrates how a simple entity-association query is composed in S.E.A.L.:

“Find robbers who have the skill ‘*Lock-Picking*’ and who have the skill ‘*Planning*’.”

```
SELECT nickname
FROM robber ASSOCIATED_WITH(
 skill THROUGH robber_skill,
 <skillname = 'Lock-picking'> AND
 <skillname = 'Planning'>)
```

The base entity-type is restricted based on the participation constraint specified in the `ASSOCIATED_WITH` clause. The associated entity-type, *skill*, is specified explicitly as is the relationship type *robber_skill*. The specification of roles is not needed, since the relationship type *robber_skill* associates only the *robber* and *skill* entity types. Two specifications of attribute constraints are used in the query which are combined into an expression with the AND operator.

Example 7. The following demonstrates the role of inference in S.E.A.L.:

“Find robbers who robbed the ‘*Loan Shark*’ bank.”

```
SELECT nickname FROM robber
 ASSOCIATED_WITH(<bankname =
 'Loan Shark'>)
```

In this query, the associated entity type, the relationship type, and the roles of the both entity types were all omitted. With regard to the *robbers* schema, given in Section 3, S.E.A.L. is able to infer that the specification `<bankname = 'Loan Shark'>` refers to the associated entity type *bank* through the relationship type *robbery*. The corresponding roles are also inferred.

Example 8. The following shows how an entity-association query, where the relationship type has a participation constraint, is constructed in S.E.A.L.:

“Find robbers with the skill ‘*Guarding*’ who had it tested at a testing location called ‘*Harvard*’.”

```
SELECT nickname FROM robber
 ASSOCIATED_WITH(skill,
  <skillname = 'Guarding'>
  ASSOCIATED_WITH(test_location,
   <locationname = 'Harvard'>))
```

S.E.A.L. interprets the query through the following two main steps. First, it infers the association between *robber* and *skill* tables through the *has_skill* table using the attribute *skillname*. Next, it infers the association between *has_skill* and *test_location* tables through the *skill_test* table using the attribute *locationname*. As a result, S.E.A.L. constructs a SQL statement with two levels of nesting.

Example 9. The final example is a query where roles cannot be inferred and must be specified:

“Find robbers who have been taught by ‘*Bugsy Malone*’.“

```
SELECT nickname FROM robber
 ASSOCIATED_WITH(robber
 AS teacher THROUGH mentoring,
 <nickname= 'Bugsy Malone'>)
```

In this example, the role of the associated entity type (also a *robber*) was specified as a *teacher*. It was necessary to specify at least one role, because the *mentoring* relationship involves two entities of the same type (a *robber* acting as a *teacher* and a *robber* acting as a *pupil*).

6.3 Rules and Guidelines for Implementing S.E.A.L. Compatible Databases

The S.E.A.L. interpreter has a set of rules and conventions to which a schema must adhere if it is to be used with S.E.A.L. These rules and conventions provide a consistent way to describe metadata as well as simplifying the interpreter code by reducing the number of special cases needed. These rules follow a common disciplined database design approach and are illustrated by the *robbers* schema in Figure 2. The most significant guidelines are the following:

- All attributes that have different meaning (whether stored conventionally or in EAV form) must have distinct names.
- EAV database structures should be implemented using an attribute lookup table, named ⟨*entity*⟩*_attributes* and a relationship table named ⟨*entity*⟩*_eav* with the structure (*entityid*, *attributeid*, *value*), where ⟨*entity*⟩ is the name of the entity type the attributes belong to.
- The referential integrity (foreign key) constraints have to be named after the name of the role the referenced table plays in the relationship type.

These, and the other requirements, all represent good database design practices, and are not difficult to satisfy. We note that the first requirement is needed to avoid ambiguities induced by cyclic database structures (Ullman 1982).

6.4 Inference Algorithms

S.E.A.L. gives users considerable flexibility when specifying a participation constraint via an `ASSOCIATED_WITH` clause. While the base entity-type and an expression constraining attribute values are required, the associated entity type, the relationship type, and the roles of both the base and associated entity types are optional.

If any of these types or roles is omitted, S.E.A.L. will attempt to infer them based on the attributes used in the specification expression. To be able to perform translation without ambiguity, S.E.A.L. needs to find exactly one valid (*relationship type*, *associated entity type*, *base entity role*, *associated entity role*) combination for the base entity-type and attributes used.

6.4.1 Inferring possible combinations

The first step is to infer all the possible combinations of relationship types, associated entity types, and roles that are consistent with the specified base entity type. Any types or roles specified in the query constrain the search. Subject to these constraints, S.E.A.L. searches the metadata for all tables in the database that have at least two foreign keys, at least one of which references the base entity type. Any such table is a candidate relationship type, and all tables referred to by the other foreign keys of the table are candidate associated entity types. Names of the foreign keys involved are the roles.

S.E.A.L. then attempts to reduce the set of candidate combinations by examining the attributes specified in the query (using the attribute coverage algorithm described in 6.4.2) and eliminating combinations that are not consistent with the attributes. If the set of combinations is not reduced to a single combination, then the query contains ambiguity which the user must resolve.

One kind of ambiguity arises if there are two possible relationships in the database between the base entity type and an associated entity with the specified attributes. In this case, S.E.A.L. can report the alternative relationships to the user so that the user can specify the relationship they intended.

Another kind of ambiguity arises if the relationship type of a combination involves the base or associated entity type in more than one possible role (*i.e.*, the table has two or more foreign keys referring to the same entity type) and the role is not specified in the query. In this case, the algorithm reports the ambiguity to the user, along with the possible roles, so that the user can refine the query.

Once there is an unambiguous consistent combination of ⟨*relationship type*, *associated entity type*, *base entity role*, *associated entity role*⟩, this information is passed to the translation algorithms (6.5) to construct the SQL query.

Note that a S.E.A.L query may contain multiple `ASSOCIATED_WITH` clauses and that the inference process must be applied to each clause. For nested `ASSOCIATED_WITH` clauses, the process must be applied recursively.

6.4.2 Attribute coverage algorithm

For a given (*relationship type*, *associated entity type*, *base entity role*, *associated entity role*) combination to be valid with respect to a query, the associated entity type together with the relationship type must contain (in either EAV or regular column form) all attributes referred to in an `ASSOCIATED_WITH` clause. A combination with this property is said to "cover" the `ASSOCIATED_WITH` clause. The attribute coverage algorithm checks a (*relationship type*, *associated entity type*) pair to determine if it covers the query.

Regular attributes can be checked from the metadata, but checking for the existence of an EAV attribute requires a query against the database. For efficiency, attributes are grouped into sets corresponding to their expected location and all EAV attributes used in a query are checked at once rather than individually.

6.5 Translation Algorithms

Once a query has been parsed and all necessary inferences are completed, the translation into SQL is performed by a bottom-up, recursive traversal of the query tree. In this design, the various parts of the query tree are responsible for their own conversion into SQL with relatively little interdependence.

At an abstract level, the translation of a S.E.A.L. statement produces SQL blocks of the form

```
base_entityid IN (SELECT
base_entityid FROM relationship INNER
JOIN associated_entity ON … WHERE
AssociationConstraint)
```

for each Association Constraint in an `ASSOCIATED_WITH` clause. These SQL blocks are connected by logical operators in accordance with the structure of the

Association Constraint Expression and placed in a SQL `WHERE` clause in the following way

```
SELECT attribute_list FROM
base_entity_type WHERE
(SQL_block_expr);
```

The form of the generated SQL statement has a form very close to the second SQL query in Example 2.

7 Implementation of the S.E.A.L. Interpreter

When the S.E.A.L. interpreter first starts, it connects to the PostgreSQL database and collects metadata about the database that it needs for translating queries. When a user issues a query, the interpreter parses and translates the query. If it identifies any ambiguity or errors in the query, the interpreter not only reports the problems, but also gives suggestions to the user on how to resolve them. Once the query is successfully translated, it outputs the SQL, which can then be run against the database.

The SEAL prototype interpreter is implemented in a Java, JDBC, and PostgreSQL environment. After retrieving metadata, translating a query comprises three main stages: parsing, inference, and translation.

Since the principles of the inference and translation process were described in section 6.4, this section briefly describes only the metadata retrieval and parsing.

7.1 Retrieval of metadata

PostgreSQL provides an 'Information Schema' that is a collection of views and tables describing structures of databases contained in a cluster. These tables and views can be queried through regular SQL.

S.E.A.L connects to PostgreSQL through JDBC and issues custom queries against the PostgreSQL Information Schema to collect metadata on attribute names, table names, table primary keys, table foreign keys, and foreign key names. Metadata, which carries information about role names, is stored in an internal data structure.

EAV attributes are not collected during the metadata acquisition phase, since an entity type may have many thousands of EAV attributes, and many are most probably not needed by a query. Any EAV attributes used in a query are checked at the translation time (which also allows for updates to the EAV attributes during the interaction with the interpreter).

7.2 Parsing

The parser for the interpreter was created with the ANTLR (ANTLR http://www.antlr.org) parser generator. ANTLR allows the programmer to write a context free grammar along with Java code to describe actions to perform on rule/token matches. In the case of the S.E.A.L interpreter, the rules defined are relatively simple: the parser simply generates an OO representation of the query tree substituting base and associated entity-types, relationship types and roles with table, join-table and foreign key names, respectively, contained in the metadata structure.

8 Testing

We performed three kinds of testing. One was testing the claim that entity-association queries are complex. The other was comprised of an extensive sequence of functional tests on the S.E.A.L prototype interpreter. A limited number of performance tests was our third kind of tests. The section briefly describes the tests.

8.1 Testing complexity of entity-association queries

To prove the claim that entity-association queries with a conjunctive condition on two different values of the same attribute are hard to define, we asked a group of 47 students to produce SQL statements for the following entity-association queries within given time limits:

1 Find robbers who have the '*Planning*' skill. [5 minutes],

2 Find robbers who have been taught by '*Bugsy Malone*'. [6 minutes], and

3 Find robbers who have the '*Planning*' and '*Lock-Picking*' skills. [10 minutes].

The students had just completed an intensive training in SQL and were familiar with the structure of the *robbers* database (which excludes any effect of surprise on their performance). The aim of the first two queries was to test students' SQL expertise with simple entity-association queries, while the third query was an entity-association query with a conjunctive condition on two values of an attribute. The outcomes of the test are given in the table below.

Query	Correct answers [%]
1	92
2	56
3	32

The fact that 92% and 56% of students defined correctly the first two queries within the given time limits shows that they possessed a fair level of SQL expertise. The fact that almost 70% of them failed to give a correct answer to the third question supports the claim that entity-association queries with a conjunctive condition on two values of an attribute are hard to define.

8.2 Functional testing

The goal of the functional testing was to check that S.E.A.L meets the expected functional requirements. These tests comprised the following steps:

- Defining a query against the *robbers* database in English,
- Expressing the query in S.E.A.L syntax,
- Expressing the query in SQL using an expert's knowledge,
- Running the S.E.A.L interpreter to produce a SQL query,
- Comparing the S.E.A.L generated SQL query and the SQL query produced by the expert, and
- Running both SQL queries against the *robbers* database and comparing results.

The examples listed in previous sections of the paper are a representative set of the test queries that were run. The actual testing comprised numerous variants of these queries and a number of queries that went beyond the initial requirements of the S.E.A.L. project. The query variants included more complex conditional expressions on attributes, multiple associations of the base entity type with associated entity types, multiple associations of relationship types, and conditional expressions on associated entity type EAV attributes. The SEAL prototype interpreter passed all these tests successfully.

The queries that went beyond the project requirements identified some limitations of the current implementation and are discussed in the Conclusion as a part of the future work.

8.3 Performance testing

There were two parts to the performance testing: comparing the performance of Nested-In and Set theoretic operator approaches to produce entity-association queries and comparing the interpreter overhead to the query execution time.

All performance tests were performed on an Intel Pentium 4 processor at 3.2 GHz, with1.5 GB DDR2 memory at 533 MHz , and a Seagate 80Gb SATA disk, using Net-BSD 4.99.9, and PostgreSQL Version 8.2.4.

Two databases were used: the *robbers* database and a EAV database consisting of a table representing an entity type and two other tables representing its EAV data structure. The *robbers* database was small; the smallest table contained just 5 tuples, while the largest table contained 40 tuples. The EAV database was considerably larger. It was populated by randomly generated data. The entity table contained 3,000 tuples, the *entity_attributes* table contained 2,500 tuples, and the *entity_ea*v table contained 1,000,000 tuples.

8.3.1 Comparing the performance of Nested-IN with Set theoretic operator approach

A set of experiments was conducted to compare the performance of the ‘nested IN’, ‘nested equi-join’, and ‘set theoretic operator’ approaches described at the start of Section 5. The goal of these experiments was to help in deciding which of the three approaches to adopt for the implementation of the S.E.A.L interpreter. The performance of the ‘nested EXISTS’ approach was not tested, since the use of the SQL EXISTS operator results in correlated nested queries, which are known to have worse query costs than equivalent nested non correlated queries (ElMasri&Navathe 2006).

A number of queries involving conjunctive conditional expressions on randomly chosen (*attribute* = *attribute_name*, *value* = *value_data*) pairs were executed on the EAV database. Such queries are indicative of queries involving EAV attributes. The difference in performance was negligible with all three SQL query implementations taking approximately 3 ms to execute.

Another set of experiments involving conjunctive conditional expressions on randomly chosen (*attribute* = *attribute_name*) pairs were executed on the EAV database. Such queries are indicative of entity-association queries involving conventional relational mapping of entity and relationship type structures. The ‘nested IN’ implementation took approximately 12.3 ms, outperforming the ‘nested equi-join’ implementation which took 13.75 ms, and the ‘set theoretic’ implementation which took 15.8 ms. The outcomes of these experiments led to the decision to implement the SEAL interpreter using the ‘nested IN’ approach.

8.3.2 Comparing interpreter overhead with the query execution time

Multiple experiments were performed on the robbers schema to compare the time the S.E.A.L interpreter took to translate S.E.A.L queries with the time the queries took to execute against a small database. The following table presents the measurements for three characteristic entity-association queries.

	S.E.A.L. Syntax	Tran	Exec
Ex7	SELECT nickname FROM robber ASSOCIATED_WITH(bank THROUGH robbery, <bankname = 'Loan Shark'>)	0.2	0.4
	SELECT nickname FROM robber ASSOCIATED_WITH (<bankname = 'Loan Shark'>)	3.6	
Ex8	SELECT nickname FROM robber ASSOCIATED_WITH (skill THROUGH robber_skill, <skillname = 'Guarding'> ASSOCIATED_WITH (test_location THROUGH skill_test, <name = 'Harvard'>))	0.4	0.7
	SELECT nickname FROM robber ASSOCIATED_WITH (<skillname = 'Guarding'> ASSOCIATED_WITH (test_location, <name = 'Harvard'>))	3.7	
Ex4	SELECT nickname FROM robber ASSOCIATED_WITH (robber_attributes THROUGH robber_eav, <attribute = 'music'> AND <value = 'Latin'> AND <attribute = 'haircut'> AND <value = 'Mohawk'>)	1.4	1.2
	SELECT nickname FROM robber ASSOCIATED_WITH (<attribute = 'music'> AND <value = 'Latin'> AND <attribute = 'haircut'> AND <value = 'Mohawk'>)	4.1	
	SELECT nickname FROM robber[haircut = 'Mohawk' AND music = 'Latin']	0.6	0.7

The first column specifies the example where the semantics of queries are defined. The second column contains the S.E.A.L. syntax of the query. The third and fourth columns contain the average query translation time and the average query execution time in milliseconds. For each query semantics, the first table row contains a S.E.A.L. expression where only entity type roles are omitted requiring only a small amount of inference. The second row contains a S.E.A.L. expression where both the associated entity and the relationship types are omitted requiring a considerable amount of inference.

The first query (Ex7) is a simple entity association query. The second query (Ex8) requires nested S.E.A.L. expressions, produces a SQL statement with two levels of nesting, and is a more complex query than the first one. The third query (Ex4) involves a conjunctive condition on EAV attributes of the base entity type. The first two S.E.A.L. expressions treated the EAV structure as a conventional entity-association structure; the third treated EAV attributes as base entity type attributes, forcing SEAL to find their real source by inference.

The table shows that in the case of nearly full syntax expressions, S.E.A.L. translation time is of the same order of magnitude as the query execution time against a small database. The inference generally incurs an overhead that is just a few times greater than in the case of expressions that require practically no inference. The last table row highlights an interesting S.E.A.L. feature. Although the query requires inference, the translation time is lower than in the case of practically no inference. This is because the SEAL expression in the last row does not contain an ASSOCIATED_WITH clause, which incurs many checks. Also, S.E.A.L. produced a simpler and more efficient SQL code resulting in a faster execution time against the robber database.

The S.E.A.L interpreter was then used against the EAV database to perform the queries retrieving entities based solely on EAV attribute conditions. As an indicative result, a query on a conjunctive attribute condition took 1 ms to translate and 12.3 ms to execute.

The experiments showed that the S.E.A.L interpreter incurs an overhead that is negligible in interactive use.

9 Conclusions and Future Work

This paper describes S.E.A.L., a highly declarative extension to SQL for the specification of entity-association queries in an easy and natural way. Entity-association queries constitute a class of queries often needed by users of databases in Medicine, Biology, Chemistry, Genetics and similar fields of science. These queries ask for data about entities based on their participation in relationships with other entities. The complexity of defining entity-association queries in SQL prevents users from exploiting the full potential of information contained in their databases. S.E.A.L is intended to alleviate the problem.

9.1 Contribution

The paper contains the following contributions:

- The definition of the S.E.A.L language syntax that allows association queries to be expressed concisely and simply.
- A description of the S.E.A.L prototype interpreter, including the inference and translation algorithms.
- A set of rules for defining S.E.A.L. compatible database schemas, mainly consisting of good database design practice,
- A description of the Entity Attribute Value (EAV) database structure, since these database structures are particularly suited for databases in many fields of science and in generic eCommerce web database applications, for which queries constitute a subclass of entity-association queries and are hard to define.

With S.E.A.L, end-users can issue entity-association queries without a high level of database proficiency and without detailed knowledge of the database implementation. Users need to know only a part of the conceptual schema, and because of its inference capability, even this requirement is alleviated, since S.E.A.L possesses features of query tools with a Universal Relation Interface. In many cases, a user needs to declare only an entity type and conditions on a number of attributes and can ignore their location.

Another feature of S.E.A.L is that it abstracts EAV storage. Attributes stored in EAV form and in conventional relational form are treated uniformly in queries. Finally, S.E.A.L is not bound to a particular database schema since it is able to use the metadata about the schema extracted from the database system.

Functional and performance testing performed have shown that S.E.A.L represents a good proof of concepts and that it incurs an insignificant overhead.

9.2 Future work

While this work provides a general solution to supporting entity-association queries including queries on EAV structures, there are a number of areas which can be expanded on. Desirable functional features currently not supported by S.E.A.L which need to be addressed in future work include the following:

- Comparison operators other than equality to use in Attribute Constraints,
- Support for relationship cardinalities other M:N,
- Allowing the attribute list after the SELECT clause to contain attributes other than base entity type attributes,
- Extending the support for EAV structures to relationship type EAV attributes, and
- Support of queries with nesting based on associated types of an associated entity type and queries containing logical combinations of `Associations`.

The S.E.A.L. interpreter is currently a standalone program that interacts with a database. This means that users must decide which query system to use for a given task. A very desirable improvement would be to integrate the interpreter into PostgreSQL, as an extension of the SELECT syntax, which would make S.E.A.L. queries very much more accessible and useful to most users.

10 References

ANTLR: ANother tool for Language Recognition http://www.antlr.org/. Accessed 6 Sep 2008.

Corwin, J., Silberschatz A., Miller, P.L. and Marenco, L. (2007): Dynamic tables: An architecture for managing evolving, heterogeneous biomedical data in relational database management systems. *J of the American Medical Informatics Association* **14**(1):86-93.

Dinu, V. and Nadkarni, P. (2006): Guidelines for the effective use of entity-attribute-value modeling for biomedical databases. *International Journal of Medical Informatics* **76**(11-12):769-779.

Elmasri, R. and Navathe, S. (2007): *Fundamentals of Database Systems; 5th edition* Addison-Wesley, Inc.

Nadkarni, P. M. (1996): QAV: Querying entity-attribute-value metadata in a biomedical database. *Computer Methods and Programs in Biomedicine* **53**(2):93-103.

Nadkarni, P. M. and Brandt, C. (1998): Data extraction and ad hoc query of an entity-attribute-value database. *Journal of the American Medical Informatics* **5**(6):511–527.

Nadkarni, P. M., Brandt, C., Frawley, S., Sayward, F., Einbinder, R., Zelterman, D., Schacter, L., and Miller, P. (1998): Managing attribute–value clinical trials data using the ACT/DB client-server database system. *Journal of the American Medical Informatics Association* **5**(2):139-151.

Elmasri, R. and Navathe S. B. (2006): *Fundamentals of Database Systems.* Fifth Edition, Pearson-Addison Wesley.

Ullman, J. D. (1982): *Principles of Database Systems.* Rockville, Maryland, Computer Science Press, Inc.

On Inference of XML Schema with the Knowledge of an Obsolete One

Irena Mlýnková

Charles University, Faculty of Mathematics and Physics, Department of Software Engineering
Malostranské nám. 25, 118 00 Prague 1, Czech Republic
Email: `irena.mlynkova@mff.cuni.cz`

Abstract

The XML has undoubtedly become a standard for data representation and manipulation. But most of XML documents are still created without the respective description of their structure, i.e. an XML schema. Hence, in this paper we focus on the problem of automatic inferring of an XML schema for a given sample set of XML documents. Contrary to existing approaches we propose an algorithm that exploits additional input information – an obsolete XML schema. Consequently, we are able to exploit the information which was correct once and to infer the schema more efficiently.

Keywords: XML Schema, validity, schema inference, schema correction.

1 Introduction

Without any doubt XML (Bray et al. 2006) is currently a de-facto standard for data representation. Its popularity is given by the fact that it is well-defined, easy-to-use and, at the same time, enough powerful. To enable users to specify own allowed structure of XML documents, so-called *XML schema*, the W3C[1] has proposed two languages – DTD (Bray et al. 2006) and XML Schema (Thompson et al. 2004, Biron & Malhotra 2004). The former one is directly part of XML specification and due to its simplicity it is one of the most popular formats for schema specification. The latter language was proposed later, in reaction to the lack of constructs of DTD. The key emphasis is put on simple types, object-oriented features and reusability of parts of a schema or whole schemas.

On the other hand, statistical analyses of real-world XML data show that a significant portion of XML documents (in particular, 52% (Mignet et al. 2003) of randomly crawled or 7.4% (Mlynkova et al. 2006) of semi-automatically collected[2]) still have no schema at all. What is more, XML Schema definitions (XSDs) are used even less (only for 0.09% (Mignet et al. 2003) of randomly crawled or 38% (Mlynkova et al. 2006) of semi-automatically collected XML documents) and even if they are used, they often (in 85% of cases (Bex et al. 2004)) define so-called *local tree grammars*, i.e. languages that can be defined using DTD as well.

In reaction to this situation a new research area of automatic inference of an XML schema has opened. The key aim is to create an XML schema for the given sample set of XML documents that is neither too general, nor too restrictive. Currently there are several proposals of respective algorithms (see Section 2), but there is still a space for further improvements. In this paper we focus on inferring of a schema from a sample set of XML documents in a special situation when we are provided with the original, but already obsolete schema. According to statistical analyses of real-world XML data (Mlynkova et al. 2006) it is quite a common case, since the XML schema is usually considered as a kind of data documentation. Since the schema is not used as it is supposed to be, i.e. for checking the correct structure of XML documents, it is usually not updated in case the respective data are. Hence, in this paper we propose an algorithm which infers a correct schema on the basis of the knowledge of the outdated one. Contrary to existing approaches that would infer a correct schema regardless the existing one, we are able to exploit the information which was correct once and to infer the schema more efficiently.

The paper is structured as follows: Section 2 overviews existing papers on automatic inference of XML schemas as well as issues related to our stated problem. Section 3 provides background information on languages for XML schema definition. Section 4 describes their relation to the theory of automata and grammars and introduces the problem of schema inference. Section 5 describes the proposed solution in detail. And, finally, Section 6 provides conclusions and outlines possible future work.

2 Related Work

The existing solutions to the problem of automatic inference of an XML schema can be classified according to several criteria. Probably the most interesting one is the type of the result (i.e. DTD or XSD) and the way it is constructed, where we can distinguish heuristic methods and methods based on inferring of a grammar.

Heuristic approaches (Moh et al. 2000, Wong & Sankey 2003, Garofalakis et al. 2000) are based on experience with manual construction of schemas. Their output does not belong to any special class of grammars and, hence, we cannot say anything about its features. They are based on generalization of a trivial schema using a set of predefined heuristic rules, such as, e.g., "if there are more than three occurrences of an element, it is probable that it can occur arbitrary times". These techniques can be further divided into methods which generalize the initial schema until a satisfactory solution is reached (e.g. (Moh et al. 2000, Wong & Sankey 2003)) and methods which generate a number of candidates and then choose the optimal one (e.g. (Garofalakis et al. 2000)). While in the first case the methods are threatened by a wrong step which can cause generation of a suboptimal result, in the latter

This work was supported in part by the Czech Science Foundation (GAČR), grants number 201/09/P364 and 201/09/0990.

[1] `http://www.w3.org/`

[2] Data collected with the interference of a human operator.

case they have to cope with space overhead and specifying a reasonable function for evaluation of quality of the candidates.

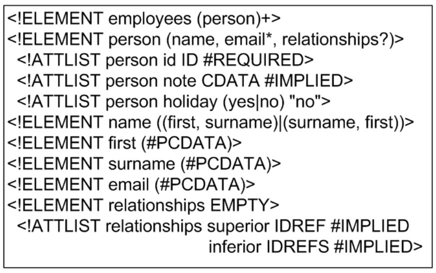
On the other hand, methods based on *inferring of a grammar* (Ahonen 1996, Bex et al. 2007) output a particular class of languages with specific characteristics. Although grammars accepting XML documents are context-free, the problem can be reduced to inferring of a set of regular expressions, each for a single element. But, since according to Gold's theorem (Gold 1967) regular languages are not identifiable in the limit only from positive examples (in our case sample XML documents which should conform to the resulting schema), the existing methods exploit restriction to an *identifiable* subclass of regular languages.

A set of approaches related to our stated problem, i.e. the problem of correcting an incorrect XML schema, are so-called *XML schema evolution* or *XML schema versioning* algorithms (Su et al. 2001, Tan & Goh 2005, Guerrini et al. 2007). However, their aim is opposite to ours. XML schema evolution means that the original schema is replaced by an updated schema and, hence, the effects of the update on its instances need to be solved. In particular, the approaches deal with the problem how document adaptation according to the evolved schema can be (eventually automatically) performed to make them valid again. Schema versioning means that the original documents and schemas should be preserved and a new updated version of the schema is created. Document adaptation is not an issue, but the problem of handling different versions of the same data arises.

Instead of adapting the set of XML documents according to the modified schema we have the opposite task – we want to adapt the given schema according to the set of XML documents. Among the existing works there seems to be only one approach with an aim similar to ours. In (Bertino et al. 2002) the authors propose an approach to evolving a set of DTDs to obtain structures that are correct and precise with regard to a set of XML documents. The approach is intended for a kind of dynamic repository of XML data and DTDs. It is based on exploitation of similarity of XML documents and DTDs and a set of data mining heuristics.

3 XML Schema Languages

The simplest and most popular language for description of the allowed structure of XML documents is currently the Document Type Definition (DTD) (Bray et al. 2006). It enables one to specify allowed elements, attributes and their mutual relationships, order and number of occurrences of subelements, data types and allowed occurrences of attributes. A simple example describing a database of employees is depicted in Figure 1.



```
<!ELEMENT employees (person)+>
<!ELEMENT person (name, email*, relationships?)>
  <!ATTLIST person id ID #REQUIRED>
  <!ATTLIST person note CDATA #IMPLIED>
  <!ATTLIST person holiday (yes|no) "no">
<!ELEMENT name ((first, surname)|(surname, first))>
<!ELEMENT first (#PCDATA)>
<!ELEMENT surname (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT relationships EMPTY>
  <!ATTLIST relationships superior IDREF #IMPLIED
                                   inferior IDREFS #IMPLIED>
```

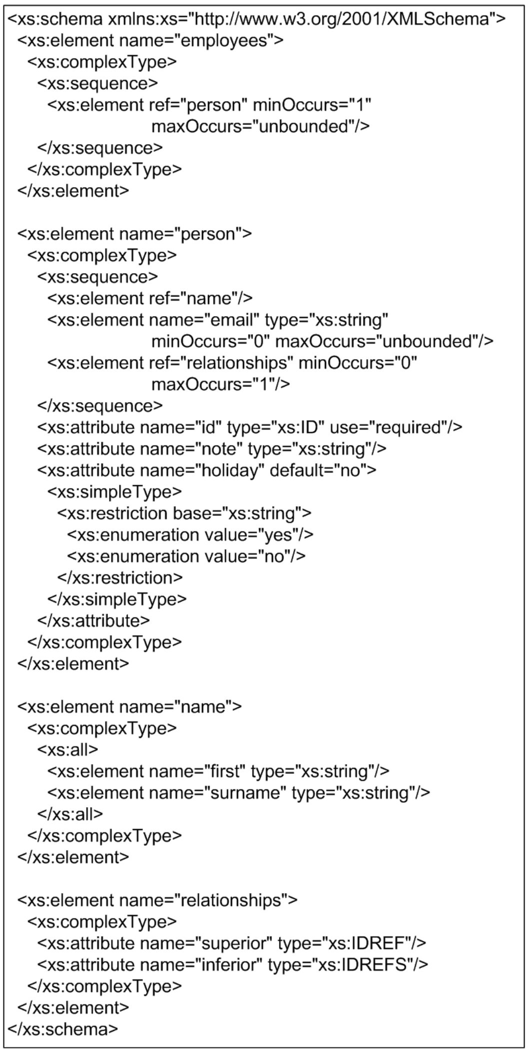
Figure 1: An example of a DTD of employees

At first glance it seems that the specification of the allowed structure is sufficient. Nevertheless, even in this simple example we can find several problems. For instance, we are not able to specify the correct structure of an e-mail address. Similarly, we cannot simply specify that a person can have four e-mail addresses at maximum. And, as we can see, the fact that the order of elements `first` and `surname` is not significant cannot be expressed easily as well. Therefore, the W3C proposed a more powerful tool – the XML Schema language (Thompson et al. 2004, Biron & Malhotra 2004).

The XML Schema language has a number of advantages. The main advantages are that:

- each XSD is a well-formed and valid XML document,
- it has a strong support of data types, both simple and complex and both built-in and user-defined,
- it enables one to re-use and re-define existing schemes or selected parts,
- it enables one to specify the allowed structure using more precise constraints (e.g. minimum and maximum allowed occurrences, ordered/unordered sequences, integrity constraints etc.) and
- it enables one to specify equivalent schemes using distinct constructs.

For example an XSD equivalent[3] to the example of a DTD in Figure 1 is depicted in Figure 2.



```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="employees">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="person" minOccurs="1"
                    maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="person">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element name="email" type="xs:string"
                    minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="relationships" minOccurs="0"
                    maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="required"/>
      <xs:attribute name="note" type="xs:string"/>
      <xs:attribute name="holiday" default="no">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="yes"/>
            <xs:enumeration value="no"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name="name">
    <xs:complexType>
      <xs:all>
        <xs:element name="first" type="xs:string"/>
        <xs:element name="surname" type="xs:string"/>
      </xs:all>
    </xs:complexType>
  </xs:element>

  <xs:element name="relationships">
    <xs:complexType>
      <xs:attribute name="superior" type="xs:IDREF"/>
      <xs:attribute name="inferior" type="xs:IDREFS"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 2: An example of an XSD of employees

[3] Having the same set of document instances.

4 Relation to Automata and Grammars

An XML schema describing the allowed structure of XML documents is an extended context-free grammar (Berstel & Boasson 2000), i.e. a grammar where nonterminals can be rewritten regardless the context in which they occur. The extension is given by the fact that on right hand sides of productions occur regular expressions.

Definition 1 *Given the alphabet* , a regular expression *(RE) over* *is inductively defined as follows:*

- $\emptyset$ (empty set) *and* ϵ (empty string) *are REs.*
- $\forall a \in$: a *is a RE.*
- *If* r *and* r' *are REs of* , *then* (rr') (concatenation), $(r|r')$ (alternation) *and* (r^*) Kleene closure) *are REs.*

The DTD language adds two abbreviations: $(r|\epsilon) = (r?)$ and $(rr^*) = (r^+)$. Also the concatenation is expressed via the '`,`' operator. The XML Schema language adds (among other extensions) another one, so-called *unordered sequence* of REs $r_1, r_2, ..., r_k$, i.e. an alternation of all possible ordered sequences of $r_1, r_2, ..., r_k$. The DTD syntax is often extended with respective '`&`' operator.

Definition 2 An extended context-free grammar *is a quadruple* $G = (N, T, P, S)$*, where* N *and* T *are finite sets of* nonterminals *and* terminals*,* P *is a finite set of* productions *and* S *is a non terminal called a* start symbol. *Each production is of the form* $A \rightarrow r$*, where* $A \in N$ *and* r *is a regular expression over alphabet* $N \cup T$.

The language generated by grammar G *is denoted by* $L(G)$.

A language generated by a grammar can be accepted by an automaton, in our case a finite state automaton.

Definition 3 A finite state automaton (FSA) *is a quintuple* $A = (Q, \ , \delta, S, F)$*, where* Q *is a set of states,* *is a set of input symbols (alphabet),* $\delta : Q \times \ ^* \rightarrow Q$ *is the transition function,* $S \in Q$ *is the start state and* $F \subseteq Q$ *is the set of final states.*

The language accepted by an automaton A *is denoted by* $L(A)$.

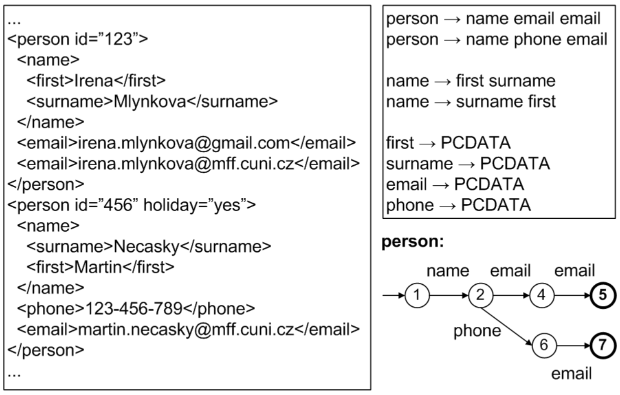
For each RE we can construct a FSA and vice versa.

4.1 Problem Statement

The studied problem can be described as follows: Being given a set of XML documents $D = \{d_1, d_2, ..., d_n\}$ (i.e. words over an alphabet T_D), we search for an XML schema s_D (i.e. a grammar $G_D = (N_D, T_D, P_D, S_D)$) s.t. $\forall i \in [1, n] : d_i$ is valid against s_D (i.e. $D \subseteq L(G_D)$). In particular, we are searching for s_D that is "enough" concise, precise and, at the same time, general.

Most of the existing approaches use the following strategy: For each occurrence of an element $e \in D$ and its subelements $e_1, e_2, ..., e_k$ we construct a production $\vec{p}_e$ of the form $e \rightarrow e_1\ e_2\ ...\ e_k$.[4] The left hand side is called *element type* and denoted $type(\vec{p}_e)$, the right hand side is called a *content model* of the element type and denoted $model(\vec{p}_e)$. The productions form so-called *initial grammar (IG)*. For each element type the productions are then merged, simplified and generalized using various methods and criteria. A common approach is so-called *merging state algorithm*, where a *prefix tree automaton (PTA)* is built from the productions of the same element type and then generalized via merging of its states. Finally, the generalized automata are expressed in syntax of the selected XML schema language.

An example of a IG and PTA for element `person` is depicted in Figure 3.



```
...
<person id="123">
  <name>
    <first>Irena</first>
    <surname>Mlynkova</surname>
  </name>
  <email>irena.mlynkova@gmail.com</email>
  <email>irena.mlynkova@mff.cuni.cz</email>
</person>
<person id="456" holiday="yes">
  <name>
    <surname>Necasky</surname>
    <first>Martin</first>
  </name>
  <phone>123-456-789</phone>
  <email>martin.necasky@mff.cuni.cz</email>
</person>
...
```

person → name email email
person → name phone email

name → first surname
name → surname first

first → PCDATA
surname → PCDATA
email → PCDATA
phone → PCDATA

Figure 3: An example of a IG and a PTA

In the existing works the rules for merging the states of an automaton differ, but they have a common aim to create a concise and precise XML schema. While the heuristic approaches exploit a set of various heuristic rules, the approaches based on inference of a grammar utilize the rules so that the result fulfills the conditions the selected subclass of regular languages states.

The problem we are dealing with is a schema inference task with a special condition – knowledge of an obsolete, i.e. incorrect and/or too general, schema s_{orig}. Our aim is to exploit this additional information in order to speed up the inference process and to make the resulting schema more precise.

5 Proposed Approach

In general the given problem can be divided into checking and correction/adaptation of the following subsets:

1. Simple data types
2. Element/attribute names
3. REs

The first two sets can be solved relatively easily. In case of simple data types we simply check whether the selected data types are not too general or too restrictive and, if necessary, we make the respective corrections. In fact, even most of the existing schema inference methods do not deal with simple types at all.

In case of element/attribute names we can select from two approaches. On one hand, we can consider and distinguish either the same or distinct names. On the other hand, we can take into account their semantics and consider that only an element/attribute name can be modified. However, it is only a question of finding the respective mapping between the names, whereas we can find such mapping only in case the changes are semantically related, such as, e.g., changing `name` into `title`. Consequently, it is only minor aspect of the problem and we will not deal with it in the following text as well.

The most important task of the given problem is to check and correct REs. In general we can encounter the following cases:

1. The original XML schema s_{orig} does not need to be corrected. The XML documents in D and valid against it and it is enough concise and precise.
2. The XML documents in D are valid against s_{orig}, however it is too general. In particular, there can occur the following cases:
   (a) Too high upper limit of occurrences (see Example 1)

[4] Attributes are often omitted for simplicity.

   (b) Too low lower limit of occurrences (see Example 2)

   (c) Occurrence of redundant data (see Example 3)

3. The XML documents in D are not valid against s_{orig} anymore. In particular, there can occur two situations:

   (a) s_{orig} does not involve items that XML documents in D do (see Example 4).

   (b) The XML documents do not involve items that are in s_{orig} mandatory (see Example 5).

Example 1 *Consider the following set of productions extracted from XML documents:*

E $\rightarrow$ A B C C C
E $\rightarrow$ A B C C

and the following production taken from s_{orig}:

E $\rightarrow$ A B C+

The production should be corrected, since the + operator should be used only in case more than 5 occurrences of an element.

Example 2 *Consider the following set of productions extracted from XML documents:*

E $\rightarrow$ A B C C C
E $\rightarrow$ A B C C

and the following production taken from s_{orig}:

E $\rightarrow$ A B? C

The production should be corrected, since the element B *is present in all document instances.*

Example 3 *Consider the following set of productions extracted from XML documents:*

E $\rightarrow$ A B C C C
E $\rightarrow$ A B C C

The following productions from s_{orig} need to be corrected since they contain redundant data with regard to the given documents:

E $\rightarrow$ A B X? C+
E $\rightarrow$ A (B | X) C+

Example 4 *Consider the following set of productions extracted from XML documents:*

E $\rightarrow$ A B C C C
E $\rightarrow$ A B C C

The following production from s_{orig} needs to be corrected since it does not involve element B *present in the documents:*

E $\rightarrow$ A C+

Example 5 *Consider the following set of productions extracted from XML documents:*

E $\rightarrow$ A B C C C
E $\rightarrow$ A B C C

The following production from s_{orig} needs to be corrected since it involves compulsory element X *not present in the documents:*

E $\rightarrow$ A B C+ X

5.1 Possible Solutions

The first possible solution is to simply ignore s_{orig} and infer a correct schema purely on the basis of D. The advantage of this approach is obvious – we use a verified approach that provides a correct result. However, we do not exploit an available and apparently useful information. Hence, our aim is to exploit this information when appropriate and, thus, to speed up the inference process and provide a more precise schema.

The second natural approach can be based on the following simple observation: The existing inference methods produce plenty of possible solutions, that are evaluated and the (sub)optimal one is selected as the result. It is caused by the fact that the approaches are based on heuristic rules that generalize the IG. The amount of generalizations is high and we do not know which is the optimal one unless we combine it with the rest of the schema. Hence, a natural idea may be that we will exploit the knowledge of s_{orig} in situations when there are multiple generalization possibilities. However, the problem is that this approach can be exploited only in case of simple REs. Otherwise, the inclusion, equivalence and intersection problem of REs cannot be solved in reasonable time and, consequently, we cannot easily find the related schema fragments.

Consequently, the solution we propose is a relaxed version of the two described approaches. We do exploit s_{orig}, however, we do not stick to it 100%. In addition, we are able to find its suboptimal correction/adaptation with reasonable complexity.

5.2 Proposed Solution

In the approach we propose we firstly divide the given problem into two independent and optional steps:

1. Correction of the input schema

2. Specialization of the input schema

In the fist step we assume there exists at least one document $d \in D$ s.t. d is not valid against s_{orig}. Hence, we need to find schema $s_{correct}$, i.e. the correction of s_{orig}, s.t. for $\forall d \in D$: d is valid against $s_{correct}$. In addition, let $_{correct}$ be the set of all possible corrections of s_{orig}. Then we want to find a correction $s_{correct}$ s.t. $dist(s_{orig}, s_{correct}) \leqslant dist(s_{orig}, s)$ for $\forall s \in {}_{correct}$, where $dist(s, s')$ is the edit distance, i.e. the sequence of operations for transforming s to s'. In other words, we want to find a correction that requires the least modifications of s_{orig}.

In the second step we assume that we have a schema $s_{correct}$, s.t. $\forall d \in D$: d is valid against $s_{correct}$. However, we want to specialize the REs involved in the schema with regard to the data in D, resulting in more precise and readable schema $s'_{correct}$.

Note that any of the steps can be used separately. On one hand, we may require only the correction step without any unnecessary schema modifications. On the other hand, we may have a correct schema but we want to make it more precise, since we know that the data are more specific.

5.2.1 Schema Correction

First of all, let us mention the fact that each content model of an XML document must be so-called *deterministic* or *1-unambiguous*.

Example 6 *Consider the following content model:*

(A B) | (A C)

It is non-deterministic, because while reading A*, the XML processor cannot know which* A *in the model is being matched without looking ahead to see which element follows. On the other hand, an equivalent content model:*

A (B | C)

is deterministic. The processor does not need to look ahead to see what follows; either B *or* C *will be accepted.*

This requirement is stated directly in the W3C specification of XML (Bray et al. 2006) and ensures that an XML processor can match the schema with the data efficiently. And, consequently, we are able to determine the validity of the documents in D efficiently as well.

The correction algorithm consists of the following steps:

1. We divide the set D into sets D_{valid} and $D_{invalid}$, i.e. valid and invalid documents, s.t. $D_{valid} \cup D_{invalid} = D$ and $D_{valid} \cap D_{invalid} = \emptyset$.
2. For $\forall d \in D_{invalid}$ we create the respective set of productions and merge them with s_{orig}.

The key step of the approach is merging a single production $\vec{p}_e$ created from an element e and its subelements in XML document $d \in D_{invalid}$ with productions of s_{orig}. The merging algorithm can be described as follows: Firstly, we identify production $\vec{q}_e$ from s_{orig} to be merged with. For this purpose we can use any of the strategies used in the existing works for grouping the productions. In most of them the productions are simply grouped according to equivalence of element types, more sophisticated approaches take into account also greater context. Since this is not the key aspect of our proposal, we will further assume the former approach.

Having the two productions $\vec{p}_e$ and $\vec{q}_e$ to be merged, we parse the $model(\vec{p}_e)$. Similarly to the approach of merging productions into a PTA, we match the elements of $model(\vec{p}_e)$ with $model(\vec{q}_e)$ until the parsing does not fail. Whenever we reach an element $e' \in model(\vec{p}_e)$ that invokes invalidity, we create a separate branch of automaton for $\vec{q}_e$ consisting of the rest of the content model staring with e'.

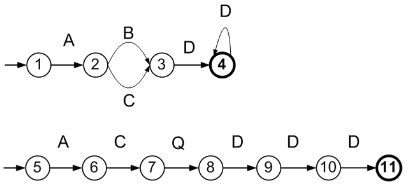
Example 7 *Consider the following example of schema production $\vec{q}_E$:*

$E \rightarrow A \ (B \ | \ C) \ D+$

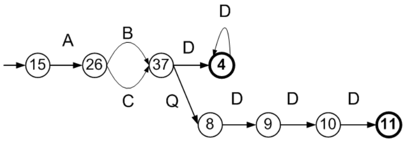
and the following example of document production $\vec{p}_E$:

$E \rightarrow A \ C \ Q \ D \ D \ D$

The automata describing the productions are depicted as follows:



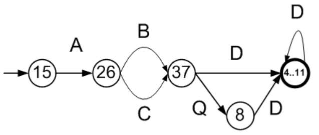
Using the above described algorithm, they are merged into the following automaton:



(Note that if we merge s_{orig} with productions of $d \in D_{valid}$, the automata of s_{orig} do not change, since there occurs no element that would violate creating of a new branch.)

After merging each of the automata, the newly created schema $s_{correct}$ ensures that each $d \in D$ is valid against $s_{correct}$. However, the respective automata, i.e. REs, are not very concise and precise. Therefore, we need to apply an approach that would merge the newly added branches more precisely.

Example 8 *Consider the merged automaton in Example 7. After more elaborate merging of states of the new branch, we get the following more concise result:*



Since there can exist multiple ways how to merge the newly added states with the original ones, we exploit a modification of existing general approach to schema inference that can cope with all the possible cases. In particular, we utilize an approach from (Vosta et al. 2008) since it is one of the recent approaches that combines most of the previously proposed and verified methods.

Firstly, note that the problem of generalization of an automaton is viewed as a kind of optimization problem.

Definition 4 *A model* $M = (\ , \ , \sigma)$ *of a* combinatorial optimization problem *consists of a search space of possible solutions to the problem (so-called* feasible region*), a set of constraints over the solutions and an* objective function $\sigma : \ \rightarrow \mathbb{R}_0^+$ *to be minimized.*

In our case consists of all possible generalizations of an automaton. As it is obvious, is theoretically infinite and thus, in fact, we can search only for a reasonable suboptimum. Therefore, we use a modification of *ACO heuristics* (Dorigo et al. 2006). is given by the features of XML schema language we are focussing on. And finally, to define σ we exploit a modification of the *MDL principle* (Grunwald 2005).

Ant Colony Optimization (ACO) The ACO heuristics is based on observations of nature, in particular the way ants exchange information they have learnt. A set of artificial “ants” $= \{a_1, a_2, ..., a_{card(\Lambda)}\}$ search the space trying to find the optimal solution $s_{opt} \in$ s.t. $\sigma(s_{opt}) \leqslant \sigma(s); \forall s \in$. In i-th iteration each $a \in A$ searches a subspace of for a local suboptimum until it “dies” after performing a predefined amount of steps N_{ant}. While searching, an ant a spreads a certain amount of “pheromone”, i.e. a positive feedback which denotes how good solution it has found so far. This information is exploited by ants from the following iterations to choose better search steps. The search terminates either after a specified number of iterations N_{iter} or if $s'_{opt} \in$ is reached s.t. $\sigma(s'_{opt}) \leqslant T_{max}$, where T_{max} is a required threshold.

The obvious key aspect of the algorithm is one step of an ant. Each step consist of generating of a set of possible continuations, their evaluation using σ and execution of one of the candidate steps. The executed step is selected randomly with probability given by σ. And this is the biggest strength of the ACO heuristics. Contrary to greedy search strategy which can get stuck in local suboptimum, ACO is able to search greater subspace of due to random selection of continuations and possible temporal moving to a worse case.

Generating a Set of Possible Continuations A single step of an ant is represented using a modification of the current automaton. As we have mentioned, most of the existing approaches exploit the merging state strategy, i.e. reduction of the set of states of the automaton on the basis of various rules, such as k,h-*context* (Ahonen 1996) which merges states with same contexts (prefixes) or s,k-*string* (Wong & Sankey 2003) which merges states with same suffixes.

We will preserve the same merging strategies, the key difference is in the set of states that can be merged. In the original algorithm, any of the states of the automaton that fulfills any of the merging conditions can be merged. In our case we do not want to modify the original automaton, since we want to preserve the information it carries. Therefore, we restrict the merging only to cases when the set of merged states involves at least one of the states of the new branch. Consequently, we can encounter the following two situations:

1. We merge the states within the new branch, i.e. we truncate the new branch.

2. We merge a state of the new branch with an original one, i.e. we reduce the number of states of the whole automaton.

Evaluation of Continuations The evaluation of moving from schema s_x to s_y, where $s_x, s_y \in$, remains in our case the same. In particular, it is defined as:

$$mov(s_x, s_y) = \sigma(s_x) - \sigma(s_y) + pos(s_x, s_y)$$

where σ is the objective function and $pos(s_x, s_y) \geqslant 0$ is the positive feedback of this step from previous iterations. For the purpose of specification of σ, most of the existing works exploit the MDL principle (Garofalakis et al. 2000). It is based on two observations: A good schema should be enough general which is related to the low number of states of the automata. On the other hand, it should preserve details which means that it enables one to express document instances in D using short codes. In other words, most of the information is carried by the schema itself and, thus, it does not need to be encoded. Hence, the quality of a schema $s \in$ described using a set of productions $R_s = \{\vec{p}_1, \vec{p}_2, ..., \vec{p}_{card(R_s)}\}$ is expressed using:

- the size (in bits) of R_s and
- the size (in bits) of codes of document instances in D expressed using R_s.

Let O be the set of allowed operators and E the set of distinct element names in D. Then we can view $model(\vec{p})$ of $\forall \vec{p} \in R_s$ as a word over $O \cup E$ and its code can be expressed as $|model(\vec{p})| \cdot \lceil \log_2(card(O) + card(E)) \rceil$, where $|model(\vec{p})|$ denotes length of word $model(\vec{p})$. The size of code of a single instance $d \in D$ is defined as the size of code of an inferring sequence of productions $R_d = \langle \vec{g}_1, \vec{g}_2, ..., \vec{g}_{card(R_d)} \rangle$ necessary to convert the initial nonterminal to d using productions from R_s. Since we can represent the sequence R_d as a sequence of ordinal numbers of the productions in R_s, the size of the code of d is $card(R_d) \cdot \lceil \log_2(card(R_s)) \rceil$.

5.2.2 Schema Specialization

In the second step of the proposed algorithm we assume that we are provided with a correct schema $s_{correct}$. Our current aim is to specify the schema using a more precise schema $s'_{correct}$. And naturally, we want to preserve the validity condition for all documents in D.

The problem of schema specialization can be divided into several steps:

1. Pruning of unused schema fragments
2. Correction of lower and upper bounds of occurrences of schema fragments
3. Correction of operators
4. Refactorization

According to user requirements, selected steps can be omitted depending on the respective application. For instance a user may omit step 1. requiring that unused schema fragments should be preserved. In fact, even the whole specialization process can be omitted in case we want to preserve the information from the original schema s_{orig} as much as possible.

Unused Schema Fragments The aim of this step is to identify schema fragments that are not used in the sample XML documents D. The approach can be described as follows: For each element e in the given schema $s_{correct}$ we preserve a *usage flag* $\varphi_{used}(e)$ that carries the information about its usage in D. At the beginning of the algorithm we set $\varphi_{used}(e) = F$ (false) for $\forall e \in s_{correct}$. Using an XML parser we parse each $d \in D$, we check usage of particular elements of $s_{correct}$ and set $\varphi_{used}(e) = T$ (true) whenever $e \in s_{correct}$ is used. After parsing the whose set D we check the flag φ_{used}. All elements $e \in s_{correct}$ s.t. $\varphi_{used}(e) = F$ together with the associated operators can be eliminated since they are not used in the sample data.

Example 9 *Consider the following set of productions extracted from XML documents:*

E → A C
E → A B B B C
E → A B C
E → A B B B B

The following production from $s_{correct}$ involves fragment Q? *not used in the documents.*

E → A B* C? Q?

Parsing the content model of the first production we set φ_{used}(A) $= T$ and φ_{used}(C) $= T$, i.e.

E → A B* C? Q?
T F T F

Parsing the second production we set φ_{used}(A) $= T$, φ_{used}(B) $= T$, φ_{used}(B) $= T$, φ_{used}(B) $= T$ and φ_{used}(C) $= T$, i.e.

E → A B* C? Q?
T T T F

Similarly we process the remaining productions which do not change the current settings. Finally, we can see that schema fragment Q? *is not used in any of the input documents and, hence, we can specialize the schema to:*

E → A B* C?

Note that since we assume that each $d \in D$ is valid against $s_{correct}$, the elimination of unused schema fragments is a correct application which preserves correctness of the content models, as well as validity of the data in D. It can be proven as follows: Since the data are valid, a candidate for elimination must be an optional schema fragment, i.e. an item of a sequence associated with either ? or * operator or an item of a choice. Hence the elimination causes either truncating of the sequence or reduction of options of the choice. The schema fragment can be either a single element or a sequence of elements. In the latter case, again due to the assumption of validity, all the elements in the sequence have φ_{used} of F and, hence should be eliminated.

Finally, note that this simple strategy can be applied on both DTDs and XSDs, since their treatment in case of used and unused schema fragments is the same.

Occurrences In the second step we want to correct the allowed numbers of occurrences of schema fragments, i.e. operators ?, + and * in case of DTD or minOccurs and maxOccurs attribute values in case of XSD. Similarly to the previous case for each fragment f in the given schema $s_{correct}$ we preserve *minimum repetition flag* $\varphi_{min}(f)$ and *maximum repetition flag* $\varphi_{max}(f)$ that carry the information about its minimum and maximum amount of successive occurrences in D.

At the beginning of the algorithm we set $\varphi_{min}(f) = \infty$ and $\varphi_{max}(f) = 0$ for each fragment $f \in s_{correct}$. Using an XML parser we again parse each $d \in D$. For each repeating sequence of a schema fragment f we determine its length l_f, i.e. the amount of repetitions. If $l_f < \varphi_{min}(f)$, we set $\varphi_{min}(f) = l_f$. If $l_f > \varphi_{max}(f)$, we set $\varphi_{max}(f) = l_f$.

Example 10 *Consider the following set of productions extracted from XML documents:*

E → A
E → B
E → A A

The following production from $s_{correct}$ should be specialized.

E → A+ | B | (C D)

Parsing the content models of the productions we set φ_{min} and φ_{max} as follows:

		E → A+ |	B |	(C D)
Start:	φ_{min}	∞	∞	∞
	φ_{max}	0	0	0
E → A	φ_{min}	1	0	0
	φ_{max}	1	0	0
E → B	φ_{min}	0	0	0
	φ_{max}	1	1	0
E → A A	φ_{min}	0	0	0
	φ_{max}	2	1	0

The resulting values of φ_{min} and φ_{max} carry information about minimum and maximum occurrences of the respective schema fragments. In particular:

- If $\varphi_{min} = 0$, the respective schema fragment has optional occurrence.
- If $\varphi_{min} > 0$, the respective schema fragment has compulsory occurrence.
- If $\varphi_{max} > 1$, the respective schema fragment has multiple occurrence.

In case we correct an XSD, we can use φ_{min} and φ_{max} as new values for `minOccurs` and `maxOccurs` attributes of respective schema fragments. In case we correct a DTD, we transform the values of φ_{min} and φ_{max} to respective DTD operators – see Table 1, where rep_{min} is the minimal occurrence which induces generalization to arbitrary occurrences.

φ_{min}	φ_{max}	**DTD operator**
0	1	?
0	$> rep_{min}$	*
> 0	$> rep_{min}$	+

Table 1: Transformation of φ_{min} and φ_{max} to DTD operators

In addition, note that the values of φ_{max} also carry the same information as φ_{used}. In particular:

- If $\varphi_{max} = 0$, then $\varphi_{used} = F$.
- If $\varphi_{max} \neq 0$, then $\varphi_{used} = T$.

Consequently, using φ_{min} and φ_{max} we can also identify the unused schema fragments.

Operators Apart from unused schema fragments and imprecise minimum and maximum occurrences, there can occur also too general combinations of operators and allowed occurrences. In particular, we will deal with various combinations of '`|`' (`choice`), '`,`' (`sequence`) and '`(`', '`)`' (`group`) constructs of DTD (XSD).

In general, there can occur two situations, so-called *grouping* and *degrouping*. We will depict them by simple rules listed in Figure 4.

$$a?, b? \rightarrow (a, b)?$$
$$a?, b^* \rightarrow (a, b^+)?$$
$$a?, b? \rightarrow a|b$$
$$a?, b^* \rightarrow a|b^*$$

Figure 4: Grouping and degrouping rules

In general, both types of rules transform the content models to more restrictive ones. Hence, naturally, we can apply the rules only in case the input data are valid also against the more restrictive version.

The algorithm for finding candidates for grouping and degrouping is similar to the previous case: For each of the REs that conform to the left hand sides of the rules in Figure 4, we need to check that the input data conform to their right hand sides as well. While parsing the documents in D, for each of the candidate schema fragment f and for each grouping/degrouping rule $r_1, r_2, ..., r_k$ we preserve the respective flags $\varphi_{r_1}(f), \varphi_{r_2}(f), ..., \varphi_{r_k}(f)$ carrying the information whether or not the instances of f in D conform to right hand side of the respective rule. At the beginning of the algorithm we set the flags $\varphi_{r_1}(f) = T, \varphi_{r_2}(f) = T, ..., \varphi_{r_k}(f) = T$. While parsing the documents, whenever we encounter a document instance that does not fulfill the right hand side of a rule r_i, we set the respective $\varphi_{r_i}(f) = F$. After parsing each $d \in D$ we can apply grouping and degrouping rules only in case the respective flag remains positive, i.e. there occurs no document instance in D that would not be valid against it.

Example 11 *Consider the following schema production:*
`E → A? B? C D*`
In case the document productions are as follows:
`E → A B C D D D D`
`E → A B C`
`E → C D D`
we can apply operation grouping resulting in the following schema production:
`E → (A B)? C D*`

On the other hand, if the document productions are as follows:
`E → A C D D D D`
`E → B C`
`E → A C D D`
we can apply operation degrouping resulting in the following schema production:
`E → (A | B) C D*`

Note that we can use a much wider set of grouping and degrouping rules, depending on user requirements on schema correction. However, if the set of rules is too wide, it can generate too large set of possible combinations and, hence, we should use the classical merging state algorithm instead, since it would enable one to find the suboptimal solution efficiently. The decision remains in hands of a user and his/hers requirements for schema specialization.

Refactorization A natural last step of each of schema inference method is *refactoring*, i.e. improving readability and simplifying structure while preserving the functionality of the resulting schema. A demonstrative set of rules is depicted in Figure 5.

$$a?? \rightarrow a?$$
$$a^{++} \rightarrow a^+$$
$$a^{**} \rightarrow a^*$$
$$a^*? \rightarrow a^*$$
$$a?^* \rightarrow a^*$$
$$a^{+*} \rightarrow a^*$$
$$a^{*+} \rightarrow a^*$$
$$a?^+ \rightarrow a^*$$
$$a^+? \rightarrow a^*$$
$$aa^* \rightarrow a^+$$
$$a^+a^* \rightarrow a^+$$
$$a?a^+ \rightarrow a^*$$
$$(ab)|(ac) \rightarrow a(b|c)$$

Figure 5: Merging of operators

The specified rules enable one to remove duplicate occurrence operators, to merge sequences of distinct occurrence operators into a single one, to merge sequences of the same fragments, to avoid nondeterministic content models etc. Naturally, there can exist various other sets of refactorization rules depending on the requirements of respective applications.

5.3 Complexity

The proposed algorithm consists of schema correction and schema specialization. Schema correction is performed using the ACO heuristic whose complexity in the worst case is limited by the allowed number of iterations, number of steps of an ant and number of ants, i.e. $O(N_{iter} \times N_{ant} \times card(\ \ ))$. On the other hand, schema specialization is based on linear parsing of XML documents in D, i.e. $O(|D| \times max_{i=1}^{card(D)}(|d_i|))$, where $|d_i|$ denotes the number of elements and attributes in document d_i. Naturally, if we decide to perform the schema specialization using the classical ACO heuristic, it will have the same complexity as the correction algorithm. Consequently, the ACO heuristic enables one to search a greater space of possible solutions and, hence to find a better solution, but at the cost of efficiency.

In general, even if we use the ACO heuristic instead of the proposed simple heuristic strategy, the inference algorithm will be still more efficient than the original one that does not take into account the original schema. The reason is that we do not begin with simple PTA, but with an XML schema that is at least partly correct and enough generalized. If we consider the worst case, i.e. the case when the input schema s_{orig} is completely incorrect, the proposed approach builds a classical PTA from the given XML documents and merges them. The unused schema fragments of s_{orig} are then simply removed in linear time.

6 Conclusion

The aim of this paper was to propose an algorithm for automatic inference of an XML schema which exploits an additional information – the original, possibly incorrect or too general schema. We have proposed a two-step approach. Firstly, we correct the schema so that the input XML documents are valid against it, whereas the new schema preservers the information carried by the original one as much as possible. Secondly, we propose a heuristic approach that enables one to specify the schema more precisely. In particular, we propose two alternatives which differentiate in efficiency and quality of the result.

Currently, we are dealing with throughout implementation of the proposal, since we intend to apply it on a representative set of real-world XML data as well as to exploit it in existing applications (Dokulil et al. 2007). We assume that similarly to paper (Bex et al. 2007) we will discover that the real-world data need special treatment since they do not involve all the constructs allowed by the W3C specifications.

Our future work we will focus mainly on integrating of user interaction which is the key aspect in case multiple solutions are available and searching the optimum is made only using heuristics. In fact, there seems to be no work, that would deal with this topic in detail, taking into account reasonable requirements for user's skills and amount of decisions to be made. Next, we will deal with inference of further XSD specific features, in particular integrity constraints. And, finally, since for further processing of the respective XML data also constraints that cannot be expressed in XSD may be useful, we will try to get also beyond its expressive power.

References

Ahonen, H. (1996), *Generating Grammars for Structured Documents Using Grammatical Inference Methods*, Technical Report A-1996-4, Dept. of Computer Science, University of Helsinki.

Berstel, J. & Boasson, L. (2000), XML Grammars, *in* 'Mathematical Foundations of Computer Science', LNCS, Springer, pp. 182–191.

Bertino, E., Guerrini, G., Mesiti, M. & Tosetto, L. (2002), Evolving a Set of DTDs According to a Dynamic Set of XML Documents, *in* 'EDBT '02', Springer-Verlag, London, UK, pp. 45–66.

Bex, G. J., Neven, F. & den Bussche, J. V. (2004), DTDs versus XML Schema: a Practical Study, *in* 'WebDB'04', ACM, New York, NY, USA, pp. 79–84.

Bex, G. J., Neven, F. & Vansummeren, S. (2007), Inferring XML Schema Definitions from XML Data, *in* 'VLDB'07', ACM, Vienna, Austria, pp. 998–1009.

Biron, P. V. & Malhotra, A. (2004), *XML Schema Part 2: Datatypes (Second Edition)*, W3C.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E. & Yergeau, F. (2006), *Extensible Markup Language (XML) 1.0 (Fourth Edition)*, W3C.

Dokulil, J., Tykal, J., Yaghob, J. & Zavoral, F. (2007), Semantic Web Repository And Interfaces, *in* 'SEMAPRO'07', IEEE Computer Society, Los Alamitos, USA, pp. 223–228.

Dorigo, M., Birattari, M. & Stutzle, T. (2006), *Ant Colony Optimization – Artificial Ants as a Computational Intelligence Technique*, Technical Report 2006-023, IRIDIA, Bruxelles, Belgium.

Garofalakis, M., Gionis, A., Rastogi, R., Seshadri, S. & Shim, K. (2000), XTRACT: a System for Extracting Document Type Descriptors from XML Documents, *in* 'SIGMOD'00', ACM, New York, NY, USA, pp. 165–176.

Gold, E. M. (1967), 'Language Identification in the Limit', *Information and Control* **10**(5), 447–474.

Grunwald, P. (2005), *A Tutorial Introduction to the Minimum Description Principle*. `http://homePAGES.cwi.nl/~pdg/ftp/mdlintro.pdf`.

Guerrini, G., Mesiti, M. & Sorrenti, M. A. (2007), XML Schema Evolution: Incremental Validation and Efficient Document Adaptation, *in* 'XSym'07', Springer, Vienna, Austria, pp. 92–106.

Mignet, L., Barbosa, D. & Veltri, P. (2003), The XML Web: a First Study, *in* 'WWW'03', ACM, New York, NY, USA, pp. 500–510.

Mlynkova, I., Toman, K. & Pokorny, J. (2006), Statistical Analysis of Real XML Data Collections, *in* 'COMAD'06', Tata McGraw-Hill, New Delhi, India, pp. 20–31.

Moh, C.-H., Lim, E.-P. & Ng, W.-K. (2000), Reengineering Structures from Web Documents, *in* 'DL'00', ACM, New York, NY, USA, pp. 67–76.

Su, H., Kramer, D., Chen, L., Claypool, K. & Rundensteiner, E. A. (2001), XEM: Managing the Evolution of XML Documents, *in* 'RIDE '01', IEEE Computer Society, Washington, DC, USA, p. 103.

Tan, M. & Goh, A. (2005), Keeping Pace with Evolving XML-Based Specifications, *in* 'Current Trends in Database Technology – EDBT '04 Workshops', Springer, Heraklion, Crete, Greece, pp. 280–288.

Thompson, H. S., Beech, D., Maloney, M. & Mendelsohn, N. (2004), *XML Schema Part 1: Structures (Second Edition)*, W3C.

Vosta, O., Mlynkova, I. & Pokorny, J. (2008), Even an Ant Can Create an XSD, *in* 'DASFAA'08', LNCS, Springer, pp. 35–50.

Wong, R. K. & Sankey, J. (2003), *On Structural Inference for XML Data*, Technical Report UNSW-CSE-TR-0313, School of Computer Science, University of NSW.

Elliptic Indexing of Multidimensional Databases

Ondrej Danko[1] **Tomáš Skopal**[2]

[1] Comenius University in Bratislava, FMUK, Department of Information Systems
Odbojárov 10, 820 05 Bratislava, Slovak Republic
Email: `ondrej.danko@fm.uniba.sk`

[2] Charles University in Prague, FMP, Department of Software Engineering
Malostranské nám. 25, 118 00 Prague, Czech Republic
Email: `skopal@ksi.mff.cuni.cz`

Abstract

In this work an R-tree variant, which uses minimum volume covering ellipsoids instead of usual minimum bounding rectangles, is presented. The most significant aspects, which determine R-tree index structure performance, is an amount of dead space coverage and overlaps among the covering regions. Intuitively, ellipsoid as a quadratic surface should cover data more tightly, leading to less dead space coverage and less overlaps. Based on studies of many available R-tree variants (especially SR-tree), the eR-tree (**e**llipsoid R-tree) with ellipsoidal regions is proposed. The focus is put on the algorithm of ellipsoids construction as it significantly affects indexing speed and querying performance. At the end, the eR-tree undergoes experiments with both synthetic and real datasets. It proves its superiority especially on clustered sparse datasets.

1 Introduction

In the last decades, the demand for efficient searching in large multimedia databases has begun emerging much more often than anytime before. Especially the applications from areas like medicine, geography or CAD experienced an absence of techniques which would enable them to efficiently search in protein databases, geographical maps or CAD datasets. Because the nature of geographical or medicine data differs, a custom solution would be always required to provide efficient retrieval. To solve this problem, simple, yet powerful idea is applied – *feature transformation*. Each object of a dataset is transformed into a tuple of n-dimensional space representing that object (based on certain transformation rules). Afterwards, a multidimensional indexing technique is employed to enable fast retrieval. The most obvious query type in multidimensional indexing is *window query* (or range query). A window query simply specifies some portion of the n-dimensional space, a window (set of intervals on all dimensions), and returns all dataset tuples inside.

2 Related Work

To efficiently search for tuples inside a query window, we have to employ a *spatial access method* (SAM, or called a multidimensional index). In this section we give an overview of the most successful SAMs, all based on R-tree.

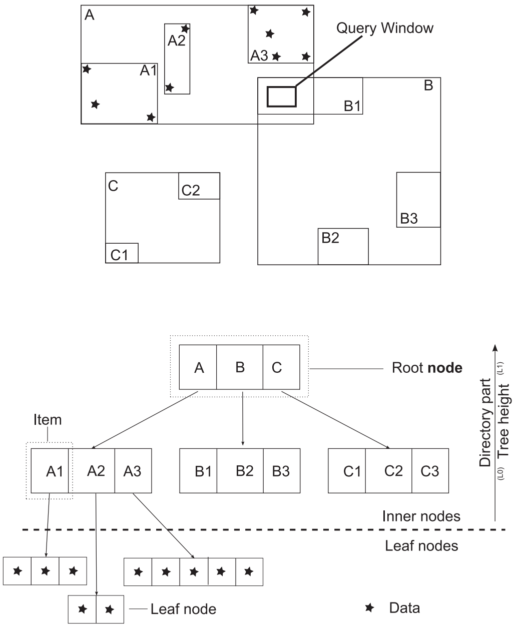


Figure 1: R-tree index structure.

2.1 R-tree

The R-tree (Guttman 1984) is a height-balanced multidimensional index structure similar to B-tree. The basic idea behind the R-tree is to hierarchically partition the search space into nested regions. The space partitioning is neither complete nor disjoint, while the nested hierarchies are formed as paths from the root node to the leaf nodes. The root node region encloses regions of all its child nodes. The R-tree consists of inner and leaf nodes. A leaf node holds the data tuples and consists of entries $\langle I, data\rangle$, where I represents a minimum bounding rectangle (MBR) of *data*. Similarly, an inner node consist of entries $\langle I, child\_pointer\rangle$ where I is MBR of all entries contained in node referenced by *child_pointer*, see Figure 1. Searching in R-tree index structure means traversing those paths of the tree that intersect with the search region (query window in our case).

2.2 Other SAM

An extensive study of the original Guttman's R-tree on different data distributions led to proposal of some optimizations, as:

- minimization of leaf level node region overlaps

- minimization of leaf level regions surface
- minimization of volume of inner nodes
- maximization of storage utilization

2.2.1 R*-tree

The R*-tree (Beckmann et al. 1990) introduces *forced reinsertions*, because R-tree-based structures are highly sensitive to the order in which are tuples inserted. When a leaf node becomes overfull, some portion of its member tuples which are most distant from the center of the node's MBR are deleted from index and reinserted again. This improvement pushed the storage utilization to 71% -76%. The R*-tree also prefers squared MBRs over rectangular.

2.2.2 R$^+$-tree

The key idea behind the R$^+$-tree (Sellis et al. 1987) is an overlap-free splitting in the tree directory (the inner nodes). Generally, there is no guarantee that such splitting exists. In case there exists no overlap-free splitting, the R$^+$-tree introduces a *forced splitting.* When considering the example in Figure 1, to get rid of overlap between the A and B regions, we necessarily need to split the region B1. Of course, in some situations forced splits need to be propagated until we reach leaf nodes, whereas the number of forced splits can exponentially increase till we reach them. As a side effect of forced splits, unlike regular R-tree, the R$^+$-tree also cannot guarantee 50% space utilization. On the other hand, the authors of R$^+$-tree claim their modification requires 50% less page accessess (on average) to process a query, compared to the R-tree.

2.2.3 SS-tree

The SS-tree (White & Jain. 1996) was introduced to support similarity searches (meaning nearest neighbor queries) in higher dimensions (thus **S**imilarity **S**earch-**tree**). The SS-tree uses spheres instead of MBRs as page regions. When comparing properties of MBRs with spheres, it should be said that:

- Bounding spheres tend to produce regions bigger in *volume* than MBRs.
- Bounding spheres tend to produce regions of smaller *diameter* than MBRs.

When using spheres, the first mentioned property decreases the performance of window queries, while the second one favors the nearest neighbor queries. Another advantage of spheres is that they require less space to be stored than MBRs. For MBR we need to store two n-dimensional vectors, while for sphere it is enough to store one n-dimensional center and a one-dimensional diameter. This allows higher fanout of nodes, thus it eventually decreases the tree height. For performance reasons the spheres of SS-tree are not minimum volume spheres, but use centroids as their centers. Thus, the center of the sphere is computed as the average in each dimension of the data tuples being bounded. The diameter is then calculated so that it covers all tuples.

The SS-tree uses forced reinserting when an overflow is encountered; 30% of tuples with highest distance from the center of sphere are reinserted. While the storage utilization of R*-tree is just 70-75%, the SS-tree reaches 85% on average. The splitting is based on variance. First, the dimension with highest variance is choosen. Then a split plane orthogonal to that dimension is found, so that the sum of variances in both, the new node and the old node, is minimized. The authors of SS-tree claim the insertion uses significantly less CPU time, compared to R*-tree (5-10x less). This is mainly because of simplistic insertion and linear split algorithm, compared to quadratic split algorithm of R*-tree. When querying, the SS-tree outperforms the R*-tree by a factor of two (approximately).

2.2.4 SR-tree

The SR-tree (Katayama & Satoh 1997) is merely a combination of SS-tree and R*-tree. The authors presented an extensive comparison of MBRs and spheres properties. They defined a region in SR-tree as an intersection of MBR and sphere, gaining both advantages – a small volume of the MBR and a small diameter of the sphere. This extension should bring reasonable query performance for both window and nearest neighbor queries. The insertion and split algorithms are taken from the SS-tree and they are controlled solely by spheres. The SR-tree slightly outperforms both SS-tree and R*-tree. As the most significant inefficiency of this approach, the authors discuss storage requirements of SR-tree region, which are 1.5× larger than that of R*-tree and 3× larger than that of SS-tree.

2.2.5 X-tree

The X-tree (Berchtold et al. 1996) provides overlap-free split whenever it is possible, that is, just splits that do not lead to degeneration of the tree are allowed. Otherwise, the X-tree creates variable-sized directory nodes, so-called supernodes, to keep the hierarchy spatially compact. Since the supernodes can be quite large (which leads to partial sequential search), the X-tree could be seen as a hybrid of a linear array-like and a hierarchical R-tree-like directory. The main advantage is X-tree's performance when querying high-dimensional data, where it outperforms the R*-tree by up to two orders of magnitude.

3 The eR-tree

As many studies (Beckmann et al. 1990, Katayama & Satoh 1997) indicate, the performance of the R-tree-based structures is mostly determined by the amount of region overlaps and dead space coverage. Most of the R-tree variants try to handle this problem by revisiting the splitting algorithms (e.g., the overlap-free one (Sellis et al. 1987)) or introducing concepts like forced reinserts, which fight against the dynamic behavior of R-tree indexing. Our idea is to substitute the usually employed MBRs with arbitrarily rotated ellipsoids. Intuitively, ellipsoid, as a quadratic surface, could *cover* the data more tightly, leading to regions smaller in volume (less dead space coverage) and possibly less overlaps (i.e., smaller regions naturally produce less overlaps). To grasp our "motivation", consider Figure 2a – 10 randomly generated tuples are covered by an ellipsoid and an MBR. The volume of the ellipsoid is 0.074 and the volume of the MBR is 0.181 so the ellipsoid is 2.4× smaller in volume than the MBR.

3.1 Index Structure

The structure of eR-tree is based on the original Guttman's R-tree (Guttman 1984). The most straightforward application of ellipsoids in R-tree would substitute all MBRs with ellipsoids. However, our preliminary tests revealed the following facts:

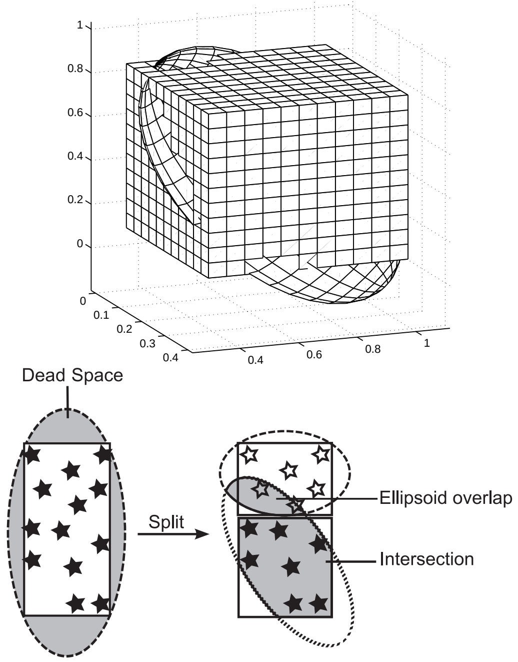


Figure 2: (a) Covering of 10 randomly generated tuples by ellipsoid and rectangle. (b) Dead space coverage and non-overlap-free split.

- The test for an ellipsoid and a query window (QW) intersection (employed in the query algorithm) is approx. 50× more expensive in terms of CPU costs than a test for MBR and QW intersection (for more details see Section 3.4).
- Most of the filtration/pruning is performed in nodes just one level above the leaf nodes; we call them *pre-leaf* nodes (and pre-leaf level).

Following the above observations and taking into account high storage requirements of ellipsoids (being quadratic with respect to the dimensionality), we decided to investigate the eR-tree variant with ellipsoids utilized only in pre-leaf nodes. The other nodes will use usual MBRs.

Even though ellipsoids cover the tuples more tightly in average, there are some situations when MBRs are superior to them:

- When splitting, the ellipsoids tend to produce more overlaps then MBRs on dense data.
- When the data distribution tends to rectangular clusters, the ellipsoids will cover more dead space.

In Figure 2b, we can notice the effects stated above – an unnecessary dead space coverage on the left, and an overlap between ellipsoids on the right (after splitting). To solve these problems, we decided to define pre-leaf node region as an *intersection* of an ellipsoid and an MBR, consider Figure 3. A similar idea is engaged in SR-tree (Katayama & Satoh 1997), where regions are defined as an intersection of a sphere and an MBR.

3.2 Ellipsoid Theory

In this section we will discuss the problem of enclosing a set of data tuples by an ellipsoid.

Definition 1 *Let an ellipsoid $\varepsilon(c, Q)$ in R^n with center in $c \in R^n$ and shape matrix $Q \in R^{nxn}$ be the set of points*

$$\epsilon(c, Q) = \{x \in R^n |\ (x - c)^T Q(x - c) \leq 1\}$$

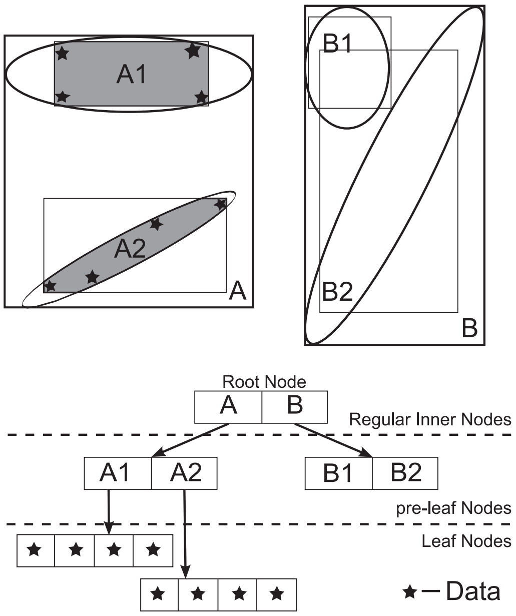


Figure 3: An eR-tree with pre-leaf regions defined as intersection of ellipsoid and MBR.

where the shape matrix Q is a symmetric positive semidefinite coefficient matrix representing some quadratic form.

The volume of an ellipsoid $\varepsilon(c, Q)$ is given by formula

Theorem 1 $Vol(\varepsilon) = \frac{\pi^{n/2}}{\Gamma(n/2+1)} \frac{1}{\sqrt{detQ}}$ *where Γ is a gamma function.*

Further details and references on proof of the theorem could be found in (Sun & M. Freund 2004). It is evident to require an ellipsoid of minimum volume to be employed by eR-tree, hence, we define the minimum volume covering ellipsoid (**MVCE**) and outline the algorithm of MVCE construction with some performance experiments.

Definition 2 *For a given set $S = \{x_1, \ldots, x_k\}$ of n-dimensional tuples we define the* Minimum Volume Covering Ellipsoid *as any ellipsoid $\varepsilon(c, Q)$ for which*

$$\forall x \in S : (x - c)^T Q(x - c) \leq 1 \qquad (containment)$$
$$\varepsilon_1(c_1, Q_1), \forall x \in S : (x - c_1)^T Q_1(x - c_1) \leq 1 \Rightarrow$$
$$Vol(\varepsilon_1) \geq Vol(\varepsilon) \qquad (min.\ volume)$$

Basically, we are aware of three distinct approaches how to construct MVCE. The first one published in early 80's is based on eigenvalue decomposition (Barnes 1982). Almost ten years later Welzl published an algorithm based on randomized iterative construction (Welzl 1991). Finally, Kchachiyan formulated the problem of computing MVCE as an optimization problem using interior-point method (Khachiyan & Todd 1993).

After careful examination of all methods of MVCE construction we decided to use the Kchachiyan optimization construction (Khachiyan & Todd 1993). The complexity of this algorithm was improved in (Todd & Yildirim 2007, Sun & M. Freund 2004, Kumar & Yildirim 2005). The latter brings the *Core sets*[1] as a byproduct. In this paper we will stick to MVCE construction described in (Moshtagh 2005).

[1] Core set is a small subset of the input tuples whose covering is "almost" same as the covering of the entire input, hence it can be used to optimize on large set of tuples.

We want to obtain the resulting ellipsoid of minimum volume and covering all the tuples from S. Thus, the formulation of MVCE is the following:

$$\begin{array}{lll} \text{minimize} & det(Q^{-1}) & (1) \\ & \text{while preserving} & \\ & (x_i - c)^T Q(x_i - c) \le 1 \qquad i = 1, \dots, |S| & \\ & Q \succ 0 & \end{array}$$

Since this problem is not a convex optimization problem, we can rewrite it, by substitution of $A = Q^{1/2}, b = Q^{1/2}c$, as:

$$\begin{array}{rll} \text{minimize} & log(det(A^{-1})) & (2) \\ \text{while preserving} & \|Ax_i - b\| \le 1 \qquad i = 1, \dots, |S| & \\ & A \succ 0 & \end{array}$$

which is a convex optimization problem, but unfortunately still difficult to solve. Luckily, the dual problem is much easier. To solve the dual problem, *lifting* of the original problem needs to be defined. This means all tuples $S = \{x_i, \dots, x_k\}$ $x_i \in R^n$ need to be moved to R^{n+1}. We can set $(x_i^l)^T = [x_i^T, 1]$ $i = 1, \dots, k$ and define $S^l = \{\pm x_1^l, \dots, \pm x_k^l\}$. The $mvce(S^l)$ will be symmetric around the origin of R^{n+1} and the $mvce(S)$ will by obtained as an intersection of $mvce(S^l)$ with the hyperplane $H = \{(x, 1) \in R^{n+1} | x \in R^n\}$. The lifted optimization is then

$$\begin{array}{rll} \text{minimize} & log(det(M^{-1})) & (3) \\ \text{while preserving} & (x_i^l)^T M x_i^l \le 1 \qquad i = 1, \dots, |S| & \\ & M \succ 0 & \end{array}$$

where M is the decision variable. Now, dual Lagrangian can be formulated and optimized. The optimization is carried out by Conditional Gradient Ascent method. The asymptotic complexity of this algorithm is *linear* in the number of tuples being covered by the ellipsoid. For further details we refer to (Moshtagh 2005) and (Kumar & Yildirim 2005). The former deals with details of optimization and solution extraction, in the latter the asymptotic complexity is derived.

As a *stopping criterion* of the optimization, the average distance of all tuples which lie outside the approximated ellipsoid to its boundary is used. While we need to have all the tuples strictly included in the resulting ellipsoid, we are performing a post-processing on the ellipsoid obtained from the approximation. In the post-processing, we locate the tuple which lies furthest from the boundary of the ellipsoid and then we scale the ellipsoid, so this tuple and all others lie inside. In the result, the larger stopping criterion, the faster computation of MVCE approximation but also larger volume. For example, a value 0.1 of the stopping criterion means the furthest tuple is at most 0.1 distant from the surface (considering unitary radius of an ellipsoid).

We conducted some small performance experiments to evaluate the adequacy of this constructing technique of MVCE for our purposes. In Table 1, we can observe the impact of the size of S and the dimensionality on the time needed to construct MVCE. The stopping criterion of the algorithm was set to 0.01.

The rest of experiments were focused on the stopping criterion, considering 3-dimensional space. In Table 2, we can notice the efficiency of the algorithm depends mostly on this parameter. The experimental results prove this algorithm meets the requirements for our purposes with stopping criterion at most 0.01.

SetSize	dim. 2	dim. 5	dim. 10	dim. 15	dim. 20
10	0.003	0.002	0.000	0.002	0.000
20	0.009	0.008	0.002	0.002	0.002
40	0.008	0.006	0.008	0.005	0.002
60	0.016	0.013	0.017	0.003	0.008
80	0.019	0.020	0.017	0.013	0.011
100	0.022	0.025	0.028	0.023	0.019
140	0.036	0.051	0.052	0.055	0.033
160	0.047	0.054	0.094	0.072	0.053
200	0.069	0.081	0.155	0.117	0.094
220	0.089	0.103	0.184	0.148	0.114
250	0.119	0.105	0.156	0.131	0.125
280	0.334	0.345	0.388	0.325	0.250
300	0.366	0.411	0.437	0.383	0.267
350	0.522	0.555	0.642	0.559	0.411
400	0.701	0.769	0.800	0.705	0.578
450	0.892	0.919	1.059	0.970	0.781
500	1.133	1.167	1.295	1.252	1.012

Table 1: Time (seconds) required to construct MVCE as a parameter of dimension (2, 5, 10, 15, 20) and set size.

SetSize	0.1	0.01	0.001
10	0.084	0.003	0.053
20	0.003	0.003	0.060
40	0.003	0.012	0.069
60	0.009	0.013	0.128
80	0.003	0.019	0.156
100	0.006	0.022	0.163
140	0.000	0.041	0.362
160	0.009	0.047	0.716
200	0.013	0.072	0.588
220	0.016	0.075	0.744
250	0.009	0.081	0.850
280	0.031	0.316	2.853
300	0.034	0.372	3.363
350	0.056	0.494	4.650
400	0.062	0.634	6.613
450	0.078	0.850	8.169
500	0.103	1.044	10.013

Table 2: Time (seconds) required to construct MVCE as a parameter of stopping criterion (0.1, 0.01, 0.001) and set size.

3.3 Indexing

The algorithm of insertion is described in many available literature, e.g., the original (Guttman 1984), therefore we emphasize just our modifications. First, in Algorithm 1 see the procedure which locates the appropriate leaf for new entry accommodation.

Algorithm 1 ChooseLeaf

```
Require: P – tuple to be inserted
CurrentNode = RootNode
while CurrentNode is not leaf node do
    if CurrentNode is pre-leaf node then
        CurrentNode = getSubBranchEll(P) {return the child of current node, which region is closest to the P. The distance of ellipsoid and P is considered.}
    else
        CurrentNode = getSubBranchMBR(P) {returns the child of current node, which MBR region being enlarged by P produces smallest enlargement. If enlargement should cause overlap with other regions of CurrentNode and there exists region of CurrentNode which won't produce overlap after being enlarged by P, than the sub-branch of this non-overlap-producing region is chosen. I.e. we prefer rather to cover more dead space, than to produce overlaps.}
    end if
end while
return CurrentNode
```

The leaf splitting strategy significantly determines the performance of R-tree. In Algorithm 2, see our splitting algorithm, which tries to separate data tuples based on the dimension with the maximum variance. First, we choose the dimension in which the data is spread the most. Then we sort the data according to values in this dimension and find the split position.

Algorithm 2 MinVar Split

```
Require: $\{p_1, \ldots, p_k\}$ set of entries to be split
$maxVar = 0.0$
for $i = 0$ to $dimension$ do
    $var = computeVarianceInDimension(i, \{p_1, \ldots, p_k\})$
    if $var > maxVar$ then
        $maxVar = var$; $splitDimension = i$
    end if
end for
$\{p_1', \ldots, p_k'\} = sortByDimension(splitDimension, \{p_1, \ldots, p_k\})$
    {so that $\{p_1' \leq p_2' \leq \ldots \leq p_k'\}$ holds in $splitDimension$}
$diff = p_k'[splitDimenstion] - p_1'[splitDimenstion]$
for $i = 2$ to $k - 1$ do
    if $p_1'[splitDimension] + diff/2 < p_i'[splitDimension]$ then
        $splitOrder = i$
        $break$
    end if
end for
return $\{p_1', \ldots, p_{splitOrder}'\}, \{p_{splitOrder+1}', \ldots, p_k'\}$
```

3.4 Querying

The querying algorithms are also described in many available literature, e.g., in the original (Guttman 1984). Therefore, we will focus just on the ellipsoid and query window intersection test. The problem of deciding whether a query window $QW(ql, qh)$, $ql, qh \in R^n$ and an ellipsoid $\varepsilon(c, Q)$ intersect, can be formalized as a convex optimization problem of form:

$$\begin{array}{lll} \text{minimize} & (x-c)^T Q(x-c) & \\ \text{while preserving} & ql \leq x_i \leq qh_i & i = 1, \ldots, n \end{array} \tag{4}$$

where the objective variable is x. If $x \leq 1$, then QW and ellipsoid intersect, otherwise their intersection is empty. We have conducted some experiments to compare the speed of intersection test of MBR×QW and Ellipsoid×QW, see Table 3. The optimization was carried out with the *loqo*[2] solver. We can observe, that Ellipsoid×QW test is significantly slower than MBR×QW test, however, it is still significantly faster than a seek to secondary storage.

	dim.=2	=4	=6	=8	=10
MBR Ellipsoid	0.000312 0.0403	0.0005 0.0445	0.000812 0.0458	0.000686 0.0493	0.000812 0.0602
	dim.=12	=14	=16	=18	=20
MBR Ellipsoid	0.00112 0.0676	0.00119 0.106	0.0012 0.23	0.00137 0.522	0.00194 1.01

Table 3: Time (milliseconds) required to evaluate whether Ellipsoid/MBR and Query window intersect, depending on growing dimensionality.

4 Experimental Results

To evaluate the eR-tree, we have used two datasets:

Clustered dataset is a synthetic dataset and contains sets of $1 \cdot 10^4 - 2 \cdot 10^6$ tuples, where each cluster is composed of 1,000 uniformly distributed tuples. Individual tuples were randomized prior to storing (so they are not indexed "cluster by cluster", but randomly). We will refer to this dataset as to **SCU dataset**.

Real dataset was generated from the well-known IMDB database[3]. The 5-dim. tuples represent records `[movie_id, director_id, movie_genre_id, movie_production_year, movie_kind_id]`. The entire dataset consists of 392,689 records. The cardinality of individual attributes can be found in Table 4. We will refer to this dataset as to **IMDB dataset**.

Attribute	MIN	MAX
movie_id	1	1137185
director_id	31	1824450
movie_genre_id	2	29
movie_production_year	1519	2013
movie_kind_id_id	1	7

Table 4: Cardinality of individual attributes in the IMDB dataset.

We have generated various query sets to evaluate the eR-tree querying performance. Each query set consisted of 1,000 individual queries, and the performance results for one query set are presented as the average of all 1,000 trials. A query set is characterized by its *selectivity*, i.e. the number of hits each individual query returns. The querying tests are performed with query sets of absolute selectivity 3 and 50 tuples.

The eR-tree was implemented in C++, while all the experiments were carried out on 2.2GHz AMD Turion Processor with 1GB of RAM and 5400rpm hard drive on WinXP with NTFS file-system. The source code was compiled with VC 8.0.

For a comparison, the results for eR-tree are presented along with results for R-tree (being a baseline), however, we used an R-tree version extending the original Guttman's version (Guttman 1984) by small improvements. These improvements try to avoid region overlaps at the cost of higher dead space coverage[4], and relax the minimum required node utilization below 50%.

[2] Available at `http://www.princeton.edu/~rvdb/loqo/`

[3] Available at `www.imdb.com`

[4] As discussed in (Beckmann et al. 1990), the regions overlaps have greater impact on the search performance than the dead space

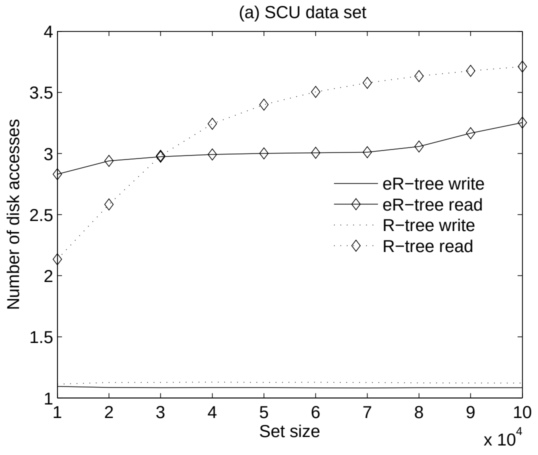


Figure 4: Insertion costs as a parameter of a set size – SCU dataset

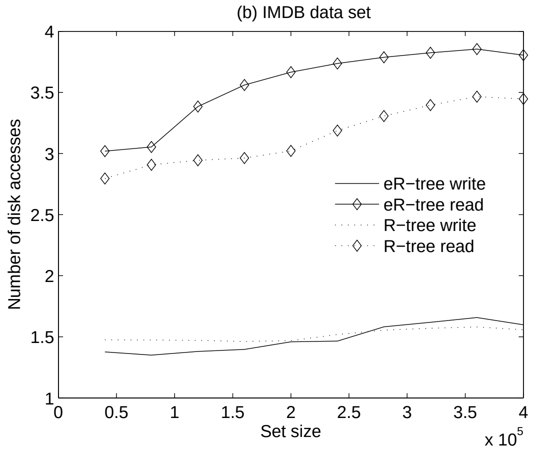


Figure 5: Insertion costs as a parameter of a set size – IMDB dataset

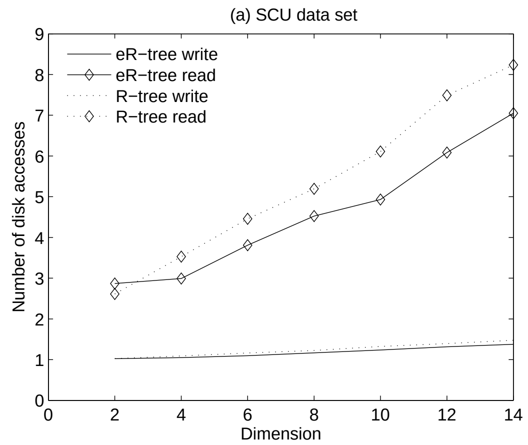


Figure 6: Insertion costs as a parameter of dimension: SCU dataset

4.1 Indexing

To meet the page size of the disk drive, the size of all the nodes of eR-tree and R-tree was set to 8KB. The fanout of eR-tree nodes can be found in Table 5. The heights of the trees for SCU and IMDB datasets can be found in Table 6. The columns marked with (*) represent the theoretical tree heights with 100% utilized nodes.

		Dimension					
		3	4	5	8	10	12
Fanout	inner node	146	113	93	60	48	40
	pre-leaf node	39	26	19	9	6	4
	leaf node	255	204	170	113	92	78

Table 5: Node fanout

		Height			
		eR	R	eR(*)	R(*)
Data set	SCU	5	8	3	2
	IMDB	3	3	3	2

Table 6: Tree heights

On the SCU dataset the eR-tree's index size was 3.62 MB, compared to 5.2 MB for R-tree (average node utilization of eR-tree reached 69.7%, while R-tree reached only 50.2%). On the real IMDB dataset the eR-tree index size was 18.2 MB, compared to R-tree's 15 MB (having 57.8% and 63.1% average node utilization, respectively).

In Figures 4, 5, see the insertion costs for a single tuple in terms of the number of required reads and writes. In Figure 4 the result for the SCU dataset with size varying from $1 \cdot 10^4$ to $1 \cdot 10^5$ is presented. In Figure 5, see an analogical experiment for the IMDB dataset.

Next, we have observed the insertion costs as a parameter of dimensionality. In Figure 6 see the measurements for the SCU dataset and Figure 7 for the IMDB dataset. For a more detailed comparison, see (Danko n.d.).

4.2 Querying

In Figures 8-11, see the querying results for SCU and IMDB dataset, with respect to varying set size. On the left y-axis the total I/Os required to evaluate a query are plotted, on the right y-axis the number of searched leaf regions during evaluation is plotted. It can be seen, eR-tree on the SCU dataset clearly outperforms the R-tree – the eR-tree requires only 72% of I/O operation required by the R-tree. The difference is even more noticable on query set with selectivity 50. On the IMDB dataset, the R-tree slightly outperforms eR-tree with selectivity of the query set equal to 3, however, with query set of selectivity 50 the eR-tree outperforms the R-tree significantly.

How about the impact of a dimensionality on the querying performance? This question is answered in Figures 12-15, where the SCU and IMDB datasets with varying dimensions are queried. We can notice, that eR-tree gains better results for lower dimensionalities (i.e., less than 10). In higher dimensions, the test for ellipsoid and QW intersection (recall, that it is an optimization problem) becomes more expensive, because there are "more" directions in which the optimization can go. To avoid these situations, if the intersection procedure needs 60 iterations or more, the optimization is stopped and the ellipsoid and QW are claimed to be intersected (to avoid false dismissals). As a consequence, the ellipsoid volume is overestimated (leading to more frequent intersections with QW), so extra leaf nodes need to be searched. In this paper we are not presenting the CPU time, because it could be misleading as it highly depends on a level of the intersection code optimization. However, using a general-purpose solver in our experiment, there were some situations where the eR-tree outperformed R-tree also in terms of CPU time (e.g., in low dimen-

coverage; therefore, such an observation should slightly improve the performance of the original R-tree.

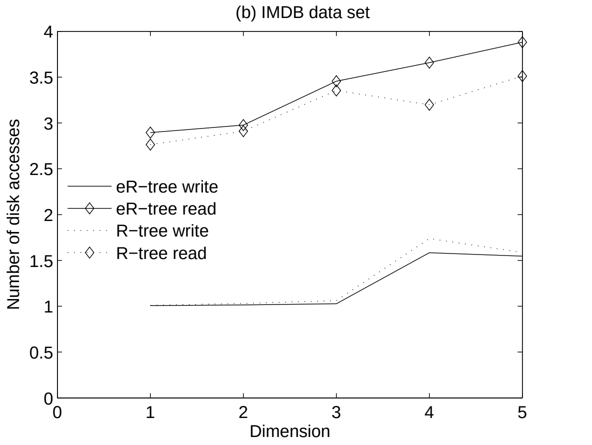


Figure 7: Insertion costs as a parameter of dimension: IMDB dataset

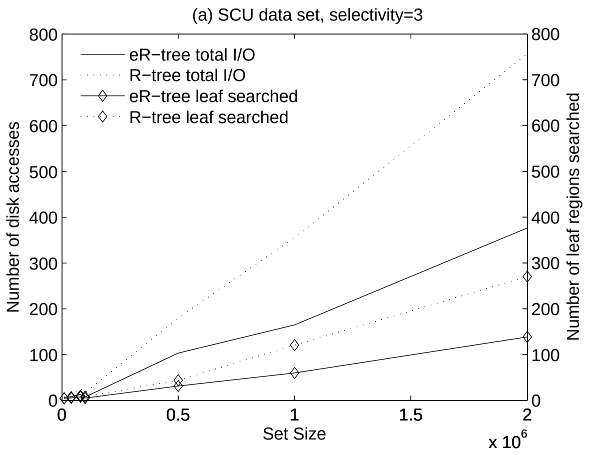


Figure 8: Querying cost as a parameter of a set size. SCU dataset, selectivity=3

sions on the IMDB data set).

5 Conclusions

In this paper, we have proposed the eR-tree, a variant of R-tree which employs minimum volume covering ellipsoids instead of usual minimum bounding rectangles. The experimental results shown that the construction of ellipsoids is efficient enough to be incorporated into R-tree-like index structures. We have found out that eR-tree significantly outperforms R-tree in terms of I/O on sparse clustered data, where ellipsoidal regions are superior to minimum bounding rectangles, because they tend to cover less dead space and produce less overlaps. However, on dense data the advantage of ellipsoids is suppressed. In the future work the attention should be paid to optimization of the ellipsoid and query window intersection test and also the splitting algorithms. The further investigations should also favor of *sparse data*, where the ellipsoids outperform minimum bounding rectangles. In particular, besides native engines, XML databases are often transformed into sparsely distributed multi-dimensional tuples (Mlynkova & Pokorny 2008, Krátký et al. October 27-31, 2002). Here the eR-tree-based indexing could be beneficial for processing of an XPath or XQuery statement.

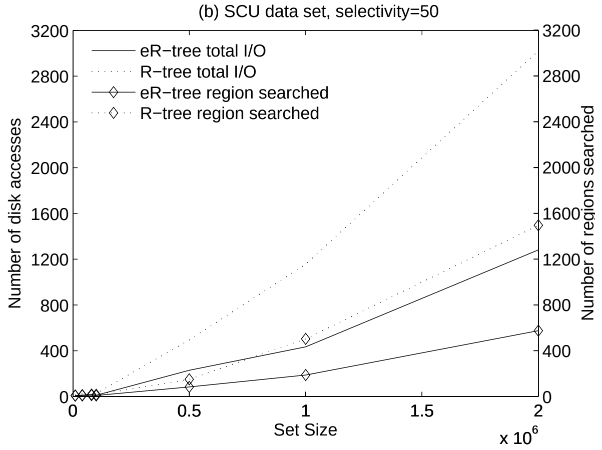


Figure 9: Querying cost as a parameter of a set size. SCU dataset, selectivity=50

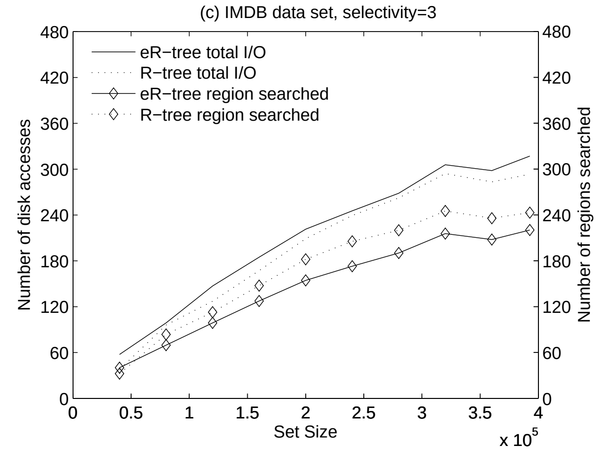


Figure 10: Querying cost as a parameter of a set size. IMDB dataset, selectivity=3

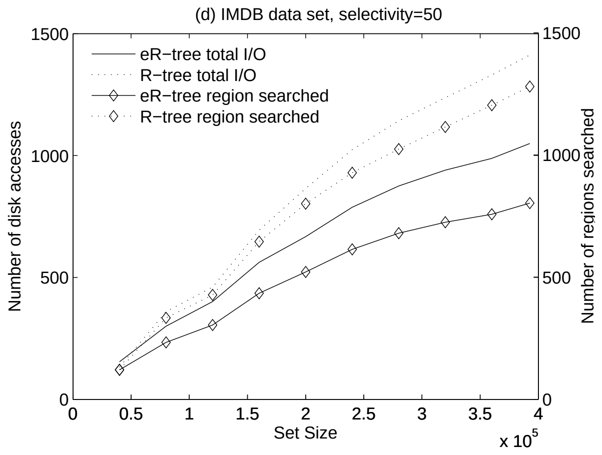


Figure 11: Querying cost as a parameter of a set size. IMDB dataset, selectivity=50

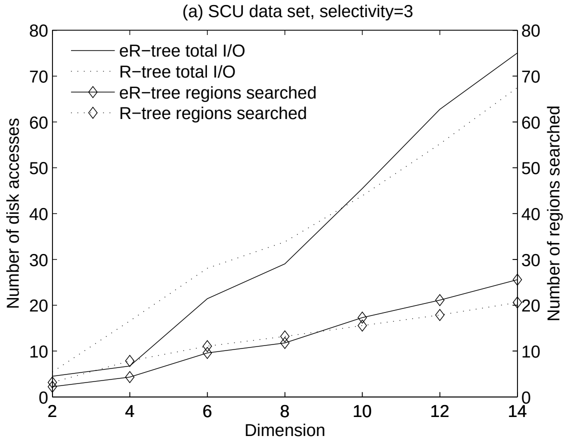


Figure 12: Querying cost as a parameter of a dimension. SCU dataset, selectivity=3

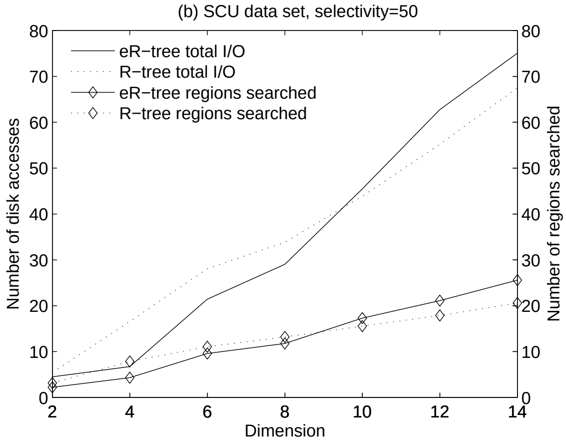


Figure 13: Querying cost as a parameter of a dimension. SCU dataset, selectivity=50

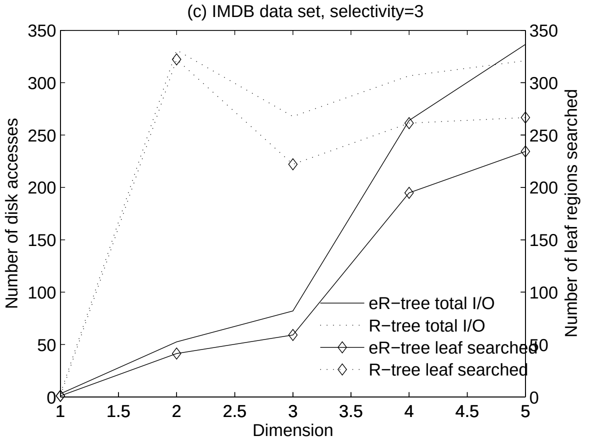


Figure 14: Querying cost as a parameter of a dimension. IMDB dataset, selectivity=3

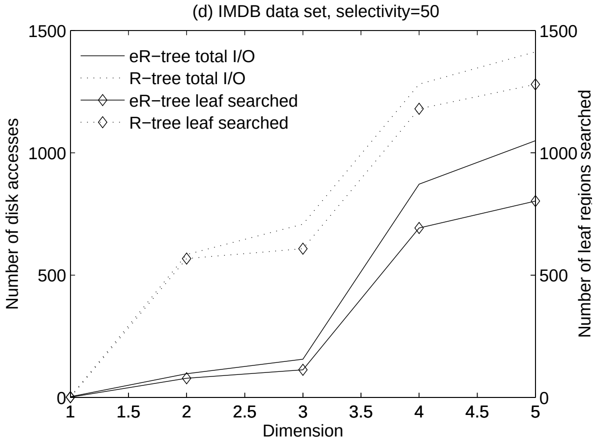


Figure 15: Querying cost as a parameter of a dimension. IMDB dataset, selectivity=50

Acknowledgments

This research has been partially supported by Czech grants: GAČR 201/06/0756 and Institutional research plan number MSM0021620838.

References

Barnes, E. (1982), 'An Algorithm for Separating Patterns by Ellipsoids', *Image Processing and Pattern Recognition* **26**(6), 759.

Beckmann, N., Kriegel, H.-P., Schneider, R. & Seeger, B. (1990), The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles, *in* H. Garcia-Molina & H. V. Jagadish, eds, 'Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23-25, 1990', ACM Press, pp. 322–331.

Berchtold, S., Keim, D. A. & Kriegel, H.-P. (1996), The x-tree : An index structure for high-dimensional data, *in* T. M. Vijayaraman, A. P. Buchmann, C. Mohan & N. L. Sarda, eds, 'VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India', Morgan Kaufmann, pp. 28–39.

Danko, O. (n.d.), 'Elliptic Indexing of Multidimensional Databases, master thesis, Charles University in Prague, `siret.ms.mff.cuni.cz/skopal/diplomky/danko.pdf`, 2008'.

Guttman, A. (1984), R-Trees: A Dynamic Index Structure for Spatial Searching, *in* B. Yormark, ed., 'SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984', ACM Press, pp. 47–57.

Katayama, N. & Satoh, S. (1997), The SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries, *in* J. Peckham, ed., 'SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA', ACM Press, pp. 369–380.

Khachiyan, L. G. & Todd, M. J. (1993), 'On the complexity of approximating the maximal inscribed ellipsoid for a polytope', *Mathematical Programming: Series A and B* **61**, 137 – 159.

Krátký, M., Pokorný, J., Skopal, T. & Snášel, V. (October 27-31, 2002), The Geometric Framework for Exact and Similarity Querying XML Data, *in* 'Proceedings of First EurAsian Conferences, EurAsia-ICT 2002, Shiraz, Iran', Springer-Verlag LNCS 2510.

Kumar, P. & Yildirim, E. A. (2005), 'Approximate minimum volume enclosing ellipsoids using core sets', *Journal of Optimization Theory and Applications* **1**, 1–21.

Mlynkova, I. & Pokorny, J. (2008), Usermap : an adaptive enhancing of user-driven xml-to-relational mapping strategies, *in* A. Fekete & X. Lin, eds, 'Nineteenth Australasian Database Conference (ADC 2008)', Vol. 75 of *CRPIT*, ACS, Wollongong, NSW, Australia, pp. 165–174.

Moshtagh, N. (2005), 'Minimum volume enclosing ellipsoid', *Convex Optimization* .

Sellis, T. K., Roussopoulos, N. & Faloutsos, C. (1987), The R+-Tree: A Dynamic Index for Multi-Dimensional Objects, *in* 'VLDB', pp. 507–518.

Sun, P. & M. Freund, R. (2004), 'Computation of Minimum Volume Covering Ellipsoids', *Discrete Applied Mathematics* **52**, 690–706.

Todd, M. J. & Yildirim, E. A. (2007), 'On khachiyan's algorithm for the computation of minimum-volume enclosing ellipsoids', *Discrete Applied Mathematics* **155**, 1731–1744.

Welzl, E. (1991), 'Smallest enclosing disks (balls and ellipsoids)', *Lecture Notes in Computer Science* pp. 359 – 370.

White, D. & Jain., R. (1996), Similarity indexing with the SS-tree, *in* 'In Proc. 12th International Conference on Data Engineering (ICDE'96)', IEEE CS Press, pp. 516–523.

Efficient XQuery Join Processing in Publish/Subscribe Systems

Ryan H. Choi[1,2] Raymond K. Wong[1,2]

[1] The University of New South Wales, Sydney, NSW, Australia
[2] National ICT Australia, Sydney, NSW, Australia
Email: {ryanc,wong}@cse.unsw.edu.au

Abstract

Efficient XML filtering has been a fundamental technique in recent Web service and XML publish/subscribe applications. In this paper, we consider the problem of filtering a continuous stream of XML data against a large number of XQuery queries that contain multiple inter-document value-based join operations in their `where` clauses. To perform efficient join operations, the path expressions from these queries are extracted and organized in a way that multiple path expressions can be joined simultaneously. The join operations are then pipelined to minimize the number of join operations and to share any intermediate join results as much as possible. Our system operates on top of many currently available XPath filtering engines as an add-on module to extend their features to support queries with join operations. Experiments show that our proposal is efficient and scalable.

Keywords: XML publish/subscribe, XML data stream, XML query processing

1 Introduction

XML has become the standard for data representation and exchange between large scale Web service applications on the Internet. We consider a Web service application that receives streams of XML messages from various data sources on the Internet, and forwards these messages to subscribed users or other applications. This type of application is called an XML publish/subscribe (pub/sub) system. One key feature of such an application is to support a large number of user subscriptions, and efficiently process XML messages coming from streams in real time. Furthermore, it is important that an XML pub/sub system is expressive enough to process complex subscriptions. Recently, there have been several research efforts on building scalable and expressive XML pub/sub systems (Chan et al. 2002, Diao et al. 2003, Gupta & Suciu 2003, Onizuka 2003, Rao & Moon 2004, Uchiyama et al. 2005, Kwon et al. 2005). In these systems, user subscriptions are expressed in XPath (Clark & DeRose 1999) queries. We use the term *queries* and *documents* to refer to user subscriptions and messages in this paper, respectively. Many previous works (Chan et al. 2002, Diao et al. 2003, Gupta & Suciu 2003, Onizuka 2003, Rao & Moon 2004, Uchiyama et al. 2005, Kwon et al. 2005) focus on how to efficiently report the set of matching query IDs for each streaming XML document. However, one feature that these previous works commonly lacks is to process queries that join multiple documents. In addition, these works are also limited in a way that, only the matching query IDs are reported—they cannot return the set of matching elements for each matching query. In this paper, we present how to process queries that contain inter-document value-based join operations efficiently. At the same time, we also present how to return all matching elements for each matching query.

Supporting queries that join multiple elements coming from multiple streams is important, since such technique allows us to deliver richer information than existing pub/sub systems. For example, a sudden price drop of a share can be detected, and a financial section of an online newspaper that contains news articles about the same share can be delivered by existing systems. However, users might be interested in what caused the sudden price drop of this share. To do that, existing systems require two queries from each user, one for detecting the price drop of the share, and the other is for selecting news articles about that share. Then, these two queries are postprocessed, and when there exists such matching news articles about that share, each subscribed user is notified. After that, the entire financial sections of the newspaper are optionally delivered. If a user prefers to receive only the related news articles about that share, this cannot be achieved using existing systems. This is because these systems are designed to be used as XML routers or brokers, where only the delivery of the complete documents to downstream routers are considered. Recent work by Hong et al. (2007) provides a partial solution to this problem. In conjunction with an XPath processor, it supports queries that join two documents. However, it is still not suitable for this situation, as it only reports matching query IDs, and can only forward the entire documents to users. Moreover, their join operations are limited in a way that only the leaves of a document can be joined. Furthermore, their technique is designed to process very small documents in size, which are not suitable for larger documents. On the other hand, our system provides a solution to this particular problem. Any nodes from documents can be joined with each other, and large documents in size can be efficiently processed. In addition, we return the complete set of matching elements for each matching query along with its ID, and they can be forwarded to users.

We consider the case where subscriptions are written in XQuery queries, and evaluate these queries against streaming XML documents whose sizes are small enough to fit into memory. In addition, the structures of streaming documents are already given to each pub/sub system. This is done during the initial handshake period between upstream and down-

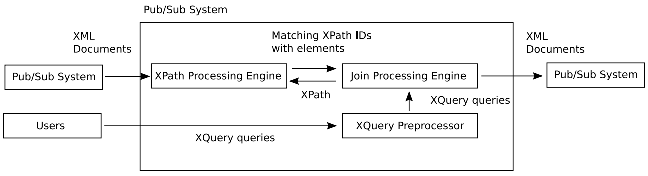


Figure 1: System Architecture

stream pub/sub systems. Once documents have been received from upstreaming pub/sub systems, n documents are stored temporarily in the buffer of size $b_{|n|}$, and are removed in a FIFO fashion when either the buffer becomes full, or after a certain period of time t_c. Join operations are performed between the documents in the buffer whose positions are in (1) between b_0 and b_i; or (2) their arrival times are in between t_i and t_j, where $t_i < t_j$. We currently support a subset of `for-where-return` clauses of XQuery. The description for each clause is as follows. A query contains multiple `for` clauses. A `for` clause expresses a node that is to be joined with a node in other `for` clause. The nodes in two `for` clauses can be from two different streams. A `where` clause expresses a list of conjunctive join operations between the nodes in `for` clauses. When a node is compared for equality, `text()` value is used if that node is a leaf node. If it is a non-leaf node, either `text()` or `string()` value can be used. A `return` clause expresses what node should be returned when a query matches the conditions in the `where` clause. Supporting `let` clauses is left as a part of our future work.

Figure 1 shows the architecture of our system. A pub/sub system sends an XML document of type t to multiple downstream pub/sub systems that have subscribed to receive documents of the same type. Upon receiving the document, a pub/sub system processes its subscriptions against it, and returns a set of subtrees in the document. This set of subtrees becomes a set of new documents, and are sent to downstream pub/sub systems that have subscribed to receive documents of the same type. Similar process is repeated at every pub/sub system.

Our pub/sub system consists of three major components. XPath Processing Engine is an XPath processor that can return matching query IDs along with all matching nodes for all matching queries for each streaming XML document. We use our previous work (Choi & Wong 2009) as XPath Processing Engine, as it satisfies the above conditions and performs efficiently. XQuery Preprocessor is used to normalize user subscriptions written in XQuery. Transformed queries are then sent to Join Processing Engine. Join Processing Engine processes queries with join operators, and it works in conjunction with XPath Processing Engine. When queries are received from XQuery Preprocessor, the XPath expressions in `for` clauses of XQuery queries are extracted, and the expressions with '`//`' and '`*`' are expanded using the document structures that have been registered initially. These expanded XPath expressions are then transferred to XPath Processing Engine. At the same time, Join Processing Engine organizes the expanded XPath expressions in a tree structure. When an XML document arrives, XPath Processing Engine returns the matching XPath IDs along with matching nodes, and they are transferred to Join Processing Engine. Once Join Processing Engine has processed join operations, all matching nodes for each matching XQuery query are reported, and they are sent to downstream pub/sub systems.

This paper is organized as follows. Section 2 presents related work in the area of processing XPath and XQuery queries on streaming XML documents. Section 3 describes our technique to process join operators in XQuery queries. Section 4 presents our experiment results for our technique. Finally, we conclude our paper in Section 5.

2 Related Work

There have been many works done in the context of processing XPath queries against streaming XML documents. Diao et al. (2004) introduce an overall architecture of a scalable pub/sub system. XTrie (Chan et al. 2002), YFilter (Diao et al. 2003), XPush (Gupta & Suciu 2003), Onizuka (2003), PRIX (Rao & Moon 2004), Uchiyama et al. (2005) and FiST (Kwon et al. 2005) present how a large number of XPath queries can be evaluated against a streaming XML document. While all these works return the set of matching XPath IDs for each document, they do not return matching nodes for matching queries, and they do not support queries with join operations. Furthermore, it is not clear how these features can be implemented.

Approaches by Olteanu et al. (2004), FluX (Koch et al. 2004), Li & Agrawal (2005), XFPro (Huo et al. 2006) and Gou & Chirkova (2007) process a single XPath and/or XQuery query against a large XML document in size. Barton et al. (2003) present how a single XPath query with both forward and backward axes can be processed for each streaming document. Chen et al. (2006) use TwigM, which is an extension of the multi-stack approach in TwigStack (Bruno et al. 2002), for a compact representation of candidate elements during query processing to decrease the overall memory usage. However, non of these works support queries with join operations.

Diao & Franklin (2003) extend YFilter such that it can return matching elements for matching queries. In their system, only XQuery queries with conjunctive predicates of the form $e_1/text() = constant$ are supported, whereas we support conjunctive predicates of the form $e_1/text() = e_2/text()$ in `where` clauses, where e_1 and e_2 can be from two different documents. XSQ (Peng & Chawathe 2005) builds an NFA for an XPath query that contains nested paths, and maintains its own buffer to store potential matching elements. This allows them to return all matching elements for a given query. However, only one query can be processed at a time. The work by Hong et al. (2007) is the most related work to our paper. It uses a customized query language to define queries with join operators between elements. In its compilation phase, the system uses Query Templates to group queries in a way that, each group contains queries with similar join statements. In the runtime, it uses a relational database to join elements. Similar to our system, it works with an existing XPath processor as an add-on module to support queries with join operators. However, their template-based approach only works for the queries that join two leaves of documents, and unlike ours, queries that join non-leaves of documents are not supported. In addition, their approach is designed to process relatively very small documents in size (i.e., documents with 3 levels deep with 16 leaves as shown in their experiments). This is because the number of Query Templates rapidly increases as the number of leaves per document increases. Due to the same reason, their approach does not scale for the queries that join elements from multiple documents. Similar patterns are observed when the number of join operations per query increases. In our experiments, we used documents, each of which contained a few hundred leaves, and we could still provide better performance. Finally, unlike ours, they only report matching query IDs, and cannot return matching elements for each matching query for each document. This feature is important in Web service applications, as it allows them to forward only the parts of documents in which the subscribers are interested.

There are some works that use XML algebras to optimize XPath and XQuery processing. Barta et al. (2005) use document summaries to calculate heuristics and statistics, and process (nested) XPath queries with that information. Gottlob et al. (2005) define and evaluate a subset of XPath queries called Core XPath, and they can process queries with aggregates and user defined functions. However, the approaches in both works are not suitable for streaming documents, as they need multiple passes of documents. In addition, they are designed to process a small set of queries against a large XML document in size. Nevertheless, they have introduced several optimization techniques that can be integrated to our system, and such work is left as a part of our future work.

Unlike all pub/sub systems above, Boncz et al. (2006) and Grust et al. (2007) process XPath and XQuery queries by building processing engines that operate on top of existing RDBMS. Boncz et al. (2006) translate XQuery queries into basic relational algebras, apply query optimizations, and evaluate queries against a large set of documents. Grust et al. (2007) use Range Encodings to preserve the tree structure of XML, and use a B-Tree to optimize and evaluate XPath queries. Systems in this category support (almost) full features of XPath/XQuery including updates. However, they are not suitable for pub/sub systems, as they process queries individually.

3 Methods

This section presents our technique and consists of five parts. The first part shows how queries are normalized prior to processing them. The second part shows how queries are prepared. The third part presents a data structure that is used to process join operations, and how to build it. The fourth part presents how join operations are processed using the data structure against a streaming document. The last part presents an optimization technique, which improves the overall performance of join operations. We illustrate our technique with running examples.

3.1 Rewriting Queries

```
for $z in docType("nasa3")//ref
for $y in docType("nasa2")//ref//name
for $x in docType("nasa1")//journal/name
for $w in docType("nasa4")//astroObject/name
where $z//other/title=$y AND $y=$x
return $y
```

(a) Q1: A query written by a user

```
for $x in docType("nasa1")//journal/name
for $y in docType("nasa2")//ref//name
for $z in docType("nasa3")//ref//other/title
where $x=$y AND $y=$z
return $y
```

(b) Q1′: A rewritten query

Figure 2: Rewriting a query

Since XQuery queries could be poorly written or auto-generated, before queries are registered to the pub/sub system, they are preprocessed such that all queries are in the similar format. First, we remove `for` clauses whose variables do not participate in neither `where` nor `return` clauses. Second, we rewrite any join operations and their path expressions in `for` clauses in a way that join operations in `where` clauses contain only variables without any path expressions. Third, we rearrange `for` clauses according to the names of document types. Lastly, a constraint table similar to Diao et al. (2003) is built using value constraints in `where` clauses. The main idea is to process constraints after finding structurally matching queries.

Streaming documents have the same document type if they share the same document structure, and are coming from the same stream. To obtain documents of the same type, we use a user defined function `docType()` in our queries. Figure 2 shows an example of how our system rewrites a query in another format. In this example, the `for` clause with `w` variable is removed, and both `$z//other/name` and `//ref` in `where` and `for` clauses are rewritten, respectively. Finally, `for` clauses are rearranged in ascending order of the types of NASA documents.

3.2 Preparing Queries

```
for $x in docType("nasa1")//src//*/year
for $y in docType("nasa2")//history//year
for $z in docType("nasa3")//revisions//yr
where $x=$y AND $y=$z AND $x/text()="1990"
return $x
```

(a) Q2

```
for $x in docType("nasa1")//src//*/year
for $y in docType("nasa2")//history//year
for $z in docType("nasa3")//journal//*/yr
where $x=$y AND $y=$z AND $x/text()="2000"
return $y
```

(b) Q3

```
for $x in docType("nasa1")//src//*/year
for $y in docType("nasa2")//history//year
where $x=$y
return $y
```

(c) Q4

Figure 3: Query examples

The rewritten queries are processed in order to compactly represent them in our system. Figure 3 shows some additional preprocessed queries that we use as running examples in this paper. The process is as follows. First, for each rewritten query, we extract all `for` clauses. For each path expression in a `for` clause, we assign a *PathID*, and store the document type *DocType* from which that path expression is extracted. For example, the path expression, `//src//*/year` and the document type, `nasa1` are extracted from the first `for` clause in Q2, and the path expression is given an $PathID = 2$. We repeat the process for all queries and collect all unique path expressions for each document type. These path expressions are then stored in a multi-hashtable.

Second, all paths with '`//`' or '`*`' are expanded so that we can efficiently evaluate them in a deterministic way in runtime (Onizuka 2003). They are expanded using a document structure of streaming documents, and we name such a structure *Structure Index*. Figure 4 shows an example of Structure Index. It is similar to DataGuide (Goldman & Widom 1997) and ViST (Wang et al. 2003), but Structure Index is used to extract data structures of documents in order to preprocess and expand queries, whereas DataGuide and ViST are used to index data to improve query processing. Structure Index is generated

```
 1: <dataset>
 2:   <history>
 3:     <revisions><year/></revisions>
 4:   </history>
 5:   <ref>
 6:     <other><name/></other>
 7:     <src>
 8:       <journal>
 9:         <date><year/></date>
10:         <name/>
11:       </journal>
12:     </src>
13:   </ref>
14: </dataset>
```

Figure 4: Structure Index

from a set of training documents that represent various structures of streaming documents. Table 1 shows expanded path expressions generated from the queries in Figure 2(b) and 3 using Structure Index shown in Figure 4. In this example, documents of type `nasa1` and `nasa2` share the same Structure Index. Structure Index for documents of type `nasa3` is similar, and can be obtained by replacing `year` and `name` nodes on line 3, 6, 9 and 10 with `yr` and `title`, respectively. It is omitted due to limited space.

Table 1: Expanded path expressions

DocType	PathID	XPath Path Expression
nasa1	1	/dataset/ref/src/journal/name
	2	/dataset/ref/src/date/year
nasa2	1-1	/dataset/ref/other/name
	1-2	/dataset/ref/src/journal/name
	2	/dataset/history/revisions/year
nasa3	1	/dataset/ref/other/title
	2	/dataset/history/revisions/yr
	3	/dataset/ref/src/journal/date/yr

Third, for each query, we replace all path expressions in `for` clauses with *PathID*s that are obtained from the multi-hashtable. A path expression is internally represented by a (*DocType*, *PathID*) pair. A query is represented by a list of (*DocType*, *PathID*) pairs along with the `return` path expression. Table 2 shows how queries in Figure 2(b) and 3 are stored in our system. This representation of queries are then used to build a tree called Join Tree. Lastly, the path expressions in Table 1 are now sent to an XPath processing engine as input at this stage.

Table 2: Query representations

QID	Path Representation	Return Path
1	(nasa1,1)(nasa2,1-1)(nasa3,1)	(nasa2,1-1)
1	(nasa1,1)(nasa2,1-2)(nasa3,1)	(nasa2,1-2)
2	(nasa1,2)(nasa2,2)(nasa3,2)	(nasa1,2)
3	(nasa1,2)(nasa2,2)(nasa3,3)	(nasa2,2)
4	(nasa1,2)(nasa2,2)	(nasa2,2)

3.3 Building Join Tree

The query representations generated in the previous section are rearranged in a tree structure in order to efficiently evaluate join operations. We name this tree structure *Join Tree*. Figure 5 shows an example of Join Tree built from the query representations shown in Table 2, and Algorithm 1 outlines how Join Tree is built.

Each node in Join Tree represents a path expression between the root and a node in an XML document of type *DocType*. For example, a node `(nasa3,2)` represents a path expression whose $PathID = 2$ in a document of type `nasa3`. In addition, a Join Tree node contains a mapping of $\{DocType \rightarrow \{QID(PathID)\}\}$. In Figure 5, empty mappings associated with Join Tree nodes are omitted for simplicity. The query ID in a mapping associated with a Join Tree node v indicates that, the query is evaluated by joining all path expressions represented by Join Tree nodes that are along the path between the root and v. In addition, this mapping is also used to find which path of a query should be processed next in order to return matching elements. For example, in Figure 5, $\{nasa1 \rightarrow \{q2(2)\}\}$ represents that if a join operation that joins `(nasa1,2)`, `(nasa2,2)` and `(nasa3,2)` nodes returns non-empty results, a query with $QID = 2$ is satisfied and should return the set of nodes expressed by the path with $PathID = 2$ from a document of type `nasa1`.

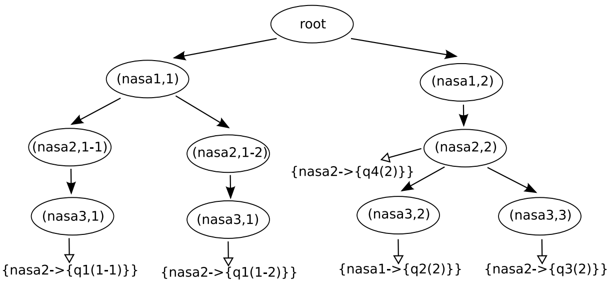


Figure 5: Join Tree

Join Tree is constructed as follows. We use the term *current context node* v_c to refer to a Join Tree node that is currently being examined and processed. We first set v_c to point to the root of Join Tree. For each query registered to the system, we retrieve the list of (*DocType*, *PathID*) pairs of the query from Table 2. For each pair from the list, we check whether a Join Tree node that represents the pair already exists as a child node of v_c. If it does, then that child node is retrieved. Otherwise, we create a new Join Tree node representing the pair, and it is added as a new child node. These steps are repeated for all (*DocType*, *PathID*) pairs. After processing the last pair, $\{DocType \rightarrow QID(PathID)\}$ mapping is created and stored in the last Join Tree node that we reach. Lastly, the above process is repeated for all queries.

3.4 Processing Join Tree

We first explain some additional data structures that we use when we process Join Tree, and explain how Join Tree is processed.

Algorithm 1 buildJoinTree(queries)

```
joinTreeRoot ← createRootNode()
for q_i ∈ queries do
    joinNode ← joinTreeRoot
    docTypeToXPathId ← {}
    for p_i ∈ q_i.getPaths() do
        node ← joinNode.getChild(p_i.getDocType(),
                                  p_i.getXPathId())
        if node = null then
            node ← createNewChildNode(joinNode,
                        p_i.getDocType(),p_i.getXPathId())
        docTypeToXPathId ← docTypeToXPathId
              ∪ {(p_i.getDocType(), p_i.getXPathId()}
        joinNode ← node
    joinNode.addXQueryId(q_i.getId())
    joinNode.
     setDocTypeToXPathId(docTypeToXPathId)
return joinTreeRoot
```

In order to distinguish different documents of the same type coming from the same stream, we assign each document a unique ID. We also keep a buffer of size b_n to store n documents of each type with path processing results. When a document of type $t(d)$ arrives, it is sent to an XPath processor as input, and the processor returns as output a set of $\{PathID \rightarrow \{v_1, \ldots, v_n\}\}$, where $\{v_1, \ldots, v_n\}$ represents a set of matching nodes for a path with *PathID*. This set is easily transformed to $\{(DocType,PathID) \rightarrow \{v_1, \ldots, v_n\}\}$, since we know the set is from a document of type $t(d)$. We use $P(d)$ to refer to this set of mappings for a document d. When a document d arrives, $P(d)$ is created and added to the buffer of type $t(d)$, and is removed from the buffer after the document has been processed. A $P(d)$ in a buffer is also sequentially removed in a FIFO fashion when the buffer becomes full or a $P(d)$ is stored for more than t_c units of time. We also maintain a global stack called Join Stack. An entry in Join Stack represents intermediate join results that have been produced so far by joining all matching nodes that are along the path between the root and a Join Tree node. Join Tree is traversed when a new document arrives from a stream and all buffers have at least one $P(d)$. Join Tree is traversed as follows. Algorithm 2 and 3 summarize the procedure.

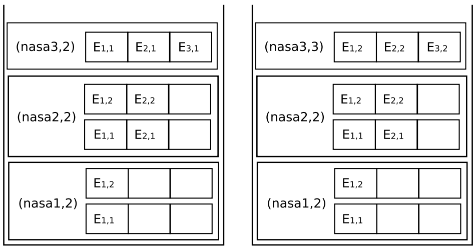
We initially set the root of Join Tree as the current context Join Tree node v_c. For each child of v_c, we obtain the type of the document that this Join Tree node represents. This can be done by checking (*DocType*, *PathID*) pair in the node. We then check whether *DocType* is the same as the type of the newly arrived document that is being processed. If *DocType* is the same, we retrieve the last $P(d)$ from the buffer that represents the document of type *DocType*. Otherwise, we obtain all $P(d)$s from the buffer of the same type. The reason for retrieving only the last $P(d)$ in the first case is to avoid duplicate join processing. For each $P(d)$ that we have obtained from the buffer, we search for the set of elements that match the path expression that v_c represents. This operation is efficiently done, as $P(d)$ is implemented in a hashtable. If an empty set is returned for the path expression, we stop traversing, and proceed to v_c's sibling nodes or backtrack. If a non-empty set is returned, we join the elements in the non-empty set with the sets of matching elements that we obtain from Join Stack, and push the newly generated sets of matching elements onto the stack. If a Join Tree node contains a mapping of $\{DocType \rightarrow QID(PathID)\}$ (see Figure 5), all query IDs are reported as matched.

Due to performance reasons, the matching elements are not immediately returned at this stage. Instead, a DOM representation of the currently streaming document is created, and each matching node from the DOM tree is decorated with the matching query IDs. We refer to such a DOM tree as a *document tree*. Matching elements are returned when Join Processing Engine finishes processing the current streaming document. Matching elements are returned in either the following two ways: (1) the decorated document tree is passed directly to an upper level application; or (2) the document tree is traversed once more to create $\{QID \rightarrow \{matching\text{-}elements\}\}$ mappings. It is possible that non-leaf nodes contain matching query IDs. In that case, subtrees rooted at these non-leaf nodes are returned as matching elements. Lastly, the above procedure is repeated for all child nodes of v_c.

Example 1. Figure 6(a) shows examples of matching elements $P(d)$ for some selected nodes in Join Tree, and Figure 6(b) and 6(c) show two instances of Join Stack after two Join Tree nodes, `(nasa3,2)` and `(nasa3,3)` in Figure 5 are traversed, respectively. In this example, Figure 6(a) shows that a Join Tree node `(nasa1,2)` has two sets of matching elements, namely $E_{1,1}$ and $E_{1,2}$, and each set contains the elements whose `text()` values are `1990` and `2000`, respectively, and all elements are from the document of type `nasa1`. Other nodes in Figure 6(a) are interpreted similarly. An entry in Join Stack contains a set of buffers (represented by a rectangle of three small boxes), and an entry in each buffer (represented by a small box) contains a set of matching elements so far. For example, the bottom entry in Figure 6(b) and 6(c) contains two buffers, each of which contains a set of matching elements denoted by $E_{1,1}$ and $E_{1,2}$. Other stack entries in Figure 6(b) and 6(c) are interpreted similarly. A buffer of sets of elements are used to represent intermediate join results. When we reach `(nasa3,2)` node in Join Tree in Figure 5, we find that the node contains $\{nasa1 \rightarrow \{q2(2)\}\}$. Since the top entry on Join Stack is not empty, we report $q2$ as matched. We retrieve matching elements $E_{1,1}$ from Figure 6(a), since $q2$ requires `text()='1990'`. The query ID=$q2$ is then added to the retrieved elements. Join Stack in Figure 6(c) is interpreted similarly. Figure 7 shows two document trees decorated with matching query IDs. Due to limited space, we use double dotted line notation between two nodes to represent an ancestor/descendant relationship in a document tree. This is similar to the double line notation commonly used in query trees.

Node	Matching Elements
(nasa1,2)	$E_{1,1} = \{e \in d(nasa1) \mid e/$`text()` $=$ '1990'$\}$ $E_{1,2} = \{e \in d(nasa1) \mid e/$`text()` $=$ '2000'$\}$
(nasa2,2)	$E_{2,1} = \{e \in d(nasa2) \mid e/$`text()` $=$ '1990'$\}$ $E_{2,2} = \{e \in d(nasa2) \mid e/$`text()` $=$ '2000'$\}$
(nasa3,2)	$E_{3,1} = \{e \in d(nasa3) \mid e/$`text()` $=$ '1990'$\}$
(nasa3,3)	$E_{3,2} = \{e \in d(nasa3) \mid e/$`text()` $=$ '2000'$\}$

(a) Matching elements



(b) At (nasa3,2) node (c) At (nasa3,3) node

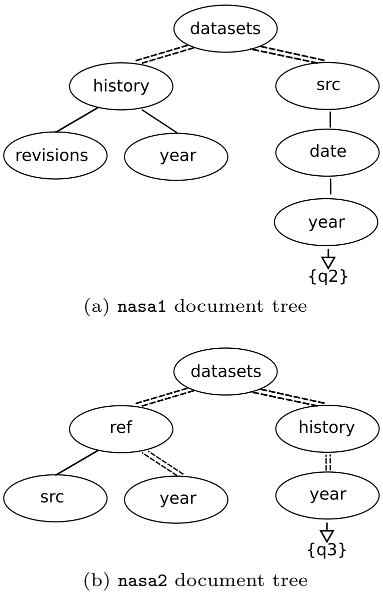
Figure 6: Examples of matching elements and the instances of Join Stack

Algorithm 2 processJoin(curDocType, joinTreeRoot)

```
joinStack ← {}
for child_i ∈ joinTreeRoot.getChildren() do
    processJoinTree(curDocType, child_i, joinStack)
```

3.5 Optimizing Join Operations

We have presented how queries are processed using Join Tree. Up to this point, we have assumed that all publishers publish new documents at the same rate. In practice, however, it is most likely that some publishers produce documents more frequently than others, and therefore, some particular types of documents are sent more frequently than other types of documents. In addition, since we process documents



(a) `nasa1` document tree

(b) `nasa2` document tree

Figure 7: Two different types of document trees are shown. Each document tree shows which node should be returned as matching element for a query (Double dotted line represent ancestor/descendant relationship in a document tree similar to the double line notation commonly used in query trees).

sequentially as they arrive, we have found that there are some join operations that are common amongst the documents that are already in the buffer, and those join operations are repeatedly performed when a new document is arrived. In this section, we present an optimization technique that allows us to further reuse some intermediate join results that were already calculated when previous documents were processed. This technique considers how frequently a particular type of document arrives, and using that information, it determines the order in which join operations should be performed. The effect of this optimization is that, once more frequently arriving types of documents are identified, it improves runtime costs when these types of documents are processed. This is done by reusing some of the join results that were previously calculated.

In order to identify the rate of which a document arrives, we maintain simple statistics about how frequently each type of document arrives at our system while processing streaming documents. These statistics can also be reinforced when similar information is available from publishers. We refer to such statistics/information for each type of a document *Frequency Factor*.

After collecting Frequency Factor for each type of document, the nodes in Join Tree are rearranged in the order of Frequency Factor such that, the least frequent type of document is placed as child nodes of the root of Join Tree, and the most frequent type of document is placed as leaves of Join Tree. Figure 8 shows an example of Join Tree whose nodes are rearranged according to Frequency Factor. In this figure, `nasa2` has been identified as the most frequent type of document, and `nasa1` has been identified as the least frequent type of document.

Join Tree is now extended to include a buffer pool. A buffer pool contains bp_n number of partitions where $bp_n \leq$ the number of Join Tree nodes. The partitions in the buffer pool are allocated in runtime in a lazy way as needed. Each partition is used to store intermediate join results between a parent and a child node when a join operation is performed. Having a buffer pool allows us to retrieve some previously calculated intermediate join results, which often prevent us from traversing the entire Join Tree from the root. Instead of traversing Join Tree when a new document arrives, we now select a group of small subtrees from Join Tree, and traverse these subtrees to perform join operations. A subtree is selected if the parent node of the subtree was previously visited, and *DocType* of the root of subtree is the same type as the streaming document. Previously visited nodes can be found in a hashtable. Checking the parent of a subtree is to ensure that, the join operations between the root of Join Tree and the parent node do not return an empty set. For each selected subtree, we do not traverse the path between the root of Join Tree and the root of subtree, if the join results for the nodes on the path between these two nodes can be retrieved from the buffer pool.

Algorithm 3 processJoinTree(curDocType, node, joinStack)

```
// curDocType: type of newly arrived document
if curDocType = node.getDocType() then
    docsToProcess ←
            getLastDocFromBuffer(curDocType)
else
    docsToProcess ←
            getDocsFromBuffer(curDocType)
for doc_i ∈ docsToProcess do
    matchingNodes ←
        doc_i.getMatchingNodes(node.getXPathId())
    if matchingNodes ≠ {} then
        joinResultsSoFar ←
            joinStack.top().joinWith(curDocType,
                                      matchingNodes)
        joinStack.push(joinResultsSoFar)
        if joinStack.top() ≠ {} then

            if node.containsMappings() then
                reportMatchingXQueries
                    (node.getMatchingXQueries())
            else
                for child_i ∈ node.getChildren() do
                    processJoinTree(curDocType, child_i,
                                    joinStack)
        joinStack.pop()
```

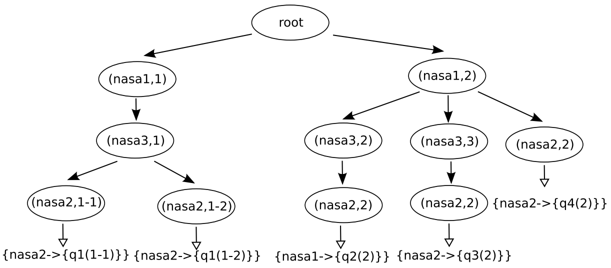


Figure 8: Rearranged Join Tree

Example 2. If the currently streaming document is of type `nasa2`, and `(nasa2,1-1)` node from Figure 8 is the only node that has a parent node which was visited previously, we only perform one join operation between `(nasa3,1)` and `(nasa2,1-1)`, as the join operations between the root and `(nasa3,1)` can be retrieved from the buffer. Other join operations are not performed, as we know they will fail eventually.

Optimized join processing is described as follows. First, we find on which level l, the nodes that have the same *DocType* as the newly arrived document are located in Join Tree, and their parent Join Tree nodes are extracted from Join Tree. Amongst these nodes, we select the nodes that were successfully traversed when a previous document had been processed. After that, we collect all child nodes from all such parent Join Tree nodes. These child nodes represent the roots of subtrees that must be traversed to perform join operations. One exception to the above selection process is when a parent node is the root of Join Tree. In that case, we simply choose all child Join Tree nodes of the root. This is the case where we do not benefit from our optimization technique, as we need to traverse Join Tree from the root. However, by carefully constructing Join Tree, this situation is minimized. Before any subtrees are traversed, all partitions in the buffer pool associated with the Join Tree nodes whose levels in Join Tree are $\geq l$ are cleared, as these partitions cannot be reused with these subtrees. To keep track of the last successfully traversed Join Tree nodes and to perform the above operations efficiently, we used an ordered hashtable of $\{DocType \rightarrow \{TraversedJoinTreeNodes\}\}$ in our implementation.

Second, for each subtree selected from the process above, we check whether the previous intermediate join results between the root of Join Tree and the parent of subtree can be found in the buffer pool. If so, that subtree is ready to be traversed. If not, we recursively check whether the intermediate join results between the root of Join Tree and any ancestor of the root of subtree can be found in the buffer pool. Once such a node is found, we start traversing back from that node to the root of the subtree. The buffer pool on the path are filled while traversing back. In the worst case, all nodes between the root of Join Tree and the root of subtree are traversed. Once the buffer pool for the root of subtree is filled up, we start traversing the subtree. While traversing a subtree, successfully traversed Join Tree nodes are marked, and our hashtable is updated accordingly in our implementation. With this approach, only a small set of subtrees are traversed, and by not traversing some upper parts of Join Tree, we skip performing some repeated join operations. Algorithm 4 summarizes the above procedures, and Algorithm 3 is modified in a way that the lines in Algorithm 5 are added to Line 12 in Algorithm 3.

Algorithm 4 processJoinOpt(curDocType, joinTreeNodeHashtable)

```
subtrees ←
        joinTreeNodeHashtable.
                getJoinTreeNodes(curDocType)
joinTreeNodeHashtable.
    clearJoinTreeNodes(curDocType)
for subtree$_i$ ∈ subtrees do
    parentOfSubtree ← subtree$_i$.getParent()
    bufferPool$_i$ ←
        parentOfSubtree.getBufferPool()
    if bufferPool$_i$ = {} then
        // recursively visit ancestor nodes
        // to find non empty bufferPool, and
        // start traversing back the same path
        // to fill up all ancestor bufferPools
        bufferPool$_i$ ← calBufferPools(subtree$_i$)
    joinStack ← {}
    if parentOfSubtree ≠ rootOfJoinTree then
        joinStack.add(bufferPool$_i$)
    processJoinTree(curDocType,subtree,joinStack)
```

Algorithm 5 processJoinTreeMod(docType,node, joinStack)

```
node.setBufferPool(joinStack.top())
joinTreeNodeHashtable.
    addJoinTreeNodes(node.getDocType(), node)
```

4 Experiments

This section presents our experiment results in detail. All experiments were executed on a Core 2 Duo 2.33 GHz laptop with 2 GB ram running Mac OS X. The SAX parser we used was Xerces Java Parser 2.8.0 (The Apache XML Project 2007). Our approach was implemented in Java 1.5, and was built on top of our previous work (Choi & Wong 2009). This section consists of three parts. The first part shows how the documents used in the experiments were prepared. The second part shows how the queries used in the experiments were prepared. The last part presents how our approach performed under various conditions.

4.1 Document Preparation

We used NASA dataset obtained from UW Database Group (2002), and this dataset was preprocessed in a similar way as Kwon et al. (2005) to generate sample XML documents. In this setting, the dataset was split into many smaller documents, and randomly selected to form three sets of documents of size [10 KB, 20 KB), [20 KB, 30 KB) and [30 KB, 60 KB). We refer to each dataset as 10k, 20k and 30k respectively in this section. In addition, we removed all `text()` values from all non-leaf nodes to make $e_1/text() = e_2/text()$ always true for all non-leaf nodes with the same names. This is to increase the number of matching queries. Finally, each dataset generated above was added to a list, and this list was duplicated to create m identical lists of documents. These duplicated lists of documents were used to represent the documents coming from different streams.

For the experiments, the documents were streamed in two ways. First, a document from each list was randomly selected and streamed until all lists became empty. Second, all the first documents from the lists were removed and streamed sequentially until all lists became empty. This is to further increase the number of matching queries in the experiments, as many join operations in queries result in joining with the copies of the same documents. We will refer to each streaming method as a random and sequential streaming method, respectively.

4.2 Query Preparation

All queries we used in the experiments were generated as follows. First, the document structure of NASA dataset was extracted. Second, we randomly chose an element name e from the dataset, and selected n_p elements from the document structure whose names are the same as e. For each selected element, the path between the root and the element was scanned. While scanning the nodes on the path, we chose a node with a probability of 80%, as well as maintaining the element orders. For the case where a node was not chosen, we replaced `/` of the next node with `//`. Amongst the chosen nodes, we replaced nodes with `*` with a probability of 10%. Moreover, we replaced `/` with `//` with a probability of 10%. This step was repeated until n_p path expressions were created. Third, the path expressions were grouped such that each group contained the path expressions whose last elements had the same names. Fourth, j number

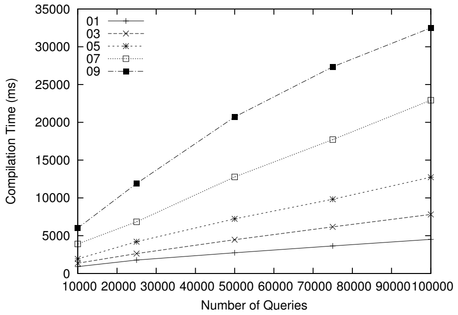


Figure 9: Compilation time vs. Number of queries that have various numbers of join operations

of path expressions were randomly selected from one path expression group, and combined to create a list of `for` clauses. A `where` clause was created by joining all path expressions found in the `for` clauses. A `return` clause was created by randomly choosing a path from a `for` clause. Lastly, the fourth step was repeated until n_q distinct queries were generated.

4.3 Results

Figure 9 shows compilation times when instances of Join Tree are built for various numbers of queries with various numbers of join operations. The compilation time increases as the number of queries increases, as the Join Tree building process is proportional to the number of queries to process. However, the rate of increase slowly drops as the number of queries increases. This is because, the number of new Join Tree nodes that must be created decreases, as nodes are shared amongst similar queries. Similarly, the compilation time increases as the number of join operations per query increases. This is because, as the depth of Join Tree increases, the number of Join Tree nodes that must be created and traversed increases.

In the following experiments, the Join Tree created in this phase was used, and all matching nodes for each query were computed as a part of join process. In addition, we used the following default values unless specified otherwise—the size of document buffer for each type of document was set to 1, each query had 5 join operations, 10k dataset was used, and documents were streamed in sequential order. Moreover, all experiments were executed with our join optimization. The processing times reported here are the average running times taken to perform join operations for each document. It also includes the times taken to calculate all matching elements for all queries. The processing times taken by an XPath processor were not included, and these can be found in our previous work. (Choi & Wong 2009)

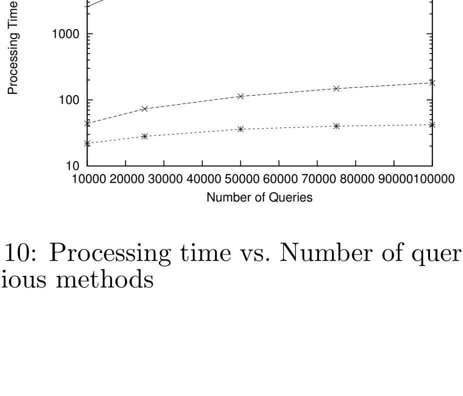


Figure 10: Processing time vs. Number of queries using various methods

Figure 10 shows how our Join Tree approach is performed against both naive Sequential Join approach and YFilter (Diao et al. 2003). In this experiment, 30k dataset was used. Compared to Sequential Join approach, the processing time and the rate of which our approach increases in time as the number of queries increases is orders of magnitude lower than that of Sequential Join approach. Due to unavailability of the implementation of a previous work by Hong et al. (2007), we were unable to compare our approach directly with Hong et al. Instead, we compare our approach indirectly with them via YFilter, as Hong et al. state that they use YFilter as an XPath processor in their implementation, and the XPath evaluation costs by YFilter are much smaller than their join processing costs.

To compare our Join Processing Engine with YFilter, we prepared XPath expressions from XQuery queries as follows. For each XQuery used in the experiment, we randomly selected a `for` clause, and then extracted the XPath path expression from it. We repeated until we collected n distinct XPath expressions, and they were processed by YFilter against the same dataset. Note that the experiment is in favor to YFilter, as it only needs to process up to $100,000$ expressions, although queries with j join operations typically need to process $100,000 \times (j+1)$ path expressions. Figure 10 shows that our join processing costs are lower than YFilter's XPath processing costs. For the following experiments, we only present the experiment results for our Join Tree approach.

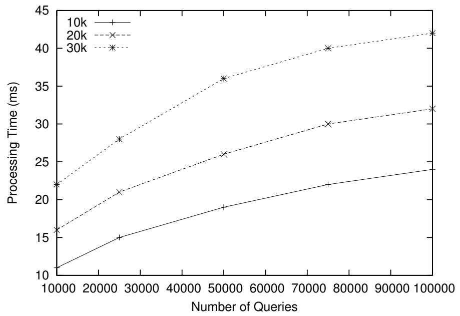


Figure 11: Processing time vs. Number of queries with various sizes of documents

Figure 11 shows the join processing times when various numbers of queries are evaluated against various sizes of documents. As the number of queries increases, the rate of which the join processing time increases drops slowly. This is because, as the number of queries increases, the total number of unique join operations that must be evaluated increases at a much slower rate than the rate of which the total number of join operations increases. In addition, as the number of queries increases, the number of join operations that are shared amongst queries also increases, which also lowers the rate of growth. Lastly, similar patterns were observed for queries with 1, 3, 7, etc... join operations, and when documents were streamed randomly.

Figure 12 shows the join processing times when various numbers of queries with various numbers of join operations were evaluated. The rate of which the processing time increases slowly drops as the number of queries becomes larger. This is because, as the number of queries increases, the number of intermediate join results that are shared amongst similar queries also increases. Therefore, the total number of join operations that must be evaluated does not increase at the same rate as the number of queries increases. Furthermore, the processing time increases as the number of join operations per query increases.

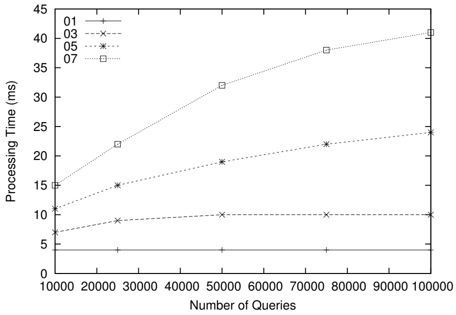


Figure 12: Processing time vs. Number of queries with various numbers of join operations

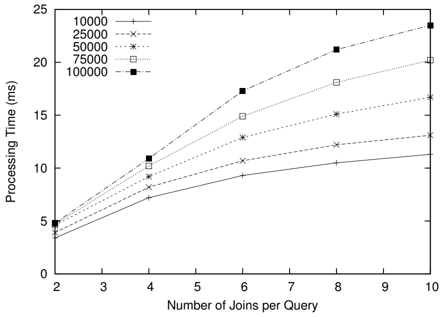


Figure 13: Processing time vs. Number of join operations with various queries

This is because, as the depth of Join Tree increases, the number of Join Tree nodes increases, which result in a larger number of join operations to evaluate.

Figure 13 shows join processing times when queries with various numbers of join operations were evaluated. In this experiment, documents were streamed in a random order. As the number of join operations increases, the growth of processing time slowly decreases. This is because the number of intermediate join results being shared are increased, and while Join Tree is being traversed, short cut evaluations are extensively performed to skip branches that do not have any matching elements. In addition, the number of matching queries has also dropped as the number of join operations increases.

Figure 14 shows processing times when the number of documents stored in a buffer was varied. In this experiment, the number of join operations per query was set to 1, and the number of queries were set to 100,000. The join processing time linearly increases as the number of buffers increases. This is because, as the number of documents in the buffer increases, the number of join operations that need to be performed every time when a new document arrives increases. We are currently working on sharing buffers to decrease the linear growth in time.

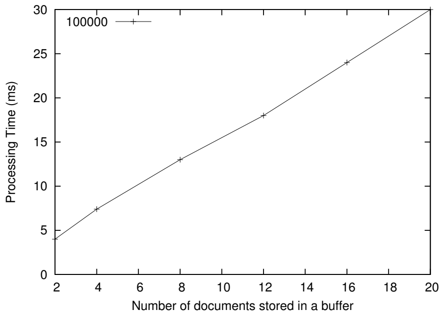


Figure 14: Processing time vs. Number of documents stored in a buffer

Table 3: Analyzing the number of join operations in both normal and optimized modes

	Name	Normal	Optimization
1	# of skipped Join Tree nodes	361	233
2	# of join operations	643	478
3	# of non-empty join operations	590	408
4	# of empty join operations	102	70
5	# of successfully traversed leaves	189	189

Table 3 shows how much improvement we get with our Join Tree optimization. The table displays the average number of join operations performed for each document. In this experiment, the total number of queries were set to 50,000, but similar patterns were observed when the total number of queries were 10,000, 25,000, etc... The first row shows the number of Join Tree nodes that were examined but not traversed further because there were no matching instances for those queries. These decisions were made without performing join operations. The second, third and fourth rows show the total number of join operations performed, and out of these join operations, how many of them produced non-empty and empty sets. The last row shows the number of Join Tree leaves that were successfully reached and therefore could find matching instances for queries. The analysis shows that the total number of Join Tree nodes that must be traversed was reduced, and hence the total number of join operations that must be performed was reduced substantially. As a result, it significantly increased the overall performance of the system, since join operations were the most expensive operations in terms of runtime costs.

5 Conclusion

We have presented an efficient approach for evaluating a large number of XQuery queries that contain inter-document value-based join operations against streaming documents. We use Join Tree to group and process similar queries simultaneously. While queries are being processed, intermediate join results are shared in order to reduce the overall number of join operations. In addition, unlike many previous works, we return all matching nodes for each matching query. Experiments have shown that, our approach can efficiently evaluate a large number of queries with join operations.

There are several possibilities for future work. First, we are currently working on to support more expressive queries, and join optimization based on the size of Join Tree when frequency information is not available. Second, we are looking at integrating some previous works to improve buffer management. Third, we are looking at integrating some of the traditional join optimizations to further improve both compile and runtime performance.

References

Barta, A., Consens, M. P. & Mendelzon, A. O. (2005), Benefits of path summaries in an xml query optimizer supporting multiple access methods., *in* 'Proceedings of the 31st International Conference on Very Large Data Bases', ACM, Trondheim, Norway, pp. 133–144.

Barton, C., Charles, P., Goyal, D., Raghavachari, M., Fontoura, M. & Josifovski, V. (2003), Streaming xpath processing with forward and backward axes, *in* 'Proceedings of the 19th International Conference on Data Engineering', IEEE Computer Society, Bangalore, India, pp. 455–466.

Boncz, P. A., Grust, T., van Keulen, M., Manegold, S., Rittinger, J. & Teubner, J. (2006), Monetdb/xquery: a fast xquery processor powered by a relational engine., *in* 'Proceedings of the ACM SIGMOD International Conference on Management of Data', ACM, Chicago, IL, pp. 479–490.

Bruno, N., Koudas, N. & Srivastava, D. (2002), Holistic twig joins: optimal xml pattern matching, *in* 'Proceedings of the ACM SIGMOD International Conference on Management of Data', ACM, Madison, WI, pp. 310–321.

Chan, C.-Y., Felber, P., Garofalakis, M. & Rastogi, R. (2002), 'Efficient filtering of xml documents with xpath expressions', *The VLDB Journal* **11**(4), 354–379.

Chen, Y., Davidson, S. & Zheng, Y. (2006), An efficient xpath query processor for xml streams, *in* 'Proceedings of the 22nd International Conference on Data Engineering', IEEE Computer Society, Atlanta, GA, p. 79.

Choi, R. H. & Wong, R. K. (2009), 'Efficient filtering of branch queries for high-performance xml data services', *To appear: Journal of Database Management* .

Clark, J. & DeRose, S. (1999), 'Xml path language (xpath)'. `http://www.w3.org/TR/xpath`.

Diao, Y., Altinel, M., Franklin, M. J., Zhang, H. & Fischer, P. (2003), 'Path sharing and predicate evaluation for high-performance xml filtering', *ACM Trans. Database Syst.* **28**(4), 467–516.

Diao, Y. & Franklin, M. J. (2003), Query processing for high-volume xml message brokering., *in* 'Proceedings of 29th International Conference on Very Large Data Bases', Morgan Kaufmann, Berlin, Germany, pp. 261–272.

Diao, Y., Rizvi, S. & Franklin, M. J. (2004), Towards an internet-scale xml dissemination service., *in* 'Proceedings of the 30th International Conference on Very Large Data Bases', Morgan Kaufmann, Toronto, Canada, pp. 612–623.

Goldman, R. & Widom, J. (1997), Dataguides: Enabling query formulation and optimization in semistructured databases., *in* 'Proceedings of 23rd International Conference on Very Large Data Bases', Morgan Kaufmann, Athens, Greece, pp. 436–445.

Gottlob, G., Koch, C. & Pichler, R. (2005), 'Efficient algorithms for processing xpath queries', *ACM Trans. Database Syst.* **30**(2), 444–491.

Gou, G. & Chirkova, R. (2007), Efficient algorithms for evaluating xpath over streams., *in* 'Proceedings of the ACM SIGMOD International Conference on Management of Data', ACM, Beijing, China, pp. 269–280.

Grust, T., Rittinger, J. & Teubner, J. (2007), Why off-the-shelf rdbmss are better at xpath than you might expect., *in* 'Proceedings of the ACM SIGMOD International Conference on Management of Data', ACM, Beijing, China, pp. 949–958.

Gupta, A. K. & Suciu, D. (2003), Stream processing of xpath queries with predicates, *in* 'Proceedings of the ACM SIGMOD International Conference on Management of Data', ACM, San Diego, CA, pp. 419–430.

Hong, M., Demers, A. J., Gehrke, J., Koch, C., Riedewald, M. & White, W. M. (2007), Massively multi-query join processing in publish/subscribe systems., *in* 'Proceedings of the ACM SIGMOD International Conference on Management of Data', ACM, Beijing, China, pp. 761–772.

Huo, H., Wang, G., Hui, X., Zhou, R., Ning, B. & Xiao, C. (2006), Efficient query processing for streamed xml fragments, *in* 'Proceedings of the 11th International Conference on Database Systems for Advanced Applications', Springer, Singapore, pp. 468–482.

Koch, C., Scherzinger, S., Schweikardt, N. & Stegmaier, B. (2004), Schema-based scheduling of event processors and buffer minimization for queries on structured data streams, *in* 'Proceedings of the Thirtieth International Conference on Very Large Data Bases', Morgan Kaufmann, Toronto, Canada, pp. 228–239.

Kwon, J., Rao, P., Moon, B. & Lee, S. (2005), Fist: Scalable xml document filtering by sequencing twig patterns., *in* 'Proceedings of the 31st International Conference on Very Large Data Bases', ACM, Trondheim, Norway, pp. 217–228.

Li, X. & Agrawal, G. (2005), Efficient evaluation of xquery over streaming data., *in* 'Proceedings of the 31st International Conference on Very Large Data Bases', ACM, Trondheim, Norway, pp. 265–276.

Olteanu, D., Furche, T. & Bry, F. (2004), An efficient single-pass query evaluator for xml data streams, *in* 'Proceedings of the 2004 ACM symposium on Applied computing', ACM, New York, NY, pp. 627–631.

Onizuka, M. (2003), Light-weight xpath processing of xml stream with deterministic automata, *in* 'Proceedings of the 12th International Conference on Information and Knowledge Management', ACM, New Orleans, LA, pp. 342–349.

Peng, F. & Chawathe, S. S. (2005), 'Xsq: A streaming xpath engine', *ACM Trans. Database Syst.* **30**(2), 577–623.

Rao, P. & Moon, B. (2004), Prix: Indexing and querying xml using prüfer sequences., *in* 'Proceedings of the 20th International Conference on Data Engineering', IEEE Computer Society, Boston, MA, pp. 288–300.

The Apache XML Project (2007), 'Xerces2 java parser'. `http://xerces.apache.org/xerces2-j/`.

Uchiyama, H., Onizuka, M. & Honishi, T. (2005), Distributed xml stream filtering system with high scalability, *in* 'Proceedings of the 21st International Conference on Data Engineering', IEEE Computer Society, Tokyo, Japan, pp. 968–977.

UW Database Group (2002), 'Xml data repository'. `http://www.cs.washington.edu/research/xmldatasets/`.

Wang, H., Park, S., Fan, W. & Yu, P. S. (2003), Vist: A dynamic index method for querying xml data by tree structures., *in* 'Proceedings of the ACM SIGMOD International Conference on Management of Data', ACM, San Diego, CA, pp. 110–121.

Access Control: What is Required in Business Collaboration?

Daisy Daiqin He[1] **Michael Compton**[2] **Kerry Taylor**[2] **Jian Yang**[1]

[1] Department of Computing, Macquarie University
Sydney Australia,
Email: {daiqin, jian}@ics.mq.edu.au

[2] ICT Centre
CSIRO Australia,
Email: {michael.compton, kerry.taylor}@csiro.au

Abstract

Access control has been studied for sometime, and there are a number of theories and techniques for handling access control for single or centralised systems; however, unique and challenging security issues concerning collaboration in the context of service oriented computing (SOC) have arisen due to the dynamic and loosely coupled nature of the environment in which these collaborations are conducted. Individual organisations usually define their access control policies independently. When a collaboration opportunity arrives, a number of problems arise, such as: determining if the collaboration is possible given the access control policies, defining the policy for the collaboration and deciding under what conditions a service is allowed to be forwarded to other parties. Furthermore, different types of collaboration, in terms of the way collaboration is carried out, require different access control support. In this paper, we propose a model encoded in description logic to capture all the necessary elements for specifying access control policy for collaboration. Based on the model, various inconsistencies between access policies from different business units are identified. The paper also shows how a description logic reasoner can be used to prove that two policies are suitable, or not suitable, for collaboration. The policy model and policies are encoded in a $\mathcal{SROIQ}$ knowledge base. Although access control policies focus on a single system or a single business party's requirements, the method presented in this paper allows a logical analysis of the suitability of potential collaboration partners. We believe this work is laying a foundation for access policy development, negotiation and enforcement for cross-organization collaborations.

1 Introduction

Although Web Service technologies provide technological support for dynamic, cross-organization collaboration, security concerns can be a barrier to the adoption of this new technology. Service collaboration through service compositions or other means, could have different access control requirements to the individuals services in the collaboration; how to provide end-to-end security guarantees is still an unsolved problem. This paper discuses a method that can be used to solve part of this problem by using technologies designed for the Semantic Web and Services to analyse the access control policies of potential collaboration partners and prove if collaboration will respect the policies. We emphasize, however, that the proposed method is not restricted to Web Service. The problem we are addressing could be applicable to generic business collaboration domain.

The access control policy of a single organisation or service is defined (in a role-based model) in terms of roles and their privileges. Given a request to access a resource or perform an operation, the service enforces the policy by analysing the credentials of the requester and deciding if the requester is authorised to perform the actions in the request.

Organisations collaborate with each other in various ways. The collaboration could be through an agent, or a direct collaboration to provide joint service. Before organizations engage in collaboration, their authorization policies need to be analyzed to decide the possibility of collaboration under the authorization constraints defined by each individual party. Therefore, the consistency of access policies of different organisations needs to be evaluated before a collaboration can be formed.

Collaborations can reveal the differences between the participants policies. For example, if a radiology institute wants to collaborate with a medical centre, accepting on line bookings from the medical centre, but inconsistencies exist between two party's access control policies, then either the collaboration could not be established securely or some negotiation is required to form the collaboration.

In a collaboration, a service can be accessed by a party that can pass it to other parties. Suppose a patient wants to keep privacy on his health records except to the attending doctors. If the policy of the doctor's medical centre allows other research institute to access the patient records, then the patient's wishes might not be respected. It is important to use access policies to control the way in which information or services are propagated between organizations.

Intuitively, the concept of 'access policy consistency' means that (for the same service) the access policies of different organizations are conflict free. And organizations are able to collaborate in the intended way securely in terms of access control policies.

Access control issues in single organisations or single domains have been well studied (Sirer & Wang 2002, Kagal et al. 2004, Bhatti et al. 2004, Srivatsa et al. 2007); however, access control in a collaborative environment has just started to attract the attention of the research community (Rouached & Godart 2007, Yau & Chen 2008), and little attention has been given to consistency study between access control policies of different collaboration participants, particularly in the context of Web Services.

Our previous analysis shows that there are different ways of collaborating, and that each imposes dif-

Copyright ©2009, Australian Computer Society, Inc. This paper appeared at the 20th Australasian Database Conference (ADC 2009), Wellington, New Zealand, January 2009. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 92, Athman Bouguettaya and Xuemin Lin, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

ferent consistency constraints on the access policies of prospective partners (He & Yang 2007). Understanding the different requirements on collaboration partners' authorization policies for different types of collaboration is important, and access control in collaboration should take individual organization's access policies into account, as well as the type of the collaboration.

Our previous work (He & Yang 2007) gave a model of access control policy and showed what sorts of inconsistencies occur between policies and how these inconsistencies affect the various patterns of collaboration. In this paper we extend those ideas by showing how Description Logic can be used to formally model policies and how a Description Logic reasoner (an automated proof engine) can analyse the inconsistencies in policies and prove if the prospective partners could collaborate securly in terms of access control policy.

Description logics are weak fragments of first-order logic. The compensation for their limited expressively is decidability, making them an option when automated proof is required. The W3C has produced the Logic OWL-DL as a Description Logic standard for the so called Semantic Web (*Web Ontology Language (OWL)* 2004). OWL-DL is based on $\mathcal{SHOIN}$, and the logic $\mathcal{SROIQ}$ (Horrocks et al. 2006) is proposed as the logic for the next OWL standard (*OWL 1.1 Web Ontology Language* 2006). We use $\mathcal{SROIQ}$ here because it is expressive for a description logic and, though not yet a W3C OWL standard, it is supported by the Protégé[1] editor and the pellet[2] (Sirin et al. 2007) reasoner.

We first develop a model of access control policies in $\mathcal{SROIQ}$ and then show how two policies can be evaluated by comparing them in the model. We encode the inconsistency tests as concepts and relations in our access control model. Individual policies expressed using the model can then be compared and tested. Given two set of policies, with the roles and privileges of the two organisations suitably related, a reasoner will prove that the tests are either satisfiable or unsatisfiable and these results can be analysed to check whether they satisfy the requirements for the particular collaboration. Since the tests are part of the general model they are generic, meaning they can be expressed once, proven to encode the required meaning and used to testing any two policies. Because Description Logics are decidable, a reasoner for $\mathcal{SROIQ}$ will always terminate with the proofs, no matter how complex the policies.

The rest of paper is organized as follows. In the preliminaries section (Section 2), We first discuss different collaboration patterns in Section 2.1 and review the syntax and semantics of $\mathcal{SROIQ}$ in Section 2.2. In Section 3, we propose a description logic model for access control policy. The inconsistency tests are defined and proved in Section 4. In Section 5, we analyze requirements for collaborative policy (5.1) and discuss and example (5.2). Related work is discussed in Section 6, and concluding remarks and outline of our future research directions are presented in Section 7.

2 Preliminaries

Before we propose a model for access control policy and conduct an analysis of the inconsistencies, we give a brief introduction to collaboration patterns and description logic.

2.1 Collaboration Patterns

Cross-organization collaborations consist of complex relationships and interactions among organizations. Some organizations collaborate with others through an agent; some organizations might play a role of 'middleman' that pass one organization's service to another. We have concluded several different types of basic collaboration between organizations in our previous work (He & Yang 2007). Some of the identified patterns are presented in Figure 1. Here, we introduce a number of patterns, but focus on one of collaboration pattern in this paper: Service Propagation (SP).

- **Composite Services**. The Composite Service we discuss here refers to the service that is based on the integration of multiple service providers. Two different cases are identified in service composition:
  1. **Composite service with agent** (**CSWA**): Multiple numbers of service providers provide their services through an centralized agent. For example in Figure 1, Health Insurance Company normally works with different medical service providers. Each party has its own security policy. The insurance company could have several service providers for same type of service and it works as an agent, and customers can only access those services through the network of the insurance company.
  2. **Joined service without an agent** (**JSOA**): Two organizations involving in a peer-to-peer collaboration and provide a joined service by integrating their business processes or part of their processes together to form a new service directly without any agent. In Figure 1, heart disease specialist clinic collaborates with a community service center to provide a post-treatment care plan to elders, they integrate part of their services directly without any agent.
- **Service Propagation** (**SP**): it depicts collaborations that involving multiple organizations and 'forward' privilege could be passed from one organization to another organization.

Service Propagation (**SP**): Service Propagation we discuss here refers to collaborations involving privilege propagation. As illustrated in Figure 1, patient is the owner of patient records who has the ultimate right regards to her/his own record. Patient can grant access right, i.e., access and forward privileges to her/his General Practitioner (GP) in a medical center so that the GP can access the patient's health records for diagnosis purpose. If the 'forward' privilege is granted to the GP, the GP could forward this access right to a third party, e.g. staff in hospital emergency room in Figure 1 example. In this scenario, patient is service owner and GP is the collaborative partner and service propagator who could grant access right to third parties.

The evaluation in the above example three parties are involved, comparison and evaluation has to be carried out twice:

1. The first one is between the patient and GP's clinic. The service owner – the patient will be responsible for the first evaluation to find the right partner whose authorization policy does not violate patient policy.

[1] Protégé 4.0 beta is available at `http://protege.stanford.edu/`

[2] Pellet is available at `http://pellet.owldl.com/`

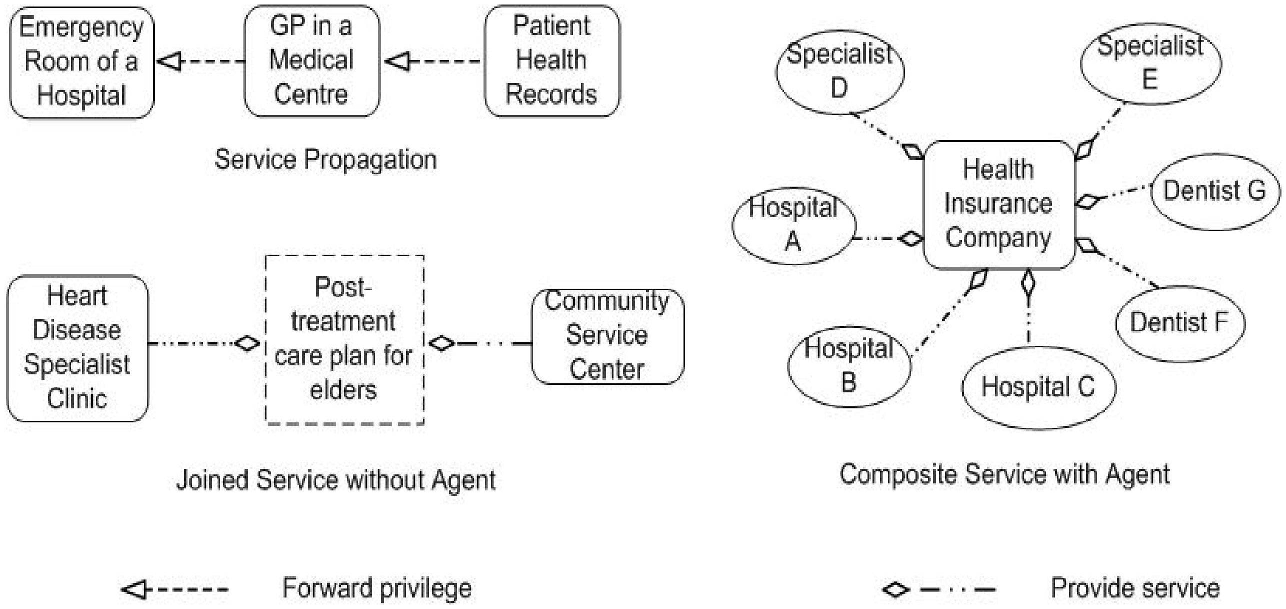


Figure 1: Cross-organization collaboration Patterns

2. The second one is between the clinic and the emergency room. The GP's clinic will be responsible for the second evaluation to find the right partner whose policy is not conflict with the patient's policy and policy of the GP's clinic.

2.2 Description Logic

In Description Logic (DL) the objects of interest (the domain of discourse) is modelled using axioms about concepts, roles and individuals. In $\mathcal{SROIQ}$ these axioms are stated in the TBox, the RBox and the ABox respectively. Figure 2 accompanies the following paragraphs on the syntax and semantics of $\mathcal{SROIQ}$ as used in this paper — Horrocks et al. (2006) give the full definition of $\mathcal{SROIQ}$. In the following, C and D range over concepts, with A for atomic concepts; S, R and subscripted versions of these range over roles; and x and y range over individuals.

Concepts are defined in the TBox, starting with atomic concepts and building more complex definitions from these. All concepts are subsumed by the universal concept $\top$, called *Thing* in OWL. The complement, union and intersection of concepts are used to define concepts in terms of others, and hierarchies of concepts are constructed using concept inclusion. Concepts can be specified as disjoint from other concepts. Existential and universal restrictions specify concepts in terms of the relationship to other concepts through roles.

Axioms defining roles are specified in the $\mathcal{SROIQ}$ RBox. DL roles, binary predicates on individuals, show the relationships between individuals. Roles can be constructed into hierarchies, composed (composition of relations) and specified as inverses of other roles. Further, roles can be stated as transitive, symmetric, reflexive or functional (single-valued).

Individuals represent members of the domain. Assertions about the existence of individuals, their classification into concepts and their participation in roles are made in the ABox.

A $\mathcal{SROIQ}$ knowledge base $\mathcal{K}$ comprises a TBox, an RBox and an ABox. The semantics of a knowledge base is given by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that specifies a set of objects for the domain $\Delta^{\mathcal{I}}$ and a function $\cdot^{\mathcal{I}}$ that maps each individual to an object in $\Delta^{\mathcal{I}}$, each concept to subset of $\Delta^{\mathcal{I}}$ and each role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of $\mathcal{K}$, written $\mathcal{I} \models \mathcal{K}$, if it is consistent with the axioms in the TBox, RBox and ABox. A concept C in the TBox of a knowledge base $\mathcal{K}$ is satisfiable if $\mathcal{I} \models \mathcal{K}$ for some interpretation $\mathcal{I}$ where $C^{\mathcal{I}} \neq \emptyset$. A concept C subsumes a concept D if $C \sqsubseteq D$ is implied by the knowledge base: equivalently, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ in all models.

Since the subsumption and satisfiablilty problems in $\mathcal{SROIQ}$ are decidable (they are also mutually reducible) a $\mathcal{SROIQ}$ reasoner can always determine the satisfiability of a concept. In this work we are mainly interested in satisfiability checking, with subsumption of minor importance. We use Protégé 4.0 beta to edit our $\mathcal{SROIQ}$ definitions and pellet to prove satisfiability.

The DL in this paper follows the convention of concepts beginning with a capital letter and roles with a lowercase letter. For simplicity we write

$$role \ : \ (ConceptA \times ConceptB)$$

to indicate that *role* is a DL role with domain *ConceptA* and range *ConceptB*. The definition of a role may be superscripted with $\mathcal{F}$ to indicate a functional role, $\mathcal{T}$ for transitive, $\mathcal{S}$ for reflective and $\mathcal{R}$ for reflexive. That instance a is classified into concept C is written $C(a)$. DL expressions are written in the so-called German DL Syntax.

3 Model for Access Control

We base our policy model on the core definition of Sandhu et al.'s (1996) Role-Based Access Control (RBAC), now a NIST standard (*ANSI INCITS 359-2004* 2004). RBAC models access control in terms of roles, job functions, and the permissions assigned to those roles. We remove notions of sessions and users from Core RBAC, sessions because they represent the dynamic rather than structural elements of access control, and in place of users we add the credentials required to access a role. The role hierarchies of Hierarchical RBAC are not captured by our current model, but we plan to investigate role hierarchies in future work. The purpose of the model is to provide a

Constructor name	Syntax	Semantics
universal	$\top$	$\Delta^{\mathcal{I}}$
empty	$\bot$	$\emptyset$
atomic concept	A	$A^{\mathcal{I}}$
concept negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
concept intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
concept union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
concept equivalence	$C \equiv D$	$C^{\mathcal{I}} \equiv D^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}}.(x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
universal restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}}.(x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
role inclusion	$S \sqsubseteq R$	$S^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
role inverse	R^{-}	$\{(y,x) \mid (x,y) \in R^{\mathcal{I}}\}$
role composition	$S_1 \circ \ldots \circ S_n \sqsubseteq R$	$S_1^{\mathcal{I}} \circ \ldots \circ S_n^{\mathcal{I}} \subseteq R^{\mathcal{I}}$

Figure 2: The constructors and semantics (for an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$) of $\mathcal{SHOIN}$

formal framework that is representative, though not prescriptive, of policy and highlights issues relevant to collaboration.

Each service in an organisation has a policy describing the privileges assigned to the roles that may access the service and, because the processing for any service is the same, we describe the model without referring further to the policies of different services in the same organisation.

Figure 3 depicts the policy model. This paragraph describes the model in general terms and then the remainder of the section describes the DL encoding of the model. The main entities in the model are roles, credentials, privileges, obligations and provisions. The users of the service are not defined and may be, for example, humans or computer agents. Users are not statically assigned to roles, rather credentials are used to authenticate the holder as authorised to act in a particular role. Each role is assigned a number of privileges, which, for example, may be the right to edit files, access resources or release information. Obligation and provision conditions are attached to each assignment of a privilege to a role. Obligations are conditions that must be satisfied after the privilege is accessed, while provisions are conditions that must be satisfied before the privilege is accessed: hence, provisions further constrain the right to access a privilege, such as restricting the time a privilege is accessed, while obligations enforce what must be done after the privilege is accessed, such as writing to a log.

There is an unfortunate name clash between DL roles, which are binary relations, and RBAC roles, which are units of authorisation in RBAC and individuals in the concept *Role* in the policy model. Here, either the context will make clear what is meant or the meaning is explicitly stated; often relation is used in place of DL role.

In the DL encoding, the concept *Role* models the set of roles, individuals from the concept *Credentials* model the set of credentials required by a role and the relation *requires* links a role to the credentials required to authenticate in that role.

$$\begin{aligned} Role &\sqsubseteq \exists requires.Credentials \\ requires^{\mathcal{F}} &: (Role \times Credentials) \\ satisfies &: (Credentials \times Credentials) \end{aligned} \tag{1}$$

The definition of *Role* ensures that every role is linked through the *requires* relation to a set of credentials. Since *requires* is defined as functional (single-valued), each role is linked to only one set of credentials.

The structure of credentials are not further modelled here; though credentials were considered more deeply in the original work (He & Yang 2007). The model assumes a preordering, *satisfies*, on *Credentials* that, in the absence of further structure, shows when one set of credentials would also satisfy the requirements of another set. For example, credentials requiring a password and a digital certification would also satisfy credentials only requiring a certificate.

The only relations directly supported in a DL are binary roles, hence the ternary relation between privileges, obligations and provisions in the policy model needs to be represented by a concept. The concept *PrivilegeAssignment* represents the ternary relation and relations *privilege*, *obligation* and *provision* link a *PrivilegeAssignment* to its associated *Privilege*, *Obligation* and *Provision*.

$$\begin{aligned} &PrivilegeAssignment \sqsubseteq \exists privilege.Privilege \\ &PrivilegeAssignment \sqsubseteq \exists obligation.Obligation \\ &PrivilegeAssignment \sqsubseteq \exists provision.Provision \\ &privilege^{\mathcal{F}} : (PrivilegeAssignment \times Privilege) \\ &provision^{\mathcal{F}} : (PrivilegeAssignment \times Provision) \\ &obligation^{\mathcal{F}} : (PrivilegeAssignment \times Obligation) \end{aligned} \tag{2}$$

*Obligation*s and *Provision* are themselves subconcepts of *PrivilegeCondition*.

$$\begin{aligned} Obligation &\sqsubseteq PrivilegeCondition \\ Provision &\sqsubseteq PrivilegeCondition \end{aligned} \tag{3}$$

As with credentials, the exact structure of privileges, obligations and provisions are not important for this paper, and again assume an ordering on obligations and provisions.

$$\begin{aligned} obl\_order &: (Obligation \times Obligation) \\ prov\_order &: (Provision \times Provision) \end{aligned} \tag{4}$$

The intent is the same as for *satisfies*: that is, if $(o_1, o_2) \in obl\_order$ then o_1 is a stronger obligation than o_2, and similarly for *prov_order*. That $(c_1, c_2) \in satisfies$ (and similarly for *obl_order* and *prov_order*) may be read as if it were written c_1 *implies* c_2 because, for the purposes of this paper, the logical meaning of, and logical relation between, credentials, obligations and provisions is more important than any structure these elements of a policy might have.

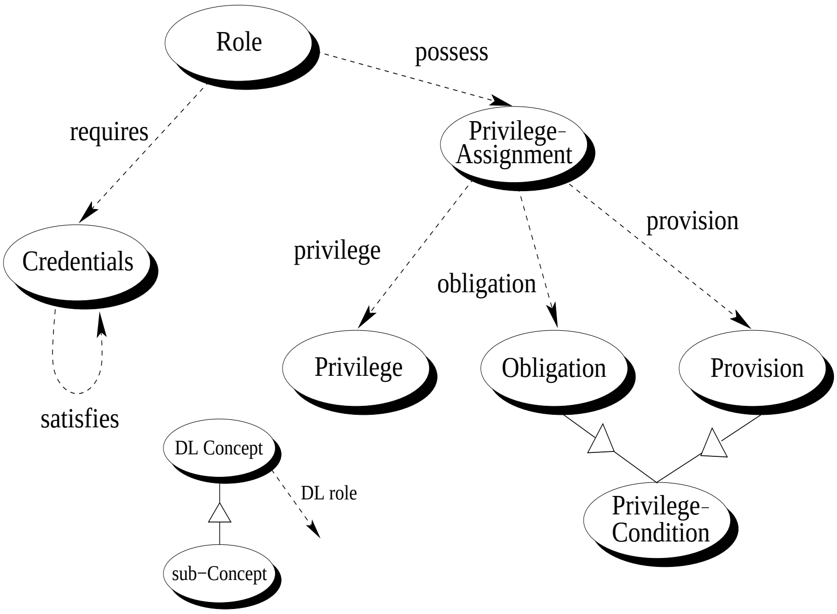


Figure 3: Policy Model (relations *privilege*, *obligation* and *provision* not shown)

A role may posses many privileges, but a particular privilege assignment can be made to only one role.

$$\begin{aligned} possesses &: (Role \times PrivilegeAssignment) \\ possesses^{-\mathcal{F}} & \end{aligned} \tag{5}$$

Note that (5) merely enforces that any particular instance of a privilege assignment cannot be allocated to multiple roles; it does not preclude exactly the same privilege and conditions being allocated to multiple roles; (5), however, enforces that if the same privilege and conditions are allocated to multiple roles, a different *PrivilegeAssignment* must be used in each case. The purpose of the restriction is to ensure that the role allocated to any particular privilege assignment can be determined (this is necessary for example in Section 4.3.1).

4 Classification of Inconsistencies

Before discussing role, credential and privilege inconsistencies, the three types of inconsistencies discussed in this paper, this section further outlines our model for consistency checking.

Assume two organisations, A and B, requiring a collaboration, where both organisations have an access control policy, P_A for A and P_B for B, encoded in the model defined in the previous section (or defined in some other way and translated into the model). For consistency and collaboration checking, P_A and P_B must first be combined into a single model and then checked. The policy model from the previous section is extended by allowing the encoding and checking for P_A and P_B.

First, the model is extended for the roles in P_A and P_B.

$$\begin{gathered} Role_A \sqsubseteq Role \\ Role_B \sqsubseteq Role \\ Role_A \text{ disjoint with } Role_B \end{gathered} \tag{6}$$

Next, a comparability relation between the roles in the two organisations is defined.

$$\begin{aligned} role\_comp_{AB} &: (Role_A \times Role_B) \\ role\_comp_{BA} &: (Role_B \times Role_A) \end{aligned} \tag{7}$$

The comparability relations in (7), the following relations in (8) and extensions to *obligation*, *provision* and *privilege*, record the notion that to meaningfully compare the policies P_A and P_B some basis for this comparison is required. Either human experts from A and B, metadata or some automated system is required to specify which roles in $Role_A$ relate to which roles in $Role_B$ and what privileges are equivalent; while this process is important for checking two policies, it is not dealt with in description logic and for the purposes of this paper it is simply assumed that that the relations are supplied along with P_A and P_B.

The relations $role\_comp_{AB}$ and $role\_comp_{BA}$ reflect the notion that for some collaborations the correspondence between roles will almost be an equivalence but, in other cases, a single role may be related to many roles in the other organisation: for example, doctor in one organisation might relate to both physician and specialist in the other.

A similar notion to the role comparisons, though in this case an equivalence, is introduced for privileges.

$$\begin{gathered} Privilege_A \sqsubseteq Privilege \\ Privilege_B \sqsubseteq Privilege \\ Privilege_A \text{ disjoint with } Privilege_B \\ priv\_equiv_{AB} : (Privilege_A \times Privilege_B) \\ priv\_equiv_{BA} : (Privilege_B \times Privilege_A) \\ priv\_equiv_{BA} \equiv priv\_equiv_{AB}{}^{-} \end{gathered} \tag{8}$$

Similar definitions to, say, (8) could be added for relating credentials, obligations and provisions between P_A and P_B; however, it is more convenient to combine the two policies using the ordering relations already present in the model. In the combined model, *obligation*, *provision* and *privilege* are orderings over the obligations, provisions and privileges of both organisations.

Further to these definitions, parts of the model must be closed. DLs work under the open-world assumption, meaning that a DL knowledge base defines a domain in terms of known concepts, roles and individuals but, also, that the definition is open and anything not explicitly stated, but consistent with the given definitions, may also be present in the domain. A closed-world assumption, on the other hand, takes the definitions as all there is to know about the domain.

For the purposes of this paper, the open-world assumption means that a DL reasoner may add to the policies and, in particular, further roles or comparabilites could be added. Parts of a knowledge base are closed by stating that the given concepts or instances are really all there is. For example, $Role_A \equiv \{a, b, c\}$ closes $Role_A$ by stating that the roles a, b and c are the only inhabitants. It is then necessary to specify that these inhabitants are distinct — essentially, adding a unique name assumption. Closing a relation r means explicitly stating that the set of (a, b) tuples specified for r are the only tuples in r. For the tests given in the remainder of this section, $Role_A$, $Role_B$, *possesses*, $role\_comp_{AB}$, $role\_comp_{BA}$, $priv\_equiv_{AB}$ and $priv\_equiv_{BA}$ all need to be closed. For the remaining concepts and relations, the tests take advantage of open-world reasoning and so closure is not necessary.

The following shows how to investigate inconsistencies between P_A and P_B, given that they are in a combined model that has been suitably closed.

4.1 Role Inconsistencies

Role inconsistencies between P_A and P_B are present when there are roles in one that have no comparable roles in the other. If P_A has roles with no equivalent in P_B, then this inconsistency indicates that, for at least those roles, P_A admits privileges that P_B does not and, hence, that P_A might allow some users privileges that P_B would not allow users with similar credentials.

The Missing Role Node concept, MRN_A, represents the roles in $Role_A$ that have no comparable role in $Role_B$. The concept B_comp is defined to be all the roles in $Role_A$ that have some comparable role in $Role_B$. $Role_A$ is defined such that all roles in organisation A must be classified as in MRN_A or B_comp.

$$\begin{aligned} MRN_A &\sqsubseteq Role_A \\ B\_comp &\equiv Role_A \sqcap \exists role\_comp_{AB}.Role_B \\ MRN_A &\text{ disjoint with } B\_comp \\ Role_A &\sqsubseteq B\_comp \sqcup MRN_A \end{aligned} \tag{9}$$

The combined effect of (9) is that all roles in $Role_A$ are classified into either MRN_A or B_comp and, if and only if all roles in $Role_A$ have some comparable role, MRN_A is inconsistent, otherwise some roles in $Role_A$ do not have comparable roles in $Role_B$ and these are classified in MRN_A. A similar definition to (9) is made for MRN_B, the roles in $Role_B$ with no comparable role in $Role_A$.

4.2 Credential Inconsistencies

There is an inconsistency in credentials when roles judged as comparable have different credential requirements. Such an inconsistency means that the two organisations have different requirements on what needs to be established before the privileges associated with a role can be accessed and could mean that equivalent roles in the two organisations have access to similar privileges but with a less stringent authorisation requirement in one organisation.

The goal is to establish if, for comparable roles, P_A has more stringent authorisation requirements than P_B (or similarly from P_B to P_A). The model has a relation for comparable roles and an ordering on credentials, but DL isn't expressive enough to directly encode the required property. In general, it isn't possible to directly express a property of the form“if a and b are related by r and a and b are related by r' and r'' to c_a and c_b respectively, are c_a and c_b related via r'''?” With the open-world assumption the required property can be expressed indirectly. Two things are required to express such a property in a DL: the first is the pattern in Figure 4, and the second is a technique often used with satisfiability which is to express properties negatively rather than positively.

Given the relations

$$\begin{aligned} r_1 &: (C_1 \times C_3) \\ r_2 &: (C_2 \times C_4) \\ r_3 &: (C_3 \times C_4) \end{aligned} \tag{11}$$

A new relation r relating individuals in C_1 and C_2 only if they are related through $r_1 \circ r_3 \circ r_2^-$ is defined by

$$r_1^- \circ r \circ r_2 \sqsubseteq r_3 \tag{12}$$

and by ensuring that

$$\begin{aligned} Domain(r) &\sqsubseteq C_1 \sqcap \exists r_1.C_3 \\ Range(r) &\sqsubseteq C_2 \sqcap \exists r_2.C_4 \end{aligned} \tag{13}$$

(12) and (13) ensure that r has only the right pairs. The correctness of this definition is shown by the following proof.

$$\begin{gathered} (c_1, c_2) \in r \\ \rightarrow \\ \exists c_3 c_4.(c_1, c_3) \in r_1 \wedge (c_2, c_4) \in r_2 \wedge (c_3, c_4) \in r_3 \end{gathered}$$

Proof. The domain and range restrictions in (13) are required to show the existence of a c_3 and c_4 from $(c_1, c_2) \in r$ and (12) is used to finish the proof.

□

While this scheme does project a relation, it may not capture the intended meaning if r_1 and r_2 are not functional. If r_1 and r_2 are not functional, (12) has the potentially unwanted side-effect of adding extra pairs to r_3, and there also isn't enough information in r to represent the multiple ways individuals in C_1 and C_2 might relate through the chain $r_1 \circ r_3 \circ r_2^-$.

Figure 4: **Pattern:** Projecting a Relation

First, define a relation disjoint with *satisfies*.

$$\begin{aligned} &not\_satisfies \ : \ (Credentials \times Credentials) \\ &not\_satisfies \text{ disjoint with } satisfies \end{aligned} \tag{10}$$

The relation *not_satisfies* is not the inverse of *satisfies*, it may not even relate everything that isn't related by *satisfies*, instead the open-world assumption means it can be any relation on *Credentials* that shares no tuples with *satisfies*. A DL reasoner is free to choose any relation satisfying this constraint in an attempt to satisfy the definitions below.

Next, the pattern in Figure 4 is used to project the *not_satisfies* relation between *Credentials* to a relation $nsat_{AB}$ between $Role_A$ and $Role_B$. In the pattern replace C_1 by $Role_A$, C_2 by $Role_B$, C_3 and C_4 by *Credentials*, r_1 and r_2 by *requires*, r_3 by *not_satisfies* and r by $nsat_{AB}$, giving definition (14). Since *requires* is functional, the projection relates roles if and only if their credentials are related by *not_satisfies*. Note that the reasoner will infer further restrictions on the stated types for relations such as $nsat_{AB}$ because of the restrictions on the other relations in (14). In fact, stating a domain and range for $nsat_{AB}$ isn't necessary at all; however, for clarity, the most comprehensive definitions of most concepts and roles are used in this paper.

$$\begin{aligned} nsat_{AB} &: (Role_A \times Role_B) \\ requires^{-} \circ nsat_{AB} \circ requires &\sqsubseteq not\_satisfies \end{aligned} \tag{14}$$

Lastly, $role\_comp_{AB}$ and $nsat_{AB}$ are combined. The requirement is for the combination, $role\_comp_{AB}\_nsat$, to relate two roles r_A and r_B if and only if $(r_A, r_B) \in role\_comp_{AB}$ and $(r_A, r_B) \in nsat\_role$, which is achieved by forming the intersection of $role\_comp_{AB}$ and $role\_comp_AB$.

$$\begin{aligned} role\_comp_{AB}\_nsat &: (Role_A \times Role_B) \\ role\_comp_{AB}\_nsat &\sqsubseteq role\_comp_{AB} \\ role\_comp_{AB}\_nsat &\sqsubseteq nsat_{AB} \end{aligned} \tag{15}$$

Two roles r_A and r_B are thus related by $role\_comp_AB\_nsat$ exactly when r_A and r_B are comparable and the credentials required for r_A are not more stringent than those required for r_B.

To prove that the required meaning is captured by $role\_comp_{AB}\_nsat$ requires showing (in the underlying semantics)

$$\begin{aligned} (r_A, r_B) \in role\_comp_{AB}\_nsat \quad &\rightarrow \\ (r_A, r_B) &\in role\_comp_{AB} \wedge \\ \exists c_A, c_B.(r_A, c_A) &\in requires \wedge \\ (r_B, c_B) \in requires \wedge (c_A, c_B) &\notin satisfies \end{aligned}$$

the proof of which follows easily from the proof in Figure 4 and the definitions of $role\_comp_{AB}\_nsat$ and $nsat_{AB}$.

A DL reasoner will classify $role\_comp_{AB}\_nsat$ as unsatisfiable when all roles in P_A have stricter credential requirements than the corresponding roles in P_B: that is, the credential requirements for each role in $Role_A$ also satisfies (via *satisfies*) the credential requirements for the corresponding roles in $Role_B$. If this is not the case, a DL reasoner will classify pairs of roles into $role\_comp_{AB}\_nsat$ where the credential requirements for the role from $Role_A$ do not also satisfy the requirements for the roles from $Role_B$.

4.3 Inconsistencies in Privileges

He & Yang (2007) discussed three inconsistencies for privileges: inconsistencies in the privileges allocated to comparable roles and two types of inconsistencies in the conditions associated with comparable privileges for comparable roles. The four tests to determine the presence of these inconsistencies given by He & Yang are fine-grained enough to pinpoint an inconsistency between comparable roles and the privilege causing the inconsistency. However, the combined effect of the four tests is, essentially, equivalent to determining if comparable roles in the two collaborating parties have equivalent privileges and, if for equivalent roles and equivalent privileges, one party has weaker conditions than the other.

Here, only a test to determine if equivalent roles have comparable conditions for equivalent privileges is discussed. The roles and privileges causing an inconsistency can still be recovered.

The *Pair* concept (16) defines a pair (a, b) as an individual p, with $Pair(p)$, $(p, a) \in left$ and $(p, b) \in right$.

$$\begin{aligned} right^{\mathcal{F}} &: (Pair \times Thing) \\ left^{\mathcal{F}} &: (Pair \times Thing) \\ Pair &\sqsubseteq \exists left.Thing \\ Pair &\sqsubseteq \exists right.Thing \end{aligned} \tag{16}$$

Figure 5: **Pattern:** Pairs

4.3.1 Different Conditions

If the conditions on equivalent privileges allocated to comparable roles are different, then these roles in the two organisations access a privilege with different, perhaps incompatible, restrictions. However, if the difference is one were the conditions can be meaningfully compared, then the comparison will reveal that one role accesses the privilege under stronger restrictions than the other.

The situation is similar to credentials, in that the test ultimately relies on the orderings between conditions; however, the relations between both privileges and roles also need to be taken into account. Further, the credential checking relied on the functional relation *requires*; the projection pattern used with *requires* is not applicable for *possesses* because *possesses* is not functional. For non-functional relations a projection onto a single relation can't represent all the possible relationships: for example, each role may relate through *possesses* to a number of *PrivilegeAssignment*s, so a single relation between *Role*s can't capture the many possible relationships to privileges and their associated conditions. However, the same general principle can still be used if *possesses* is first converted to a concept.

*Pair*s (Figure 5) can be a more convenient representation of a relation than the corresponding DL role because the expressions allowed on roles and concepts are different. In this case, using the pairs-for-relations pattern (Figure 6) for *possesses* introduces functional relations *left* and *right* and, thus, the relations between roles, privileges and conditions can be projected as a relation between *Pair*s of *Role*s and *PrivilegeAssignment*s.

A relation, $pa\_eq\_nord_{AB}$, that relates *PrivilegeAssignment*s from P_A and P_B when the *Privilege*s are related, but the conditions are not, is defined by

- projecting both *obl_order* and *prov_order* to a $(PrivilegeAssignment \times PrivilegeAssignment)$ relation (see Figure 4 and 14)
- forming a subrelation of both the above two relations (as was done in (15)) and then defining $not\_prov\_obl\_order_{AB}$ as a relation disjoint from this (see also (10))
- projecting $priv\_equiv_{AB}$ to a relation between *PrivilegeAssignment*s (Figure 4) and defining $pa\_eq\_nord_{AB}$ as a subrelation of this and $not\_prov\_obl\_order_{AB}$ (again, see (15)).

Pairs *Pair_possess_A* and *Pair_possess_B* are defined (using the pattern in Figure 6) as pairs representing the *possesses* relations for P_A and P_B.

Relations $role\_comp_{AB}$ relates comparable roles in $Role_A$ and $Role_A$, $pa\_eq\_nord_{AB}$ relates *PrivilegeAssignment*s and the specialisations of *left* and *right* on *Pair_possess_A* and *Pair_possess_B* are functional. The projection pattern (Figure 4)

Pairs, as defined in (16) (Figure 5), can be used to represent DL roles. DL roles (binary relations) are simply sets of (a, b) pairs; hence, if *left* and *right* from *Pair* are used to represent the domain and range, a set of *Pair* individuals can represent the information in a role. The same principle applies for *PrivilegeAssignment* (2), which represents a ternary relation. A *Pair* representation of a relation can even be derived from an existing relation. Given

$$r \ : \ (C_1 \times C_2) \tag{17}$$

a sub-relation can be derived.

$$\begin{aligned} &\mathit{Pair\_r} \sqsubseteq \mathit{Pair}\\ &\quad \mathit{left\_r} \sqsubseteq \mathit{left}, \mathit{right\_r} \sqsubseteq \mathit{right}\\ &\mathit{Pair\_r} \sqsubseteq \exists \mathit{left\_r}.C_1, \mathit{Pair\_r} \sqsubseteq \exists \mathit{right\_r}.C_2 \end{aligned} \tag{18}$$

$$\mathit{left\_r}^{-} \circ \mathit{right\_r} \sqsubseteq r \tag{19}$$

The proof that the relation represented by the pair is contained in r follows easily from definition (18), which makes *Pair_r* a type of pairs of C_1 and C_2 individuals, and definition (19), which ensures that if $(p, c_1) \in left\_r$ and $(p, c_2) \in right\_r$, with $Pair\_r(p)$, then $(c_1, c_2) \in r$.

Figure 6: **Pattern:** Pairs for Relations

can thus be used to project $\mathit{role\_comp}_{AB}$ and $\mathit{pa\_eq\_nord}_{AB}$ to relations between *Pair_possess_A* and *Pair_possess_B*. Finally, these two relations are combined (as was done in (15)) to produce $\mathit{rcomp\_pa\_nord}_{AB}$, which is unsatisfiable if the privileges for comparable roles have more stringent constraints in P_A than in P_B and contains *Role-PrivilegeAssignment* pairs with uncomparable conditions otherwise.

5 Access Control Policy Requirements and Example

This section first examines the inconsistencies to consider for the Service Propagation pattern and then discusses and example of using a Description Logic reasoner to analyse policies for collaboration.

5.1 Access Control Policy Requirements

Several cross-organization collaboration patterns were identified previously (He & Yang 2007), and some of these were reviewed in Section 2. The different collaboration patterns result in different consistency requirements on the policies of the prospective collaboration partners. The inconsistencies between policies can result in collaboration being accepted, rejected or can require negotiation to remove inconsistencies. This paper focuses on Service Propagation and on inconsistencies that can be automatically accepted or rejected.

Service Propagation implies a 'forwarding' behaviour from the service owner to the collaborative partner. Unwanted 'forwarding' could happen if the partner has authorization policy that is less restrictive than the owner's. Therefore, policy of collaboration partner should be more restrictive than the service owner's. There are three typical ways to have one policy looser than the other:

- One set of policies has more roles to access the same service than the other set of policies;
- One set of policies has less credentials required for an equivalent role than the other set;
- For an equivalent role, more privileges, or privileges with weaker conditions, are assigned in one set of policies than the other set.

Based on above principles, the following defines the requirements on collaboration partners for Service Propagation. The requirements are in terms of the definitions in the previous section the following assumes organisation A is the service owner and B is the collaborator.

1. Every role defined by the partner should have a a comparable role in the service owner. In terms of Section 4.1, concept MRN_B (not given in the text, but similar to MRN_A) should be unsatisfiable;
2. The credentials required for equivalent roles should be be more restrictive in the partner: $\mathit{role\_comp}_{BA}\mathit{\_nsat}$ should be unsatisfiable;
3. More privileges should be granted to roles in P_A than their equivalents in P_B (not discussed in previous section);
4. Inconsistencies in conditions are generally negotiable, but if the collaboration partner has stricter conditions, then the collaboration is acceptable: that is, $\mathit{rcomp\_pa\_nord}_{BA}$ unsatisfiable.

5.2 Example

The following example demonstrates how an access control policy can be presented in the DL policy model and how the inconsistencies between the access control policies of collaboration partners can be evaluated using a DL reasoner.

If services and their policies are listed in a repository, a service searching for a collaboration partner can search the repository for a service with the required functionality and then check if the access control policies are compatible. Assume a medical centre that requires a collaboration with a pathology centre. The two will collaborate on the tests of patients, their records and the results of tests. The centre will search for a medical service registry for potential pathology collaborators and test the access control policies to find a suitable partner. The policies of the medical centre and two potential collaborators are shown below.

1. Medical Clinic:
   - Attending doctors have the privilege to access and forward patient information;
   - a provision is attach to the forward privilege: receipients must be a doctor in the chosen pathology institute.

   A policy must be defined with individuals classified as follows: $\mathit{Role}(\mathit{mc\_doctor})$, $\mathit{Credentials}(\mathit{mc\_doctor\_id})$, $\mathit{PrivilegeAssignment}(\mathit{mc\_pa})$, $\mathit{Privilege}(\mathit{access})$, $\mathit{Privilege}(\mathit{mc\_forward})$, $\mathit{Provision}(\mathit{to\_partner\_pathology\_doctor})$ and $\mathit{Obligation}(\mathit{no\_obligation})$. With $(\mathit{mc\_doctor}, \mathit{mc\_doctor\_id}) \in \mathit{requires}$, $(\mathit{mc\_doctor}, \mathit{mc\_pa}) \in \mathit{possess}$, $(\mathit{mc\_pa}, \mathit{mc\_forward}) \in \mathit{privilege}$, $(\mathit{mc\_pa}, \mathit{no\_obligation}) \in \mathit{obligation}$, $(\mathit{mc\_pa}, \mathit{to\_partner\_pathology\_doctor}) \in \mathit{provision}$ and a privilege assignment for *access*.

2. Pathology institute X:

   - Attending doctors have privilege forward patient information;
   - a provision attach to forward privilege: receipients must be doctors in X

   The policy definitions are as follows: $Role(doctor_X)$, $Credentials(doctor\_and\_path\_id)$, $PrivilegeAssignment(pa_X)$, $Privilege(forward_X)$, $Provision(to\_X\_pathology\_doctor)$ and $Obligation(no\_obligation_X)$. With $(doctor_X, doctor\_and\_path\_id) \in requires$, $(doctor_X, pa_X) \in possess$, $(pa_X, forward_X) \in privilege$, $(pa_X, no\_obligation_X) \in obligation$, $(pa_X, to\_X\_pathology\_doctor) \in provision$.

3. Pathology institute Y:

   - Attending doctors have privilege to forward patient information;
   - a provision is attach to forward privilege: receipient must be either a doctor in Y or staff in a collaborating research institute.

   The policy definitions are similar to above, with the important provision $Provision(to\_Y\_pathology\_doctor\_or\_research)$.

To find a suitable collaboration partner, the medical centre must test its policy with pathology institutes X and Y. The policies are tested in the combined model; one test with institute X and one with Y. Assume in both cases that the medical centre is policy A and that each pathology institute is policy B (in terms of A and B as discussed in the previous section). The relationships for $role\_comp_{AB}$ and others are straightforward, importantly provision $to\_X\_pathology\_doctor$ is more restrictive than $to\_partner\_pathology\_doctor$, hence $(to\_X\_pathology\_doctor, to\_partner\_pathology\_doctor) \in prov\_order$, while $to\_Y\_pathology\_doctor\_or\_research$ is less restrictive than the provision in the medical centre and so is not related by $prov\_order$.

In the comparison with institute X the reasoner proves the required concepts as unsatisfiable, and thus shows that Pathology institute X is a suitable partner for collaboration. However, in testing with institute Y, the reasoner proves that concept $rcomp\_pa\_nord_{BA}$ is satisfiable (because Pathology institute Y allows extra forwarding privileges), and thus shows that Pathology institute Y is not a suitable collaboration partner.

Patients could test if the policy of the medical centre satisfies their requirements and make their choice of suitable medical services based on these policy tests.

The models, tests and example in this paper have been encoded in $\mathcal{SROIQ}$, using Protégé, and the reasoning done with Pellet. The files are available on request.

6 Related Work

Research has been done in the area of access control / authorization control for web services. Most of the works concentrate on authorization control policy language specifications and a number of formal models have been developed (Sirer & Wang 2002, Kagal et al. 2004, Bhatti et al. 2004, Srivatsa et al. 2007). These studies provide insights on security constraints in single organization from different perspectives, which helped us to build up our authorization policy model. But these studies did not look at authorization issues in the context of cross-organization collaborations.

Policy issues in distributed systems have been actively growing over the years, Bonatti & Mogavero (2008) proposed an inclusion mappings based policy comparison method, which could be useful for rule-based policis, particularly for recursive rules. Wang et al. (2004) addressed security policy reconsiliation issues in distributed computing environments, and focus on security provisioning policy. The authors based their reconsiliation on structure of the policies, which is similar to our work. It also provides an alternative for policy specification. However, our focus in this paper is policy comparison and evaluation rather than reconsiliation and we address authorization policies.

A number of studies concentrated on authorization architecture (B.Carminati et al. 2006, Ziebermayr & Probst 2004). B.Carminati et al. (2006) suggested a brokered architecture to build composite Web services according to the specified security constraints. They used security matchmaker to find right collaboration partners who have compatible security policies, which is similar to our research. However, it did not address the issue that inconsistencies and conflicts exist between security policies of prospective partner. Nothing was mentioned about how security policies from different partners could be combined and how to solve the conflicts between these policies. Our work identified various types of inconsistencies between authorization policies and provided suggested solution. We believe some of inconsistencies are acceptable or negotiable for intended type of collaboration.

There are few papers on Web service authorization control in the collaborative environment. We are aware of the work presented by Rouached & Godart (2007), which presented a framework for managing authorization policies for Web service compositions. The proposed framework addressed authorization policy conflicts and provided methodology for conflicts detection. Yau & Chen (2008) proposed an approach to security policy integration and conflict reconciliation, which is relate to our research. The authors presented a similarity-based policy adaptation algorithms to adapt changes in collaborative groups and a negotiation-based protocol for conflict reconciliation But they neglected the fact that different types of collaboration affect the way the collaboration policy is developed as well as the requirements on collaborative partner's authorization policy. An evaluation on collaborative partner's access policy has to be carried out before the collaboration be established. Our work is to fill in this gap. We believe this is the first step toward conflicts detection and suitable collaboration partners discovery.

Description Logic (DL) has been used in differenct aspects of access control (Chae & Shiri 2007, Shields & Molloy 2007, Zhao et al. 2005, Muthaiyah & Kerschberg 2006). Shields & Molloy (2007) proposed an efficient solution to XML access control that use description logic and decidable rules to decide the permissions for an individual at the time they request information. Chae & Shiri (2007) demonstrated how to express the RBAC concept in object-oriented systems using description logic and how to use DL to make authorization decisions between the role hierarchy and the object hierachy. All of these studies are only focus on access control in single organization, access control in collaborative environment has not been well studied.

In summary, none of these studies went deep into different types of cross-organization collaboration, which could raise different requirements on access control policy of prospective collaborative partners. None of studies provided tool for access policy comparison and evaluation. Our goal is to identify these different requirements for different types of cross-organization collaboration, to analysis inconsistencies between policies and provide suggestions and solutions to inconsistencies according the collaboration type, to propose an access control model and automatic prover that could assist organizations to discover compatible collaboration partners.

7 Conclusion/Discussion

Challenging security and access control issues arise in Web Services. In particular, when cross-organisation collaborations are formed, the access control policies of the collaboration partners must be inspected. If the policies are not compatible, the collaboration does not respect the access control policy of at least one of the partners.

In this paper we have presented a Description Logic model for access control policy. Based on this model, we have demonstrated how a Description Logic reasoner can be used to find the inconsistencies between the policies of potential collaborators. The reasoner constructs a proof demonstrating the consistency, or otherwise, of the policies by reasoning about tests that we added to the policy model. With formal descriptions of access control policy and different policy inconsistencicy types, we can evaluate prospective collaborators' access policies against requirements for requested collaboration pattern and classify any inconsistencies.

The method can also be use to analyse the impact changes in policy will have for collaboration partners.

Description Logics are mathematically simple, but that does not mean that Description Logic definitions are not complex; in particular, the effect any definition has on others is not always clear. Because each definition is made in terms of restrictions that may affect other definitions, we have chosen to verify that our definitions have the intended meaning by proving that definitions are correct once translated to the model theoretic semantics.

In the future we intend to extend this work to incorporate the following:

- Refine cross-organization collaboration patterns from different perspectives, for example, from business prosess point of view.
- Take context constraints that could affect access control into consideration, e.g. access time or access location.
- Discuss Role Based Access Control issues under collaborative context, e.g. role hierarchies and separation of duty in collaboration environment.

References

ANSI INCITS 359-2004 (2004).

B.Carminati, Ferrari, E. & Hung, P. (2006), Security conscious web service composition, *in* 'IEEE International Conference on Web Services', Chicago, Illinois, USA, pp. 489–496.

Bhatti, R., Bertino, E. & Ghafoor, A. (2004), A trust-based context-aware access control model for web-services, *in* 'IEEE International Conference on Web Services', San Diego, CA, pp. 184–191.

Bonatti, P. A. & Mogavero, F. (2008), Comparing rule-based policies, *in* '9th IEEE International Workshop on Policies for Distributed Systems and Networks', New York, USA, pp. 11–18.

Chae, J.-H. & Shiri, N. (2007), Description logic framework for access control and security in object-oriented systems, *in* 'RSFDGrC', pp. 565–573.

He, D. D. & Yang, J. (2007), Security policy specification and integration in business collaboration, *in* '2007 IEEE International Conference on Services Computing (SCC 2007)', Salt Lake City, Utah, USA, pp. 20–27.

Horrocks, I., Kutz, O. & Sattler, U. (2006), The even more irresistible $\mathcal{SROIQ}$, *in* 'Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006)', AAAI Press, pp. 57–67.

Kagal, L., Paolucci, M., Srinivasan, N., Sycara, K. & G.Denker (2004), 'Authorization and privacy for semantic web services', *IEEE Intelligent Systems* **19**(4), 50–56.

Muthaiyah, S. & Kerschberg, L. (2006), Dynamic integration and semantic security policy ontology mapping for semantic web services (sws), *in* 'ICDIM', pp. 116–120.

OWL 1.1 Web Ontology Language (2006), W3C Member Submission. Available at `http://www.w3.org/Submission/2006/10/`.

Rouached, M. & Godart, C. (2007), Reasoning about events to specify authoriztion policies for web services composition, *in* 'Proceedings of 2007 International Conference on Web Services', IEEE, Salt Lake City, UT.

Sandhu, R. S., Coyne, E., Feinstein, H. & Youman, C. (1996), 'Role-based access control models', *IEEE Computer* **29**(2), 38–47.

Shields, B. & Molloy, O. (2007), Using description logic and rules to determine xml access control, *in* 'DEXA Workshops', pp. 718–724.

Sirer, E. G. & Wang, K. (2002), An access control language for web services, *in* 'SACMAT', pp. 23–30.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A. & Katz, Y. (2007), 'Pellet: A practical OWl-DL reasoner', *Journal of Web Semantics* **5**(2), 51–53.

Srivatsa, M., Iyengar, A., Mikalsen, T., Rouvellou, I. & Yin, J. (2007), An access control system for web service compositions, *in* 'Proceedings of International Conference on Web Services', IEEE, Salt Lake City, UT.

Wang, H., Jha, S., Livny, M. & McDaniel, P. D. (2004), Security policy reconciliation in distributed computing environments, *in* '5th IEEE International Workshop on Policies for Distributed Systems and Networks', pp. 137–147.

Web Ontology Language (OWL) (2004), W3C Recommendations. Available at `http://www.w3.org/2004/OWL/`.

Yau, S. S. & Chen, Z. (2008), Security policy integration and conflict reconciliation for collaborations among organizations in ubiquitous computing environments, *in* 'UIC', pp. 3–19.

Zhao, C., Heilili, N., Liu, S. & Lin, Z. (2005), Representation and reasoning on rbac: A description logic approach, *in* 'ICTAC', pp. 381–393.

Ziebermayr, T. & Probst, S. (2004), Web service authorization framework, *in* 'Proceedings of the IEEE International Conference on Web Services (ICWS'04)', San Diego, CA, pp. 614–621.

Mobile Information Exchange and Integration: From Query to Application Layer

Van T.K. Tran **Raymond K. Wong** **William K. Cheung**[α] **Jiming Liu**[α]

School of Computer Science & Engineering
University of New South Wales
and National ICT Australia
Emails: trantkv@cse.unsw.edu.au
raymond.wong@nicta.com.au

[α]Department of Computer Science
Hong Kong Baptist University
Emails: william@comp.hkbu.edu.hk
jiming@comp.hkbu.edu.hk

Abstract

Due to the popularity of mobile devices, more and more commercial applications have been developed on these devices. While commercial applications are mostly backed by relational database systems, numerous database engines have been ported to or built on these devices, for example, SQLite. Since connectivity can be unstable or slow, applications such as iAnywhere have considered offline operations while data can be synchronized with the database server whenever the devices are online. On the other hand, while Web-based and XML content are very common these days, unfortunately, these mobile versions of database engines failed to fully support them. This paper considers a translation-based method with a decentralised versioning system in place to support offline operations. Web and XML contents are stored and versioned in a distributed manner and can be synchronized with each other without connecting to a server. The schema of these data can be automatically generated on device. With these schema, a translation engine which allows querying these data using SQL by translating the query to a corresponding XML query is facilitated. We believe this framework support mobile data applications on XML or Web data in a seamless manner. Finally, an initial prototype has been implemented and described in this paper.

Keywords: Decentralised Systems, Version Control, Mobile Data Management, Heterogeneous Information Exchange, Information Integration, XML, Query Translation, Schema Inference.

1 Introduction

The success of mobile devices and wireless communication technology has enabled various applications and activities to be ported from desktop platform into mobile fashion. Although these activities are widespread through a variety of disciplines, from professional to leisure, no matter what environment they are in, collaborative attribute plays a very important and essential part to improve performance and productivity. With the tremendous support of mobile technology, work environments have become very flexible and convenient. A group of people need neither stay at the same place to collaborate, nor attach to their PCs for their work. They can work and cooperate from anywhere as it is possible and convenient for them using their mobile devices.

Working in a collaborative manner, each person works on his/her own separated tasks, yet still closely related to his/her colleagues' tasks. It is also very likely that several people may have to work on a shared document. In such cases, it is essential for one to keep others up-to-date with the status of his/her work and vice versa. Before one makes a change to the shared document, he/she has to make sure that it is an updated document, so that changes are unlikely to be undone and redone. In other cases, the shared document may need to be reverted to its previous state. To manage all these scenarios of collaboration, version control systems are of great help.

In a collaborating environment, conflicts are unavoidable. When two collaborators attempt to update the shared document at the same time, conflicts could occur. Conflicts are generally resolved via communication between collaborators or by decisions of group leaders. The latter is less preferable for its inflexibility. The system should be flexible enough to allow two group members involved in the conflicts to discuss, reach a consensus and make a decision on how to control shared document versions. However, given a mobile teamwork condition, especially when communication is not as instant and convenient as in face-to-face manner, concurrency control is challenging.

Different architectures have been introduced and investigated to support work collaboration in mobiles, including centralised and decentralised systems. Centralised systems are server-client systems where all mobile clients are connected through a central server. The central server is responsible for controlling all communications and activities amongst mobile clients. Central servers and central authorities, however, are not always available for mobile collaborations. In such a situation, the concept of decentralised system appears to be very useful. Connection preference and quality sometimes also make decentralised systems more attractive than centralised systems. Version control in centralised systems is quite straightforward since all controls and management can be done in server-side, which is not much different from applications for desktop platforms. On the other hand, version control in decentralised mobile systems is still a challenge.

There have been several decentralised version control studies around (Ellis & Gibbs 1989, Munson & Dewan 1996, Suleiman et al. 1998, Ionescu et al. 1999, Jiang et al. 2005, Su et al. 2007). These systems mainly focus on the decentralised aspect that a node can work without connection to the server. In other words, it is not necessary to connect to a server all the time for one to get work done. Instead he/she can just work on the local version and synchronise it with the server later when there is a connection. This paper, on the other hand, proposes a version control framework focusing on another feature of decentralisation, in which nodes can share documents in a peer-to-peer manner and without a central authority. We believe that this framework is much more suitable for mobile devices.

Inspired by the fact that Web-based and XML contents are very common in these days and, yet the mobile versions of database engines (mostly SQL-based, e.g., SQLite) failed to fully support them, this paper also investigates the notion of automatically generating the schema of these data on devices and introduces an XML Intelligent Agent to enable information exchange and information integration between heterogeneous data sources among the local repositories from individual mobile devices. Using the agent, a data source can query multiple data sources in its native query language and integrate the results obtained without any knowledge of the data representation, format or query language of the other heterogeneous data sources. The agent uses XML and iQuery (a derivative of XQuery) to represent queries and results. A mobile application queries the agent in a industry common query language (at the moment, most mobile data applications are still based on a subset of SQL). The agent translates the query to iQuery and routes the iQuery (or part-of) to other agents capable of processing the query. These agents translate the query to some native query language, execute it on the heterogeneous data source, obtain the result and return an XML representation of the result to the source agent. The source agent integrates all results and translates the final XML result representation to the data source's native result format (e.g., tabular, relational format) before returning it to the mobile applications.

This paper is outlined as follows: Section 2 gives an overview of related work in decentralised version control systems, query wrappers, and reference structural inference methods for XML data. Section 3 describes the motivating scenario that leads to the framework proposed and described in Section 4. Section 5 provides some experimental results to justify the proposed framework. Section 6 concludes the paper and discusses future work.

2 Related work

2.1 Version control and decentralised systems

Decentralised systems can be distributed or replicated systems. For either distributed or replicated systems, the copies of each object at all nodes must be kept consistent. Suleiman et al. (1998) presented an operational transformation algorithm that based on the notion of user's intention and using semantic properties of operations called forward and backward transpositions to serialise concurrent operations, in order to maintain the consistency amongst replicated copies. The forward and backward transpositions enable the equivalent histories to be characterised, in which forward transposition resolves the problem of concurrency operations and backward transposition changes the order of operations in the history without violating user's intention.

Jiang et al. (2005) proposed a semi-replicated architecture to maintain the consistency of the replicas in mobile environments. In this architecture, a central server acts as the agents of mobile sites and is used to backup a copy of the shared object, while mobile devices with limited resources hold only parts of the object. With this architecture, the agents that reside in the central server take full responsibility for managing operations across mobile sites, and preserving consistency of the replicated documents.

The SVK version control system introduced in (Kao 2003) is a decentralized version control system built with the robust subversion filesystem. It supports repository mirroring, disconnected operation, history-sensitive merging, and integrates with other version control systems, as well as popular visual merge tools. In other words, SVK is a way to work around the centralised design of SVN (CollabNet 2006) and should be seen as an extended client, not a replacement (Robert 2006). This SVK system is then used as a base system for our prototype presented later in this paper.

Many conventional solutions for version control on those decentralised systems have involved locking approaches. Munson & Dewan (1996) developed a framework based on a locking algorithm to prevent concurrent operations. In this framework, a user requests a lock on a particular object and the lock is held until the end of the transaction. In the meantime, other users can only read and receive updates but not modify that object. This algorithm focuses on conflict prevention rather than conflict management. Citro et al. (2007) overcame the above limitation by introducing a delayed post-locking algorithm. This algorithm is a variation of the conventional post-locking algorithm, in which the modified object is automatically locked when a conflict occurs.

The problem of locking approaches is the decreasing level of concurrency. Chianese et al. (2008) improved concurrency of the locking approach by considering frequent disconnection or inactivity periods of transaction. Besides locking approach, other strategies include operational transformation and multi-versioning (Suleiman et al. 1998, Citro et al. 2007, Ellis & Gibbs 1989). These strategies are aimed to manage and resolve conflicts in version control systems.

Generally, those above systems mainly focus on how a node can operate on its own without a central sever or how to resolve conflicts at a node, but they do not mention how nodes communicate and collaborate with one another. Ionescu et al. (1999) addressed this issue by introducing a replication architecture, in which if a change is made in a local object of a node, all other nodes will be notified to change accordingly. To use network resources efficiently, Su et al. (2007) designed an integrated consistency-control algorithm to decide a limited number of nodes get updated upon any changes in the network, instead of all nodes. This algorithm is defined based on the probability of contents selection and node update.

2.2 Query translation and information exchange

Many efforts have been invested in data conversion & query translation for information exchange and/or integration. For instance, NoDoSe (Adelberg 1998) can semi-automatically process input file using user-defined schema and GUI to specify a region in the input file for each object. It has an addition HTML parser specifically for HTML files. W4F wrapper (Sahuguet & Azavant 1999) also focus on parsing HTML to XML. It relies on the structure of HTML and its extraction language HEL is based on DOM.

The extraction rules introduced by Hammer et al. (1997), based partly on regular expression and nested structure of HTML documents, have variables for storing extracted data. Similar to (Hammer et al. 1997), as a wrapper generation tool, XWRAP (Liu et al. 2000) provides a component-based library to be used by the generated wrappers and there is an inductive learning based mechanism to determine the patterns of the document structures for wrapping. Ashish & Knoblock (1997) proposed an approach to automate the process of generating an extractor that can recognize the structure of HTML. This is done by identify tokens using HTML tags and regular expression. The nested structure of the input is detected using heuristics such as font size of heading and indentation.

In addition to data format transformation (which is needed to transform the source data and/or the output of the query to a required format), one major focus of this paper is on query language translation. Most recent efforts have been focused on translating XML queries to SQL, due to many proposals on implementing XML databases using relational database systems. For example, Krishnamurthy et al. (2004) presented an efficient method to translate XML query to SQL. Other work including (Yu 2004) focusing on rewriting the queries into a different form to facilitate efficient data integration. Different from most of these related work, since most mobile data applications have been built using an underlying SQL data access layer and most Web/mobile content are stored natively as XML or its related format, we present a query translation agent from SQL to XML. We argue that querying XML data using SQL is fundamentally more challenging and requires more 'intelligence'. This is due to the fact that the formation of an SQL query requires a good knowledge of a fixed, predefined schema. However, schemas are usually not available or stored with the content in the mobile devices. Even they are available, due to the heterogeneity of the data, it would be desirable to maintain a integrated version of the schemas of the stored content so that queries involving different content can be supported. Therefore, a lightweight, schema generation mechanism is implemented.

2.3 Structural schema inference

In order to provide a good and lightweight schema inference mechanism for the needs described above, we need to evaluate different alternatives. While most of the schema inference methods cost a similar amount of time to generate a schema, their degrees of accuracy vary. Numerous algorithms are implemented for evaluation in this paper. Due to the number of algorithms implemented, full descriptions of them are not presented here. Further description of these algorithms can be found from (Sankey & Wong 2001), and some related literature is included below.

The first known paper to address DTD generation using tradition grammatical inference methods was proposed by Ahonen (1996). The two methods proposed there are theoretically appealing, as they guarantee to infer languages falling within certain language classes. These classes are termed k-contextual and (k, h)-contextual, so named as they assume the structure to be inferred to have limited context. Another method applied to DTD generation (Young-Lai 1996), is derived from more recent work in grammatical inference. The base algorithm is known as Alergia, introduced in (Carrasco & Oncina 1994). The criterion for state equivalence in Alergia is based upon observed frequencies of transitions and finalities of a pair of states. The most recent work on schema inference by Bex et al. (2008) focused on a probabilistic algorithm that learns k-occurrence regular expressions for increasing values of k, and selects the one that best describes the sample based on a Minimum Description Length argument.

3 Motivating scenario

This section discusses a scenario that describes the motivation for our work. The scenario explains how our work is worthwhile. Let us consider a system consisting of two mobile nodes and a central server as in Figure 1. In this system, the version control function is done in the central server, placed within the Intranet of a company. The two client nodes collaborate with each other via the control of the server.

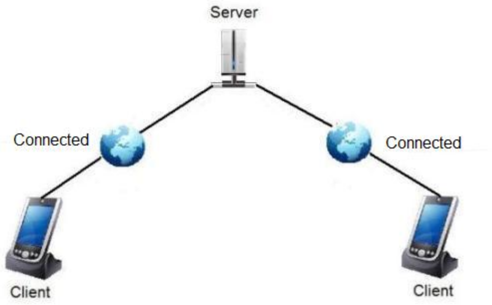


Figure 1: Centralised system

Let us now consider the situation when two employees of the company go overseas to demonstrate their product to a customer. The employees use these two mobile devices for the demonstration. Before the demonstration, one of them recognises that some changes need to be made in one mobile device, and the changes need to be updated in the other mobile. Because of the connection problem, they cannot connect to the server in the Intranet of the company back home, meanwhile they are located close to each other and good connection can be established with short-range transmission protocols like Bluetooth. It makes more sense to have them communicate directly with each other. In such a situation, the concept of decentralised system as in Figure 2 appears to be very useful.

The version control function implemented in this decentralised system will support the synchronization and update process between the two clients.

Furthermore, XML has grown in popularity as a Web publishing format, as a messaging format, as a data

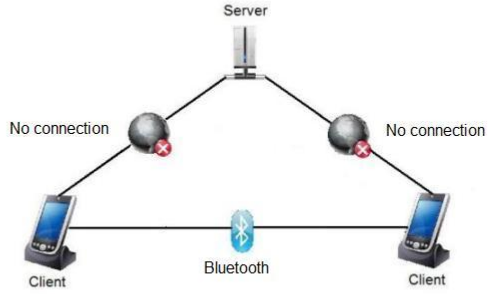


Figure 2: Decentralised system

exchange format and as a means of storing data. Data stored in XML format preserves the data representation, structure and its semantic meaning. Hence, more and more content stored in mobile devices are in XML. However, most mobile data management applications (e.g., iAnywhere) rely on an SQL access interface to the underlying content storage. Therefore, there is a need for a query wrapper that allows querying multiple XML repositories using SQL. It should also be possible for a mobile user to query / integrate multiple heterogeneous content files (possibly located in different devices) and use them as if they were a single data source, with a single global schema.

4 Proposed mobile framework

This section describes in detail our contribution, particularly:

- How mobile information is exchanged and integrated in the mobile application layer with our proposed decentralised version control framework (section 4.1)
- Down to the SQL data access layer, information exchange and integration amongst SQL databases have been studied quite intensively. However, most mobile contents are stored in XML or its related formats. This paper, therefore, focuses on the translation from SQL to XQuery to enable information exchange and integration amongst heterogeneous data sources in XML format using SQL (section 4.3)
- To achive this aim, schema generation for the underlying XML content is needed; therefore, a good and lightweight structural inference for XML data is required (section 4.2)

4.1 Decentralised version control framework

The proposed framework is a decentralised version control system (figure 3) that focuses on collaboration and consistency maintenance. The nodes communicate with one another in a peer-to-peer manner, given that good connection can be established with short-range transmission protocols like Bluetooth. Essentially, each node performs both server and client functionalities and behaviours of a version control system.

In a general version control system, the main and most popular functions are *checkout*, *update*, and *commit*. Within a traditional decentralised system, all nodes are treated equally. In other words, when a node performs *commit*, changes are applied to all other nodes. When a new node is added to the system, it can *checkout* from any random node in the system. *Update* is not really necessary here, since every node is kept current all the time.

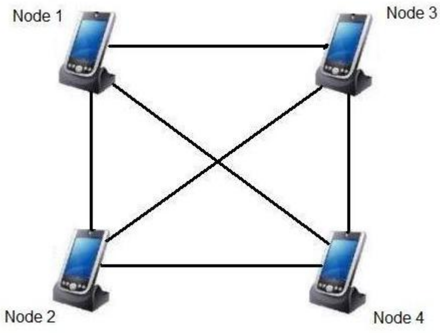


Figure 3: Proposed framework

However, not all nodes need to be kept updated all the time, since some of them may be just used for one-off tasks. To increase performance and effectiveness, our proposed framework consists of nodes not treated equally and not storing exactly replicated data. Particularly, when a node performs a change, it *commits* to its own server first, then signals other nodes that a new change has occurred in the system; however, only a few algorithm-decided peer nodes are notified to update their replicated documents accordingly. Other nodes will get updated upon requests. On the other hand, when a node requests to *checkout* or *update*, the requests will not be sent to any random nodes in the system, but to an algorithm-decided node.

In order to obtain those desired algorithm-decided nodes, our framework introduces the concept of *state table* and *active peer*, and then explain in details how several common functions (*checkout*, *update*, and *commit*) fit into this framework.

4.1.1 Active peers and state table

A *node j* is considered to be active when it gets updated, receives requests from other nodes, and makes requests to other nodes regularly. The active level of a node to another node is represented by a property called *active rate*. To decide the *active rate*, a *state table* is used. Each node maintains a *state table* for all of its connected peers. The *state table* of a *node i* consists of the identifier of any *node j* connected to it, the latest revision of *node j*, and the total number of made requests and received requests of *node j*. The *state table* also keeps a special field R_{all} representing the overall latest revision of the whole system, in order to identify its status (current or outdated) within the system. The *state table* must be updated regularly to ensure each node has its peers' latest information.

Let

- R_i denote the latest revision of *node i*
- R_j denote the latest revision of *node j*
- N_j denote the total number of made requests and received requests of *node j*
($N_j \geq 1 \ \forall j$ since every node must checkout at least one when first connecting)

- $A_i(j)$ denote the active rate of *node j* to *node i*

The parameters used in this algorithm can be easily located, collected and stored in the *state table*. The number of made and received requests of a node N_j illustrates its popularity and interaction frequency amongst its peers. The ratio $\frac{R_j}{R_i}$ defines the current state of *node j* compared with *node i*. These parameters altogether represent the active rate $A_i(j)$ of *node j* to *node i*.

$A_i(j)$ is decided as:

$$A_i(j) = \frac{R_j}{R_i} * N_j \tag{1}$$

The active rate $A_i(j)$ is used for the *commit* operation on node i. R_i is quaranteed to be the latest revision value by the operator at node i, hence $R_i = R_{all}$. The higher the ratio $\frac{R_j}{R_i}$ is, which means the higher R_j is, the more recently node j gets updated. The higher N_j is, the more traffic goes in and out of node j. All in all, the higher $A_i(j)$ is, the more active *node j* is, compared with *node i*.

Generally, this algorithm provides a simple yet practical and effective mechanism to decide the appropriate active nodes for versioning control operations.

4.1.2 Commit

Before an action is taken on *node i*, it is the responsibility of *node i*'s owner to make sure that *node i* is current or in other words, R_i is the overall latest revision. This can be obtained by simply comparing R_i with the special field R_{all} reserved in the *state table* of *node i*.

To commit a change, *node i* first commits the change to its own server, then informs all other nodes to update their special field R_{all}, and then star-synchronizes between its server and its connected nodes with highest active rates. This way, a node will always be able to maintain the updated value of R_{all} as long as it is kept connected in the system. On the other hand, if a disconnection occurs, after the reconnection, a node can retrieve the updated value of R_{all} from any other nodes that it connects to. This mechanism keeps all nodes be aware of the current state of the system.

This sequence of action is illustrated in figure 4.

In ideal cases, no other changes are made concurrently with *node i*, R_i is increased by 1 and becomes the highest revision value (R_{all}). Therefore, $R_j < R_i \; \forall j \rightarrow A_i(j) < 1 \; \forall j$.

If there exists j such that $A_i(j) \geq 1$, it is an indication that a conflict occurs. When a *commit* operation is performed, node i consists of the updated revision of the system, i.e. R_i equals to R_{all} and be the highest revision value. Hence, R_j must be either less than or equal to R_i, which makes $A_i(j) \leq 1$. Given a situation in which node j is up-to-date too, i.e. R_j equals to R_i; node j performs another *commit* operation concurrently with node i, and increases R_j by 1, meanwhile node i commits without checking R_i and R_{all} again. As a result, conflict occurs. In this case, R_j is greater than R_i, which makes $A_i(j) \geq 1$.

The conflict needs to be resolved first using appropriate strategies discussed in section 2, before *commit* can take place again.

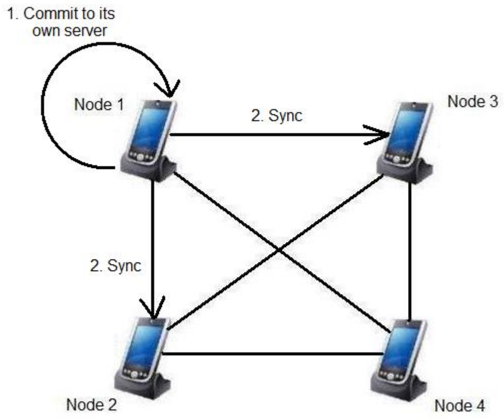


Figure 4: Operation: Commit

4.1.3 Checkout and update

Checkout and *update* are basically similar to each other and simplier than *Commit*.

When a node is added to the system, it performs *checkout* and *update* by repeatly comparing its connected peers' revision values with R_{all} of the system until a match is obtained, to make sure it gets updated with the lastest version of the shared documents. If none of its connected peers has the lastest revision, it can requests its highest revision peer to perform a different *update* request to be brought into the up-to-date state and then notify it to change accordingly.

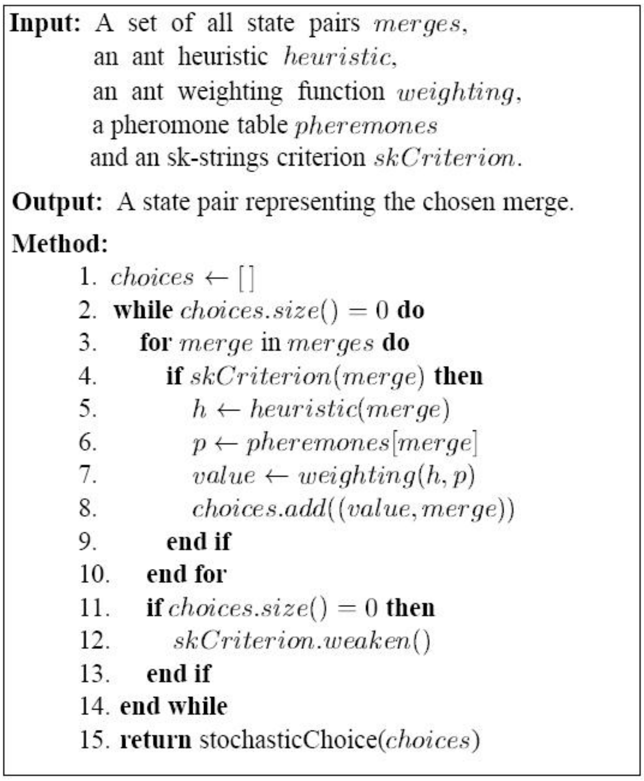
If *node i* checkouts or updates from *node j*, N_i and N_j are increased by 1 and updated in the associated *state tables*.

4.2 Structural inference for XML data

Testing results presented in (Sankey & Wong 2001) revealed that the merge family methods (sk-strings method) performed well in some cases, and the optimisation methods (Ant Colony Optimisation - ACO) better in others. This led to the development of the sk-ANT Heuristic, based on the most successful method of each type.

The sk-ANT Heuristic: The motivation for this algorithm was to create a method that would be successful for a variety of input data sizes by combining the best features of both the sk-strings and ACO techniques. One consideration was to first run the sk-strings algorithms, and then use the results to seed the ACO optimisation. However, this approach suffers from several problems. Firstly, it is not practical to attempt all possible combinations of both algorithms. Thus we would be required to choose a limited number of models resulting from the sk-strings technique to seed the second stage of the process. The simplest way to achieve this would be to choose the best models, up to a reasonable number. These models will not necessarily lead to the best results, though, as they may have already been over-generalised. More importantly, by letting the sk-strings methods run to completion we would lose many of the advantageous aspects of the ACO method. Most notably, its willingness to explore a greater breadth of the search space would be missed.

The sk-ANT method thus incorporates both the sk-strings and ACO heuristics at each step of the inference process. It is most easily described as a modified version of the ACO technique with the ants guided by an sk-strings criterion. The guiding is made progressively weaker as the model becomes smaller, to allow the advantages of the ACO method for smaller models to take effect. The main modification was made to the ant move selection of Algorithm 4, producing a new method as shown in Algorithm 5. The first major difference appears on line 4, where a merge must pass the sk-strings criterion to be considered. The outer while loop on line 2 and if statement on line 11 combine to progressively weaken the sk-strings criterion when it has become too strict. Eventually the criterion will be weak enough to let all merges pass, and the algorithm will behave identically to the original version.



Input: A set of all state pairs *merges*,
an ant heuristic *heuristic*,
an ant weighting function *weighting*,
a pheromone table *pheremones*
and an sk-strings criterion *skCriterion*.

Output: A state pair representing the chosen merge.

Method:

```
1.  $choices \leftarrow [\,]$
2.  while $choices.size() = 0$ do
3.    for $merge$ in $merges$ do
4.      if $skCriterion(merge)$ then
5.        $h \leftarrow heuristic(merge)$
6.        $p \leftarrow pheremones[merge]$
7.        $value \leftarrow weighting(h, p)$
8.        $choices.add((value, merge))$
9.      end if
10.   end for
11.   if $choices.size() = 0$ then
12.     $skCriterion.weaken()$
13.   end if
14. end while
15. return stochasticChoice($choices$)
```

Figure 5: sk-antMoveSelector Algorithm

4.3 XML intelligent agent - Query translation from SQL to XQuery

The agent is bound to one or more heterogeneous data sources and communicates with zero or more other agents. A data source queries the agent in its native query language and receives results in its native query result format. It is the agent that queries other heterogeneous data sources, communicates with other agents, integrates the results received and translates the results to the native result format of the data source. The aim is to hide the data representation used to exchange data inside the agent framework and allow data sources to query other heterogeneous data sources and integrate information using their native query language.

This section discusses SQL to XQuery translation, in the context of the XML Intelligent Agent framework. In fact, the agent translates the SQL query to iQuery (intermediate Query representation) that is in turn translated to XQuery or any other query languages. These are the steps for translating a SQL query to XQuery for further processing:

1. Convert to iQuery
   Agent A receives a SQL query, validates it and then converts it to iQuery. For a SQL query to be successfully validated, it must be:

   - specified in the SQL92 standard syntax.
   - qualify all expressions, if the query involves a join on more than one relation.
   - (optional) use Oracle's schema notation to specify the keywords used in locating a resource.

   If the query involves a join on more than one SQL relation, all expressions in the scope of that query need to be qualified. For example, (SELECT author FROM books) is valid but (SELECT author FROM books, articles) is invalid because the agent cannot figure out which relation contains the author column. The correct syntax is (SELECT books.author FROM books, articles). There needs to be a mechanism for specifying the keywords used in locating a resource. We borrow Oracle's schema identification notation to provide this feature. For example, a query of the form (SELECT author FROM BookSchema.Library) generates the keywords BookSchema and Library.

   To illustrate SQL to XQuery conversion, we use the following example:

   ```
   SELECT book, author
   FROM BookSchema.library
   ```

   The iQuery equivalent of the above SQL query is:

   ```
   <result level=1>
   {
   FOR $library IN resource("bookschema, library")//??
   RETURN
   <result_tuple>
   {$library//book}
   {$library//author}
   </result_tuple>
   }
   </result>
   ```

2. Resolve SQL Expressions
   Next, the iQuery is either processed by Agent A or sent to another agent that can process the query. For the purposes of clarity, we refer to the target agent that processes the iQuery as Agent B.

3. Resolve resource Expression
   Agent B receives the iQuery. First, it must find the XML documents required to process the query. This is a three step process:

   - For each resource keyword in each FOR clause of the iQuery:
   - Extract the resource keywords
   - Use these keywords to find the XML document
   - Replace the resource function with the XQuery document function. The argument of the document function is the name of the XML document.

The mechanism used to locate XML documents using resource keywords is not part of the agent framework. It depends on the heterogeneous resource. One method may be to concatenate all the keywords together to construct a filepath that identifies the document. Another method may be to lookup the keywords in a database that maps keywords to XML documents. Assuming, use of the first method, the result is:

```
<result level=1>
{
FOR $library IN
document(”bookschema/library.xml”)//??
RETURN
<result_tuple>
{$library//book}
{$library//author}
</result_tuple>
}
</result>
```

4. Process level Instruction
   SQL cannot handle composite types. XML documents however, are of a hierarchical nature and can be nested to any level of complexity. This is a problem. If the author element in bookschema/library.xml has child elements first_name, last_name, age; these child elements will be returned when the XQuery is processed. But SQL expects an atomic type in the author field, not a structure composed of first_name, last_name, age.
   iQuery solves this problem. The iQuery above contains an attribute in the result tag named level. A value of 1 requires the agent to confirm that the nodes selected in the RETURN clause of the XQuery have a depth of 1. The following cases are allowed:

   - node has no text node or children but only one attribute (in this case, the value of the attribute is used)
   - node has a non-empty text node and 0 or more child nodes and 0 or more attributes(in this case, the value of the text node is used)
   - node has no attributes, text node or child nodes (in this case, the ”null” string is used)

   If all XPath expressions in the RETURN clause satisfy one of the above conditions, execution proceeds to the next step. If a selected node has no text node but has two or more child nodes or attributes, an error XML document is returned as the result.

5. Identify XML attributes
   The agent learns the elements and attributes in the target XML document. If the XQuery accesses an element which exists as an attribute in the XML document, the query is modified to access the attribute and not the element.

6. Process distinct Instruction
   In addition to the level attribute, the result tag may also contain an attribute called type. This attribute can only have one value - DISTINCT. It the type attribute is set, the XQuery is modified to ensure that the query result does not contain any duplicates.

7. Process XQuery, Return SQL Relation
   Lastly, the XQuery is forwarded to the XQuery engine, processed and the resulting XML document returned to Agent A. Agent A generates a SQL result representation from the XML document and returns it to the querying data source. This process is trivial. First the agent attempts to create a metadata SQL structure from the XML document. Nested child nodes inside an element node are interpreted as the result of a GROUP BY statement. Attributes of an element node are treated as child elements. Then the XML document is converted to a SQL result based on the derived metadata.

5 Experimental results

5.1 SQL to XQuery examples

To illustrate query translation, we make use of a small database with the Entity-Relationship and relational schema as in Figure 6

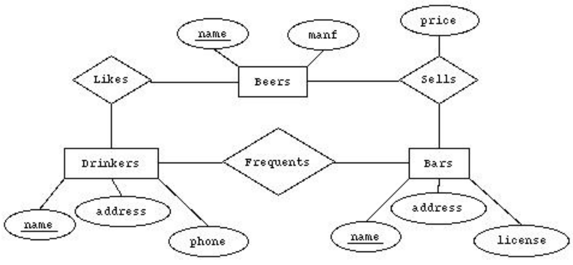


Figure 6: A sample relational database

The XML representation of the database in Figure 6 is a set of XML documents bound to an XQuery engine. Each relation in the database maps to an XML document.

SQL query to find the names and brewers of beers that John likes:

```
SELECT manf, name
FROM Beers
WHERE name IN
(
SELECT beer
FROM Likes
WHERE drinker = ’John’
)
GROUP BY manf, name
```

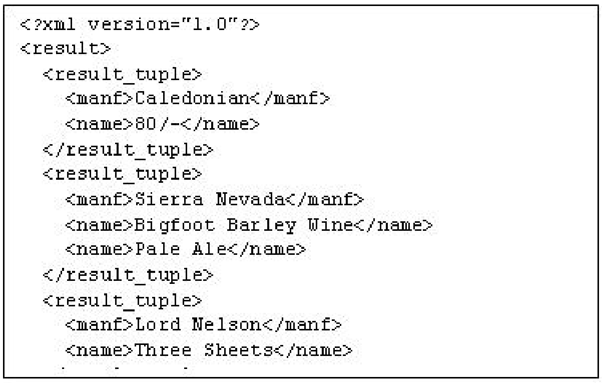
The iQuery equivalent to the above SQL query is:

```
<result>
{
FOR $beers IN
distinct(document(”beers.xml”))//beer_tuple/manf
LET $XQ_FUN1_RESULT := XQ_FUN1()
LET $beers1 := document(”beers.xml”)
//beer_tuple[manf=$beers
AND name=$XQ_FUN1_RESULT]
WHERE not(empty($beers1))
RETURN
<result_tuple>
{$beers}
{$beers1/name}
</result_tuple>
}
</result>
```

in which:

```
DEFINE FUNCTION XQ_FUN1 () RETURNS sequence
{
FOR $likes IN document(”likes.xml”)
//like_tuple[drinker=”John”]
RETURN $likes/beer
}
```

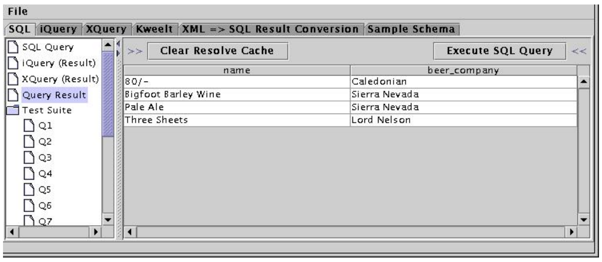
Figure 7 shows the XML Result representation.



```
<?xml version="1.0"?>
<result>
  <result_tuple>
    <manf>Caledonian</manf>
    <name>80/-</name>
  </result_tuple>
  <result_tuple>
    <manf>Sierra Nevada</manf>
    <name>Bigfoot Barley Wine</name>
    <name>Pale Ale</name>
  </result_tuple>
  <result_tuple>
    <manf>Lord Nelson</manf>
    <name>Three Sheets</name>
```

Figure 7: XML result representation

Figure 8 shows the SQL Result representation.



name	beer_company
80/-	Caledonian
Bigfoot Barley Wine	Sierra Nevada
Pale Ale	Sierra Nevada
Three Sheets	Lord Nelson

Figure 8: SQL result representation

Note that the length of the resulting XML document is minimized by grouping the results. But doing this does not affect the final SQL result. Also, note that the datatype sequence is identified in the function but it is not bound to a XML Schema or explicitly defined anywhere. This is impossible as the agent does not know the datatype of the sequence or the result of the function.

5.2 Inference algorithms

The small data set consisted of 100 sample files generated from random Probabilistic Finite State Automaton (PFSA) with a maximum of 5 states and an alphabet cardinality of 4. A total of 10 sample strings were generated for each PFSA, leading to Prefix Tree Automaton (PTA) with an average size of 42.15 states. The number of strings generated was deliberately kept small, as sparse data is an important problem in practical cases of inference. The average size of the models inferred by the algorithms ranged from 1.03 to 40.55 states, with the best models in terms of Minimum Message Length (MML) typically having between 2 and 5 states.

Figure 9 shows the success rates of several algorithms in inferring models with the lowest MML values. For each algorithm, two results have been shown. The first is the frequency of inferring the best model overall, by choosing the best of the algorithms attempts.

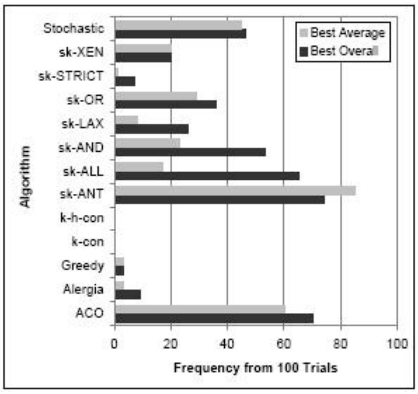
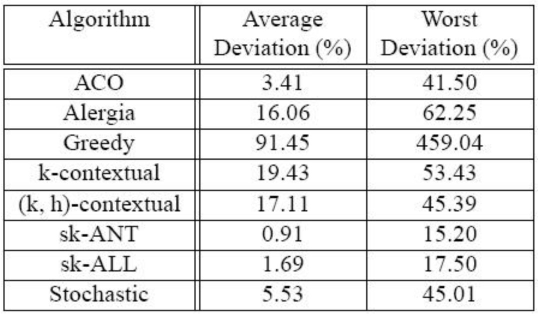


Figure 9: Success rates for small models

The second is the frequency of obtaining the best average performance, derived by averaging all of the algorithms attempts before ranking. The best overall performance is most important, whilst the best average indicates stability across diferring input parameters. The results clearly show that the sk-ANT algorithm performed best in terms of both rankings, particularly the best average ranking. The next two best algorithms were the original ACO method, and the sk-ALL heuristic (a combination of the results from all ak-strings variants.) This is one of the reasons those methods were chosen as the basis for sk-ANT. The other previously applied algorithms and reference methods performed quite poorly. Although the Stochastic method was able to infer some good models, it trailed significantly.



Algorithm	Average Deviation (%)	Worst Deviation (%)
ACO	3.41	41.50
Alergia	16.06	62.25
Greedy	91.45	459.04
k-contextual	19.43	53.43
(k, h)-contextual	17.11	45.39
sk-ANT	0.91	15.20
sk-ALL	1.69	17.50
Stochastic	5.53	45.01

Figure 10: Deviation from the best model inferred (small data set)

Statistics relating to the consistency of the algorithms over the 100 test cases were also gathered, in the form of comparisons against the best inferred models. Figure 10 shows both the average deviation and worst deviation in percent for each of the algorithms. The numbers were derived from the difference between the MML values of the best model inferred by a given algorithm as compared with the best model overall. We omit the individual sk-strings algorithms, preferring the combined sk-ALL results. The results show that the sk-ANT method is the best in terms of average and worst deviation, followed by the sk-ALL hueristic. Note that the worst case deviations may be too high for some applications. In such cases, using a combined approach with both the sk-ANT and sk-ALL algorithms would yield more consistent results.

5.3 Decentralised system

The SVK version control system mentioned previously in section 2 is used as a based system for our initial prototype. We created a C# application on top of the SVK system to handle communication between nodes. Some performance measurements have been carried out to evaluate the execution time of various file size, multiple files and multiple servers. Several nodes are set up to communicate with each other, sending and receiving requests.

Figures 11 and 12 show that the graphs of execution time of various file sizes and multiple files are almost linear. In other words, there is not much difference in terms of overhead between multiple small transactions and an equivalent big transaction. These experimental results provide a good justification for a version control framework, since collaborators can synchronise their work as frequently as possible, bit-by-bit without having to wait for the whole tasks done.

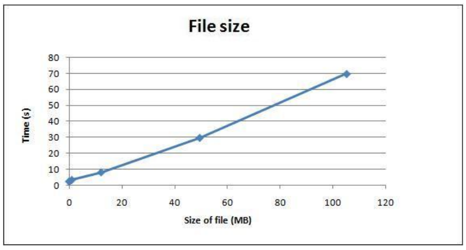


Figure 11: Execution time of various file size

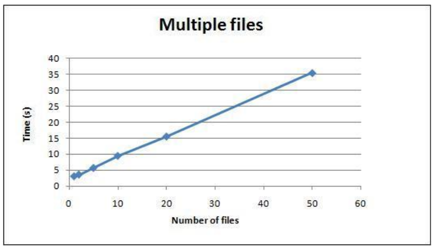


Figure 12: Execution time of multiple files

Figure 13 illustrates the same implication for execution time of multiple servers. The graph of execution time of multiple servers is also almost linear, i.e. the overhead of multiple single-server-synchronisations is not significant in comparision with one multiple-server-synchronisation. Therefore, it is not necessary for one transaction to involve a synchronisation with as many servers as possible to minimize the total transaction time; instead, single server transaction can be done as per requested any time without breaking the minimal total execution time. Ideally, the execution time, as a result, does not effect the active rate attribute of a node.

6 Conclusion

The emergent development of mobile devices and technology has significantly enhanced cooperation in many commercial applications.

Cooperation requires a sufficient version control system in mobile environment without a central server. In this paper, the proposed decentralised version control framework has successfully addressed this requirement and provides a simple yet efficient algorithm to support communication between mobile devices directly.

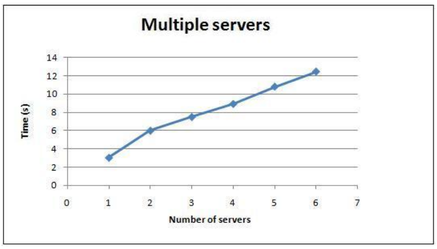


Figure 13: Execution time of one client and multiple servers

The paper has also presented a scalable, lightweight solution to enable information exchange and integration between heterogeneous data sources. The solution also allows querying heterogeneous data sources, especially in XML format, using a relational query language (for example, SQL). Thus most existing mobile data management applications can use the proposed framework for heterogeneous, XML content without expensive changes. In order to facilitate querying XML content with SQL, schema generation for the underlying XML content is needed. Therefore, we have also addressed the problem of structural schema inference for XML and evaluated various algorithms to infer schema for XML data on mobile devices.

Our algorithm for structural schema inference, as well as most of the existing ones, runs in batch mode, that is it has to be rerun whenever there are new documents added. For future work, we will investigate how to extend the existing inference technique to an incremental version, while still maintaining minimum memory and CPU usages.

Acknowledgement

William K. Cheung and Jiming Liu are partially supported by HKBU Faculty Research Grant FRG/03-04/II-70 for this work. Part of this work was done when Raymond K. Wong was visiting HKBU.

References

Adelberg, B. (1998), 'NODOSE—a tool for semi-automatically extracting structured and semistructured data from text documents', *In SIGMOD* **27**(2), pp. 283–294.

Ahonen, H. (1996), 'Generating grammars for structured documents using grammatical inference methods', Technical report, Department of Computer Science, University of Finland.

Ashish, N. & Knoblock, C. A. (1997), 'Wrapper generation for semi-structured internet sources', *In SIGMOD* **26**(4), pp. 8–15.

Bex, G. J., Gelade, W., Neven, F. & Vansummeren, S. (2008), 'Learning deterministic regular expressions for the inference of schemas from xml data', *In WWW '08: Proceeding of the 17th international conference on World Wide Web*, ACM, New York, USA, pp. 825–834.

Carrasco, R. & Oncina, J. (1994), 'Learning stochastic regular grammars by means of a state merging method', *In ICGI '94: Proceedings of the 2nd International Colloquium on Grammatical Inference*, **862**, Springer-Verlag, pp. 139–150.

Chianese, A., d'Acierno, A., Moscato, V. & Picariello, A. (2008), 'Pre-serialization of long running transactions to improve concurrency in mobile environments', *In ICDEW '08: Data Engineering Workshop* pp. 129–136.

Citro, S., McGovern, J. & Ryan, C. (2007), 'Conflict management for real-time collaborative editing in mobile replicated architectures', *In ACSC '07: Proceedings of the thirtieth Australasian conference on Computer science*, Australian Computer Society, Darlinghurst, Australia, pp. 115–124.

CollabNet (2006), 'Tigris.org: Open source software engineering tools'.
URL: *http://www.tigris.org/*

Ellis, C. A. & Gibbs, S. J. (1989), 'Concurrency control in groupware systems', *In SIGMOD* **18**(2), pp. 399–407.

Hammer, J., Garcia-molina, H., Cho, J., Aranha, R. & Crespo, A. (1997), 'Extracting semistructured information from the web', *In Proceedings of the Workshop on Management of Semistructured Data*, pp. 18–25.

Ionescu, B., Binder, J. & Ionescu, D. (1999), 'A distributed architecture for collaborative applications', *Pacific Rim Conference on Communications, Computers and Signal Processing, IEEE* pp. 525–529.

Jiang, B., Zhang, H., Chen, C. & Yang, J. (2005), 'Enable collaborative graphics editing in mobile environment', *In CSCWD* **1**, pp. 313–316 .

Kao, C.-l. (2003), 'The svk version control system'.
URL: *http://svk.elixus.org/*

Krishnamurthy, R., Kaushik, R. & Naughton, J. F. (2004), 'Efficient xml-to-sql query translation: Where to add the intelligence', *In VLDB*, pp. 144–155.

Liu, L., Pu, C. & Han, W. (2000), 'XWRAP: an xml-enabled wrapper construction system for web information sources', *Proceedings of the 16th International Conference on Data Engineering* pp. 611–621.

Munson, J. & Dewan, P. (1996), 'A concurrency control framework for collaborative systems', *In CSCW '96: Proceedings of the 1996 ACM conference on Computer supported cooperative work*, ACM, New York, USA, pp. 278–287.

Robert, O. (2006), 'Dvcs or a new way to use version control systems for freeBSD'.
URL: *citeseer.ist.psu.edu/749273.html*

Sahuguet, A. & Azavant, F. (1999), 'Building lightweight wrappers for legacy web data-sources using w4f', *In VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 738–741.

Sankey, J. & Wong, R. K. (2001), 'Structural inference for semistructured data', *In CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, ACM, New York, USA, pp. 159–166.

Su, Z., Katto, J. & Yasuda, Y. (2007), 'Efficient consistency control for mobile dynamic contents delivery network', *International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications* pp. 171–173.

Suleiman, M., Cart, M. & Ferrie, J. (1998), 'Concurrent operations in a distributed and mobile collaborative environment', *Proceedings of 14th International Conference on Data Engineering*, pp. 36 – 45.

Young-Lai, M. D. (1996), 'Application of a stochastic grammatical inference method to text structure', Master's thesis, Computer Science Department, University of Waterloo.

Yu, C. (2004), 'Constraint-based xml query rewriting for data integration', *In SIGMOD*, pp. 371–382.

Event-based Communication for Location-based Service Collaboration

Annika Hinze[1] **Yann Michel**[2] **Lisa Eschner**[2]

[1] University of Waikato, New Zealand
Email: hinze@cs.waikato.ac.nz

[2] Freie Universitaet Berlin, Germany
Email: {ymichel,eschner}@mi.fu-berlin.de

Abstract

Location-based context-aware services for mobile users need to collaborate in disparate networks. Services come and go as the user moves and no central repository is available. The user's personal information and service usage statistics need to be protected. To support service collaboration we propose a service infrastructure that relies on an event-based service-oriented architecture. We implemented a basic version of the architecture and used it for a tourism information system. An advanced version has been modelled using formal methods to evaluate privacy aspects. This paper reports about both architectures and our experiences of their application to tourism-related services.

1 Introduction

Imagine a museum guide that provides rich information about its exhibits on your mobile phone, or a car navigation system that shows the way to the nearest specialist depending on your health status. Mobile users of such services receive information tailored to their situation and preferences on their own portable devices. Context-aware mobile services use information about a user's location to deliver customized information [3]. Many of these services will only be offered in a restricted territory, e.g., close to a hospital or in a museum. For example, Figure 1 shows the areas of availability of two services A and B. As the user moves, services appear and disappear (at the four points marked) and no common network may be available.

To reach their full potential, different services need to collaborate, such as the tour guides, route planners, and health monitors in our example. For a seamless user experience, services must collaborate automatically and share user-related data as necessary. Because services are not known in advance, their collaboration has to be initiated and sustained in an ad hoc manner. At the same time, services and their communication infrastructure should protect a user's personal information, thus, collaboration must be anonymous. It must also be immediate [12].

Service collaboration raises several issues: for a seamless experience, configuration should not be required and services should be able to deal with collaboration partners that appear and vanish as the user moves around. We address these issues by developing a method for independent services to automatically collaborate in a mobile environment without knowing about each other.

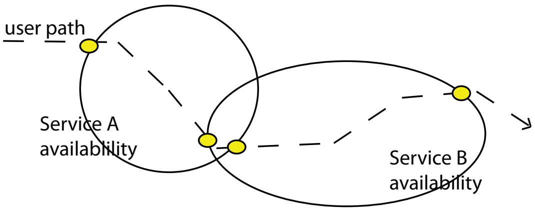


Figure 1: Mobile user and service availability.

We developed this research in the context of the tourism application TIP. The Tourist Information Provider (TIP) is a mobile tourist information system that provides information about sights to travellers based on context, i.e., their personal preferences and their location. The personal preferences are defined in profiles and describe a user's likes and dislikes. Different from other context-aware systems, the TIP system supports several services, e.g., information service [14], recommendations [13] and a map service [16]. In [12] we highlighted the need for redesigning the mobile context-aware TIP system into a service-oriented architecture.

The remainder of this paper is structured as follows: Section 2 discusses related approaches to service collaboration. Section 3 introduces our event-based SOA, Section 4 gives details of service management and collaboration. Section 5 discusses our basic architecture. Section 6 introduces advanced concepts, which are then discussed in Section 7. Section 8 briefly describes the two prototypes; the paper closes with a summary.

2 Related service collaboration approaches

Currently, location-based systems are available only separately. They do not collaborate and the user often has to obtain specialised hardware and software [1][2][3]. No infrastructure exists to support such collaboration. Interesting related issues in location-based systems have been addressed but so far do not consider service collaboration. The solution suggested in [22] provides a subscriber-based means to access location-restricted services in a mobile environment (by using a UDDI channel instead of a repository). Similarly, cloaking has been used to retain privacy

[1] Zingo Taxi, location aware taxi hailing in London, online information at http://www.zingotaxi.co.uk/ , see also online article at http://www.jacobsen.no/anders/blog/archives/2003/03/19/locationaware_mobile_services_zingo_taxis.html

[2] TomTom, personal navigation systems, online information at http://www.tomtom.com/

[3] Geominder, location-based personal reminder, online information at http://www.ludimate.com/products/geominder/

when accessing location-based services (using a centralised third party[10] or a P2P approach [4]).

Previously introduced techniques for service collaboration use a service repository to search for collaboration partners [6]. This traditional Service-Oriented Architecture (SOA) is unfit for collaboration of location-based services in mobile environments because it assumes that services are openly known and that they directly enter bilateral contracts for longer business relations. These assumptions do not hold in a changing mobile environment with disconnected network patches and locally offered services. The original Web Services Architecture from the W3C [1] is even more restrictive and not applicable for mobile environments.

SOA as a more general design principle has been applied to mobile services. However, current solutions do not cater for the complex situation of collaborating location-aware services: architectures for mobile telephones [21] connect static services on mobile devices and require services to be explicitly known. Hodes et al. suggest an architecture for composable and location-based services in an ad hoc network environment [15], in which they use a meta-service index which shares information about the available services in a given network cell. The index service is an equivalent to the SOA service repository and the same limitations apply. Similarly, Le Sommer suggests a service register for remote services using a publish/subscribe mechanism [17]. The ensuing service collaboration is not designed to be anonymous. We observe that current architectures for service collaboration are either not suitable for changing mobile environments or do not support collaboration between anonymous services. Middle-ware solutions for ad hoc environments [23, 18, 20, 11] do not support anonymous service collaboration either.

On the other hand, anonymous and immediate communication is supported in publish/subscribe systems [8]. The aim of these systems is to disseminate published information to a set of subscribers. Each subscriber receives only the information that matches their subscription; communication via the network is uni-directional and anonymous. Publish/subscribe systems use event-based communication that is initiated by the publisher; the event information is filtered and routed through the network. The focus of publish/subscribe developments for mobile and ad hoc environments [9, 5] is on efficient routing strategies in a changing and possibly disconnecting network. However, efficient routing is not required for localised services because the user is in close proximity to the service provider (e.g. in the museum) and thus the service network generally requires only single hops. The publish/subscribe paradigm does not include a service concept nor does it support ongoing collaboration between publishers or subscribers. We explore event-based communication as used in publish/subscribe systems as a means for anonymous collaboration in a service-oriented architecture for mobile location-based services.

3 Basic Architecture

The new TIP 3 architecture introduces an event-based middle-ware, with which every local service interacts. The component representing this middle-ware layer is called the broker. For client/server interaction, co-operating services exchange their information via their local brokers, i.e., every communication from a client to a server service or vice versa is only handled by the brokers and is transparent to the services. On the server-side, the TIP server sends and receives all information via the broker. The server-sided services, e.g., the information service and recommendation service, can also use a database as their information back-end. The services residing on the client-side may interact directly with the user, e.g., a user clicks on the map of the map service to submit a position to the server. In addition, the client system can submit information autonomously, e.g., the location service automatically submits a user's changed position by using the data of a locally attached GPS receiver. For technical reasons, both sides use a wrapper process that is used to effectively start and stop all running processes, i.e., the broker and services. For the server-side, this results in a TIP server process. On the client-side, this process is called the TIP client.

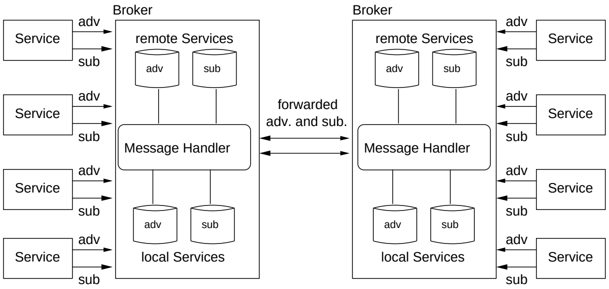


Figure 2: Local and forwarded service advertisement- (adv) and subscription- (sub) handling by the message handlers TIP 3.

The original TIP 1 [14] and TIP 2 [12] systems are based on client/server interaction. The TIP 3 system is planned to also support peer-to-peer and client-to-multi-server connections. In the peer-to-peer approach, servers can be seen as special peers with more available data and unlimited bandwidth or power.

As each service directly communicates with a broker only, the broker is responsible for routing the incoming information to the appropriate services or other brokers. When a service is started, it connects to a broker, advertises the provided information and subscribes for other information. The broker maintains this information in a registry. When a broker connects to another broker, it exchanges its registries' information with the remote broker. This principle is illustrated in Figure 2.

3.1 Event System

A generic and platform independent format for information exchange is required. Therefore, we use the XML-based *simple object access protocol* (SOAP). Slominski et al. [19] propose an event system architecture for their grid project. They use an event-based communication for submitting status information and jobs between the participating grid nodes. Some of their requirements also apply to the TIP system:

- Language and platform independence: The components used for the TIP system should neither depend on a special programming language nor on a special platform.
- Extensibility: New services should be easily added. In addition, existing services should be easily extended.
- Lightweight Publishers: Standard libraries should provide basic functionality. This is a very important fact especially on small devices such as the supposed mobile clients of the TIP system.

Figure 3 shows the basic SOAP message format for the TIP system. Any SOAP message consists of a

```
<?xml version='1.0' ?>
<e:Envelope xmlns:e="http://www.w3.org/2003/05/soap-envelope">
 <e:Header>
  <m:event xmlns:m="http://isdb.cs.waikato.ac.nz/tip/event"
          env:mustUnderstand="true">
   <m:id>{id}</m:id>
   <m:type cacheable="{boolean}" forwarded="{boolean}"
          prefetched="{boolean}">
   {type}
   </m:type>
   <m:dateAndTime>{dateAndTime}</m:dateAndTime>
  </m:event>
 </e:Header>
 <e:Body>
       [...]
</e:Body>
</e:Envelope>
```

Figure 3: Basic TIP SOAP message format for the TIP system.

root element which is the `Envelope`. The `Envelope` contains two elements: a `Header` and a `Body` element. The `Header` is comparable to a common letter head, i.e., it describes general parameters of the submitted information. For our purposes, we use the following parameters:

- `id`: a unique identifier (event number),
- `type`: the type of this event (descriptor for the contained body information),
- `dateAndTime`: the date and time this event was created.

The `type` element is also used for routing the incoming messages properly to the subscribed services without being forced to parse the whole message but just the `header` element, i.e., the `body` element is skipped for this purpose. In addition, several attributes are available for internal handling of the message:

- `cacheable`: This attribute indicates if a message is cacheable (*true*) or not (*false*).
- `forwarded`: This attribute indicates if a message was forwarded (*true*), i.e., it was received from a broker, or locally (*false*), i.e., by a service.
- `prefetched`: This attribute indicates if the message was triggered by a pre-fetching-service (*true*) or if it is an ordinary message (*false*).

The first flag (`cacheable`) is used for deciding if this message can be stored inside a cache or not. The other two attributes, `forwarded` and `pre-fetched`, affect the message routing. If a message was forwarded from a foreign broker, any reply should be sent to this broker only and not to any other services or brokers. If a message has the `pre-fetched` flag set to *true*, a reply should be sent but the reply is then not forwarded to the client's display. The reply is used only for storing the information locally prior to any user request. The `Body` element is the main container for the type specific information that is transported in any message. Therefore its structure is not defined in general but can be freely designed, depending on the type's information.

3.2 Caching

Figure 4 shows the broker of a TIP 3 client. The cache resides within the broker or can be accessed by the broker only. The use of a cache is therefore fully transparent to any service. Inside the broker, the *message handler* is responsible for efficient message handling. The aim of the message handler is to reduce the external network traffic, i.e., the traffic to remote brokers. The local traffic, i.e., the traffic of services with the broker is not considered here. Messages are handled differently depending on their direction.

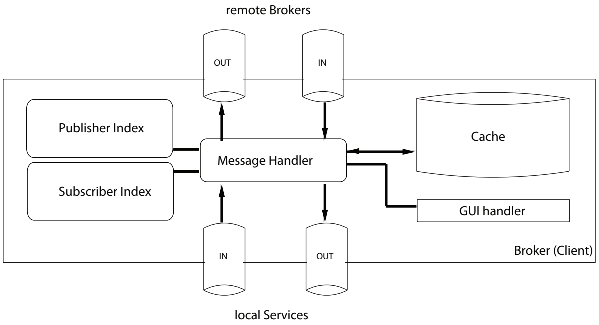


Figure 4: Internal message handling of a client-sided event-broker (basic architecture).

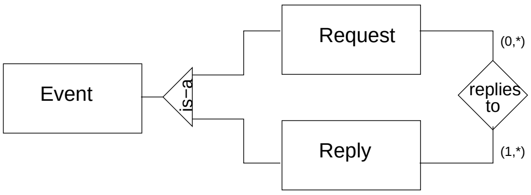


Figure 5: The entity relationship model (ERM) of the cached objects, i.e., events.

The algorithm for handling the outgoing messages to remote brokers is shown in Algorithm 1. When a message handler decides to send a message to a remote broker, it first checks if a similar message was sent before. If that is the case, it sends the still cached replies back to the local services. If no similar message could be found in the cache or cached replies are missing, the message is forwarded.

Any message that is retrieved from a remote broker is handled according to the algorithm shown in Algorithm 2. If the incoming message was a reply to a previous request (message) and is cacheable, it is stored in the cache and then forwarded to the local services.

The relationship of the cached messages, i.e., the cached events, is shown in Figure 5. An event can be either a request or a reply. A request can have zero or many replies and a reply has at least one parent it relates to but can also have more. This allows for effective re-use of cached replies that refer to more than one request. Assuming that a reference to the request object uses less memory then a reply object, this results in less space required for cache memory.

Algorithm 1 Message handling of outgoing messages.

```
if message is cached then
    if replies available then
        return replies;
    end if
else
    send message;
end if
```

Algorithm 2 Message handling of incoming messages.

```
if message is a reply then
    if cacheable then
        store message in cache;
    end if
else
    forward message;
end if
```

4 Basic Architecture – Services

In [12] we suggested the use of a *service-oriented architecture* (SOA) for TIP 3. All components, i.e., the brokers and services, are loosely coupled only. That means that every component acts autonomously and communicates with any other by using a standard protocol, preferably SOAP. As we have already seen, services interact with brokers only. The brokers forward information to other brokers or services that subscribed for a certain event type. This section describes how a service is designed, how the registration process works and how existing services were migrated.

4.1 Service Architecture

A service is a modular component that is loosely coupled to a broker. It runs as a separate process and communicates with a broker via a network connection. After a service is started, it listens on a specified port for incoming events. All outgoing events are sent to a broker whose hostname and port number have to be known by the service. A service can register and un-register itself from a broker. That means it advertises events and subscribes for events that it is interested in. Which events a certain service subscribes to or which it is going to provide depends on the type of the service, i.e, the service's purposes. Figure 6 shows this principle: A service receives an event that it has previously subscribed to. This event is then processed and the service then provides an advertised event to the broker. Services can also work differently, i.e., a service may only provide events but consume none. An example for such a service is a location service that submits a user's location at regular intervals. Therefore, it needs no events to subscribe to but only produces them. A service can also be a sink, i.e., the service only consumes events but does not produce any. An example of such a service may be a history service that only tracks the user's locations to log them into a database. Another type of service can be a split service, e.g., one part of the service is running on the client and the other part on the server-side. An example of such a service is the map service. The map is shown on the client-side but must be retrieved from the server prior to that. A running TIP system has a hybrid mixture of all possible combinations of service types introduced.

All services have the same external interfaces and therefore look identical from the exterior, i.e., broker's view. Services are also capable of receiving any event but will only process some of them, i.e., the ones they subscribed to. To reduce unnecessary communication and workload, the broker should take care that only events that it subscribed to are forwarded to each service.

4.2 Service Management

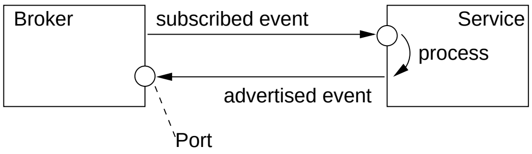


Figure 6: Event exchange between the broker and a service processing them.

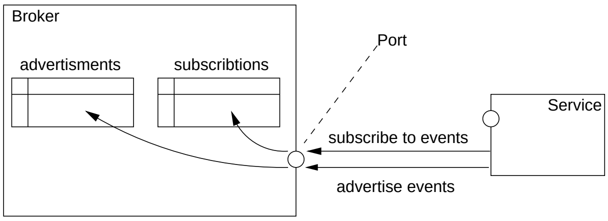


Figure 7: Event subscription and advertisement of a service to a broker.

All information is sent through event messages. The subscription and advertisement of events is wrapped into events as well. We distinguish between system events, i.e., all events that have an administrative purpose, and service events, i.e., events that are processed by the services depending on their purpose. As this section is describing the service management, we are referring to system events throughout the rest of this section. The service events are described in the next section.

Figure 7 shows a service that subscribes to and advertises events. Referring to Figure 3, the corresponding event would have the `type` element set to *Advertisement*. The `Body` would contain the description of the advertised event. If our service was, for example, a location service, it would provide *Location* events. The corresponding Body element provides the name of the event enclosed by the <m:name/>-tags. The subscription events would look similar. The `type` element would simply be set to *Subscription*.

When the broker receives advertisements and subscriptions, the information about the event type and the service that generated these events is stored in an internal registry. This registry is later used for efficient event routing as proposed above, i.e., events are directly forwarded to services that requested them, and not broadcasted.

To draw a bigger picture of the service registration process, we illustrate the sequence by using the UML sequence diagram shown in Figure 8. Assuming that the TIP server is already running and has registered its services, we start the description on the mobile user's site. A TIP user starts its TIP client residing on their mobile device. At first the client software starts the local broker as a core component that is run for every TIP client. Then the locally avail-

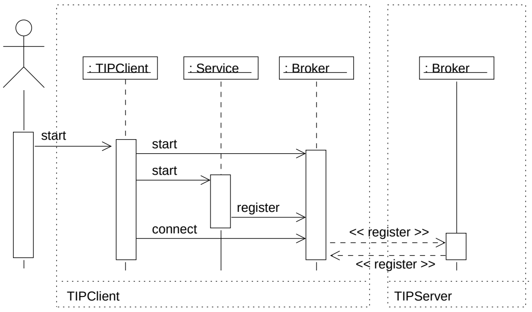


Figure 8: The sequence of the service and broker registration-procedure of the TIP 3 system.

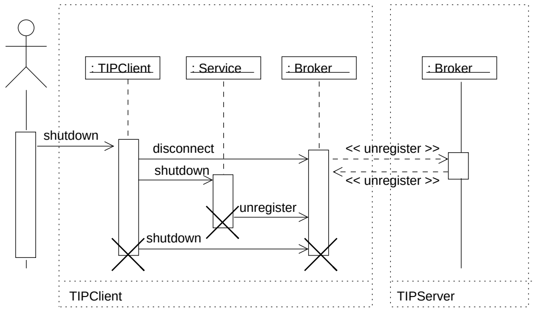


Figure 9: The sequence of the service and broker un-registration–procedure of the TIP 3 system.

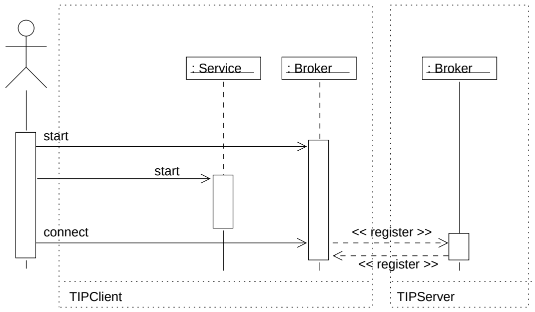


Figure 10: The idealized re-start sequence of the TIP 3 client.

able services are started and told to register with the previously started local broker. Once every service is started, the broker is forced to register itself to a remote server-broker. By registering to it, all local subscriptions and advertisements are forwarded. In addition, the remote server-broker forwards the subscriptions and advertisements of its services. Once this initialization procedure is finished, the user can start working with the TIP system. If the user wants to shut down the TIP client, an un-registration procedure is started. This procedure is shown in Figure 9 and symbolizes the reverse registration procedure: First the client-broker un-registers from the remote server-broker, i.e., the locally advertised events are revoked and the local subscriptions are cancelled. The remote server-broker does this as well for its local services. Then every local client-service is shut down by un-registering from the local broker first and then terminating. When all local services are shut down, the broker itself is shut down and terminated. Then the TIP client terminates.

An idealized procedure for a client startup is shown in Figure 10. Here the user has previously started all services plus the local broker but decided to shut down the client software. Now the client should be re-started. Again, the broker is started first. After that, all services are started, but this time, the local registration procedure is skipped because the service setup has not changed, i.e., there are neither new services nor are pre-known services missing. Therefore the known registry information is re-used. Only the broker is re-registering its local services to the remote site. Similarly, the shutdown procedure is only un-registering the broker from the remote site. Then, the local processes are simply shut down, i.e., the broker and every service is stopped. This means that the registry's entries are kept until the client services are restarted.

In addition, Figure 10 shows a sequence where only the services and the brokers are involved but not the TIP client process that was used above. This is based on the assumption that in this idealized scenario, all services are started independently, whereas in the previous example all services were started together with the broker by using a wrapper process.

4.3 Integration of Legacy Services

In TIP 2, a service is a monolithic software component that combines the data, the logic and the presentation of the provided data. For TIP 3, this single component has to become uncoupled. We use the model-view-controller (MVC) design pattern. The *model*, i.e., the state of the service, might be, for example, represented by the database back-end. Therefore it resides on the server-side in the TIP 3 system. The *view* visualizes the data available to the user. It is the representation layer and therefore part of the TIP client. The *controller*, i.e., the linkage between the two other parts, is represented by the program logic and the event layer. As both the client and the server use this event-layer for their information exchange, this component is found on the TIP server as well as the TIP client. We illustrate how selected services are migrated:

Location Service: This service is a *provide-only* service, i.e., it provides *Location* events but does not consume any other events. Therefore, this service resides on the client-side only.

Information Service: This service cannot be directly converted into a TIP 3 service. The service is split into several smaller ones. A first service uses the location-information, the user's profile and history and the available events to suggest the touristic items. Instead of sending all item information to the client, it would send the item type and identifier to the client. The client then requests the items if the complete item information is not yet locally available. This would ease the reuse of fine grained information, i.e., if an item was part of two consecutive *Information* events, it could be re-used without being requested again. In addition, this division keeps the messages small. In addition, a new service has to be run for every item type that is suggested within its provided events. For the client-side, a *Display* service displays the provided information to the user and requests the item information prior to displaying it.

Recommendation Service: Similar to the Information Service, this service has to be split into several smaller services. The service also subscribes to location information. The provided *Recommendation* event again is only a description of the item types and their recommended identifiers. The item information can be obtained by the same services that have to be introduced for the Information service and provide the item information only. This has to be handled by the introduced *Display*-service.

Map Service: This service consists of a client and a server part in the TIP 3 system. The server part provides maps and subscribes for *MapRequest* events. Based on these events, it provides maps, i.e., *Map* events. The client part displays the map on the mobile device. In addition, it displays the items obtained by the Information and Recommendation Service on these maps. When a user leaves the scope of the locally available map, a new one for the current location is requested. Therefore the client-sided map service submits a *MapRequest* event.

5 Discussion of the Basic Architecture

The basic architecture and the event-based infrastructure well support ad-hoc service collaboration. One example is the typical scenario of the GPS (location service) publishing the user's location to the broker, the information service subscribes to the location events to deliver touristic information. The map service also subscribes to location events to show the user's location on the map. It also offers locations as the user may select points on the map that are then reported as new location events.

This basic scenario can be extended to deal with a more complex situation: The user enters a museum where location information becomes available by reading RFID tags. At the same time, the GPS service is only available with reduced quality. A new information service is available within the museum and a new map (for the museum) needs to be added to the existing map service.

This situation introduces a number of challenges: alternative services with differing quality, parallel services and exclusive services; and the general question of quality of service. We will focus on two questions in particular: (1) How to ensure quality of the map service when the user location data is no longer available? (2) How does the system decide which location service to use?

For the first question: in a traditional request/response interaction the map would be aware of missing location data. This is not the case in event-based interaction. The assumption in publish/subscribe systems is that if no event data is available, no event happened. This assumption does not hold in a mobile context.

For the second question: The concept of parallel services and exclusive services has not been used in this architecture so far. In general, the concept of service types has not been described.

Both questions were approached using formal modelling, as the mere implementation of more services and coding of a particular solution was not deemed to be sufficient for a general collaboration infrastructure that aims to be open to third party services.

6 Advanced Concepts

This section introduces advanced concepts for our event-based SOA that have been verified and evaluated using formal modelling. TIP 3 services are now classified into *service categories*. Service categories are a new concept to TIP. A service category describes the functionality a service provides. A service category groups services, so that services with similar functionality belong to the same service category. Different information and recommendation services belong to the informative service category whereas map services belong to the map service category. A good example is the map service category. Two map services A and B both offer a user interface where the user sees their current location on a map. Map service A offers basic features: the map displays sights; the user can zoom in and out from the map, i.e., change the map's scale. Map service B offers the same features as map service A and some additional features: the user can also select a new location, e.g., by dragging the map; the user can select a start and an end point for a route planner that map service B co-operates with. Both map services belong to the same service category.

Similarly, TIP events are classified into *event types* as described above. An information service subscribes to location events. Whenever it has processed a location event it publishes an information event. Event types are classified into *event categories*. When the information service publishes a location event, this location event and the information events belong to the event category information events, enabling subscribers to differentiate between location events from different publishers and to treat them accordingly. A location event belonging to the information event category is treated differently by the map service from a location event belonging to the location event category, for example.

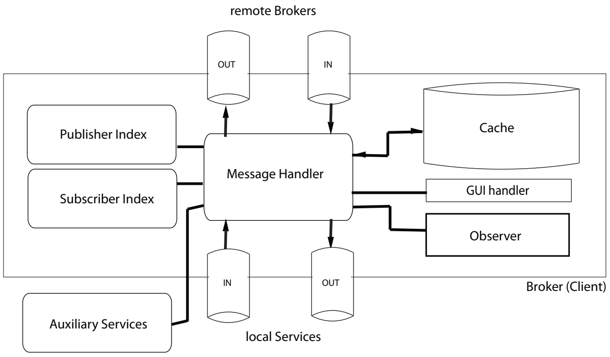


Figure 11: Internal message handling of an extended event-broker.

Figure 11 shows the advanced TIP 3 client architecture. We start with a short discussion of a TIP 3 peer. We then discuss TIP 3 services and their attributes before we discuss the observer and the broker.

A TIP 3 service offers functionalities, e.g., sights on locations and information about the sights. A service may publish events and subscribe to events. Event publishers provide an advertisement. Event subscribers provide a set of functional conditions and subscription rules. Every service provides a service description.

A service does not locate co-operation partners. It simply is subscribed to the events needed to provide its functionality. Services are grouped into service categories. For example, services providing information on sights belong to the informative service category while services offering map tiles and services displaying the map tiles belong to the map service category.

A TIP service provides a service description, an advertisement, functional conditions and subscription rules.

The service description provides information about the service. It specifies what service category the service belongs to. It defines the event categories used by the service, as well as the event types it publishes. Furthermore it informs about the quality of the published data, the service's maximum latency and the service's maximum failure rate. The service description also informs about the owner of the service and provides other administrative information.

The service advertisement tells about the data published by the service. It defines the event and service category, the event type and the quality of data. A service may provide several advertisements, one for each event type that it publishes.

The service conditions specify a subscriber's functional pre-conditions. The service defines the event it requires to be able to supply its functionality, e.g., location events. It may further specify what kind of service should publish the data, i.e., the publisher's service category. The subscriber can also decide if the publisher should be a local service, i.e., a service that is located on the same device, or an external service. The information service, for example, would require location data from a location service. If the service's pre-conditions are not satisfied, the service cannot provide its functionality. A service condition is a tuple <data category, service category, local publisher, remote publisher>. The subscription rules define these conditions in more detail – the required data format, quality of service and, if necessary, the publishing service, amongst others.

Rules enable the service to prioritise certain event types over others or to choose between several publishers. In a group, the rules are prioritised: a priority can be set on high, medium or low. Rules with high priorities are evaluated before those with lower priorities. If the rule with a higher priority has been evaluated successfully, i.e., the evaluation resulted in a subscription, rules that have lower priorities and belong to the same group are not evaluated. This gives the service the opportunity to choose between different event types from the same event category, or between different data qualities. Subscription rules are a tuple <priority, event type, event category, quality, exclusive subscription, service category of publisher, publisher, local, external, allowable latency, maximum failure rate>.

The event category is needed to select the subscribed event type if the same event type is offered by several services: location data may, for example, be published by the location service, and by the information service. A service that is only interested in the user's current location will subscribe to location data from the location category. Another service interested in location data from sights, would subscribe to location data from the information category.

When a service subscribes to events from only one publisher we call this an exclusive subscription. For example, the map service should only subscribe to location data from one location service, and not from several location services at the same time. The map service would then have to name its favourite publisher.

The service category of the publishing service can be named as well. The map service subscribes to location data both from the location service and the information service. The information service subscription is not exclusive, however. The map service then defines in a rule that it wants to subscribe to events that are published by services belonging to the category of information services.

Services can also specify if they want to subscribe to data generated locally, or if the data should be computed remote, e.g., on a client. This is needed for location subscriptions, amongst others. The allowable latency and the maximum failure rate specify features of the publishers, and prevent that a service subscribing to publishers that provide poor quality.

The observer evaluates the service conditions and rules. It monitors the connection to services or brokers, i.e., it monitors if services or brokers have been disconnected. In case of disconnection the observer removes the advertisements and subscriptions from the disconnected service or broker.

Although the observer may behave like an independent actor, it is located at the broker. The observer is called during the service startup routine. A service delivers its advertisements and subscription rules to the observer. The observer then selects some event types from the available types at the broker, i.e., from the event types other services have advertised, using those rules. When a newly registered publisher advertises its data, the subscriptions may be changed if needed or possible. When a publisher disconnects, the subscriptions are re-evaluated as well.

When a service wants to subscribe to data not available in the requested data format, the observer requests that the broker starts an auxiliary service that can convert the available data format into the requested.

The broker or event-manager provides a communication interface for local services and for other brokers. It connects local services with one another and connects to external brokers. The broker receives the events from publishers, filters them and forwards them to the respective subscribers. The broker starts auxiliary services if needed. The broker keeps track of which service publishes what data type, and which service subscribes to what data type. The auxiliary services convert from one data format to another. They are managed by the broker, i.e., if an auxiliary service is requested, the broker starts it.

The publisher and subscriber index are used by the broker to keep track of what service publishes which data, and who subscribes to which data. The subscriber index is accessed during the filtering process. The publisher index is accessed during the evaluation of rules and conditions. The TIP databases are typically located at the tip 3 server peer. They store geo-spatial data, information on sights and user data. Other services access the databases through a database service.

We refrain from showing all model parts here as the necessary level of detail for an in-depth discussion cannot be obtained. Figure 12 shows one example of the model for the (server) broker filtering incoming events. The broker is responsible for service registration, service deregistration, publishing events to the broker, and filtering events. Here we briefly sketch each step and give some details of the model for the filtering.

1. **Service registration** When a service registers with the broker, it first publishes its service description to the broker. A subscribing service provides a set of functional conditions and subscription rules. The observer evaluates the conditions. If they are satisfied, the observer evaluates the services subscription rules and subscribed the service to the events needed. Otherwise, the registration process is stalled until the conditions are satisfied, i.e., until the events needed have been advertised to the broker. Publishing services announce their advertisement.

2. **Service deregistration** When a service deregisters, its advertisement is removed. The subscriptions are removed as well. The broker does not try to filter messages to a subscriber that does not exist any more. After a time interval, services are deregistered from the broker by the observer if the service for some reason has been disconnected. This avoids deregistering of services and re-evaluation of subscription rules when a service disconnects for a short time and then re-connects.

3. **Publishing events to the broker** Services publish events to the broker. We first verify that services may publish events to the broker. If this is the case then the broker enqueues the message in its in-queue.

4. Filtering events When the broker receives events from local services or from other brokers, it filters the events and forwards them to the respective subscribers. In the first step, a service sends its message to the broker. The broker receives the event and enqueues it in its in-queue. Then the broker dequeues the message (Step 2) to first check if there are any subscribers (Step 3) or if the message was published by the database service. If the message was published by the database service, the event is forwarded directly to its recipient and the broker returns to the idle location. Otherwise the broker identifies the subscribers in Step 4. In our example, another service has subscribed to the event. In Step 5,

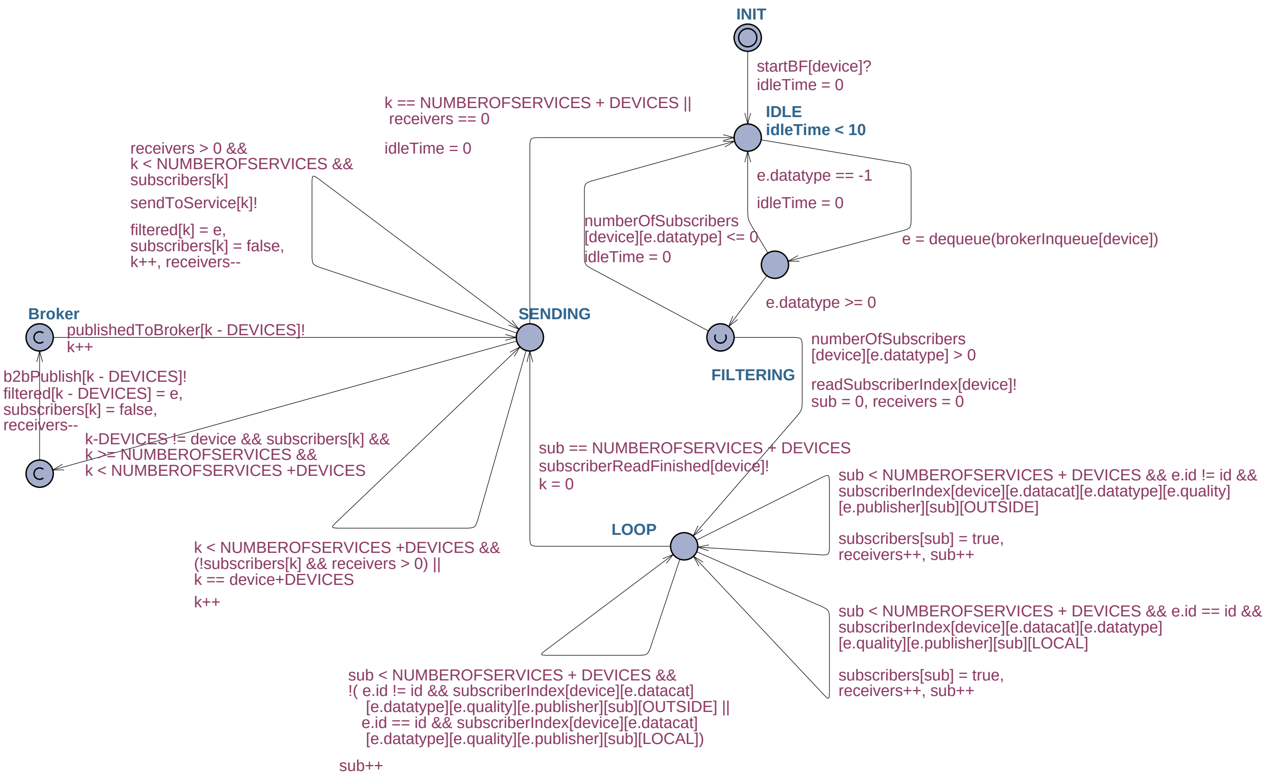

INIT
startBF[device]?
idleTime = 0
IDLE
idleTime < 10
k == NUMBEROFSERVICES + DEVICES ||
receivers == 0
idleTime = 0
receivers > 0 &&
k < NUMBEROFSERVICES &&
subscribers[k]
sendToService[k]!
filtered[k] = e,
subscribers[k] = false,
k++, receivers--
e.datatype == -1
idleTime = 0
numberOfSubscribers
[device][e.datatype] <= 0
idleTime = 0
e = dequeue(brokerInqueue[device])
Broker
publishedToBroker[k - DEVICES]!
k++
SENDING
e.datatype >= 0
FILTERING
numberOfSubscribers
[device][e.datatype] > 0
readSubscriberIndex[device]!
sub = 0, receivers = 0
b2bPublish[k - DEVICES]!
filtered[k - DEVICES] = e,
subscribers[k] = false,
receivers--
k-DEVICES != device && subscribers[k] &&
k >= NUMBEROFSERVICES &&
k < NUMBEROFSERVICES +DEVICES
sub == NUMBEROFSERVICES + DEVICES
subscriberReadFinished[device]!
k = 0
sub < NUMBEROFSERVICES + DEVICES && e.id != id &&
subscriberIndex[device][e.datacat][e.datatype][e.quality]
[e.publisher][sub][OUTSIDE]
subscribers[sub] = true,
receivers++, sub++
LOOP
k < NUMBEROFSERVICES +DEVICES &&
(!subscribers[k] && receivers > 0) ||
k == device+DEVICES
k++
sub < NUMBEROFSERVICES + DEVICES && e.id == id &&
subscriberIndex[device][e.datacat][e.datatype]
[e.quality][e.publisher][sub][LOCAL]
subscribers[sub] = true,
receivers++, sub++
sub < NUMBEROFSERVICES + DEVICES &&
!(e.id != id && subscriberIndex[device][e.datacat]
[e.datatype][e.quality][e.publisher][sub][OUTSIDE] ||
e.id == id && subscriberIndex[device][e.datacat]
[e.datatype][e.quality][e.publisher][sub][LOCAL])
sub++

Figure 12: Model for Broker message filtering.

the broker synchronises with the subscribing services through `sendToService[k]!`. The index k identifies the subscribing service and selects a communication channel. The service receives the message. For a detailed discussion of the modelling we refer to our Technical Report [7].

7 Discussion of the Advanced Architecture

The introduction of quality of service concepts, observers, rules and auxiliary services remedies the shortcomings identified in the discussion of the basic architecture (Section 5). Our interaction famework now covers alternative services, service selection, quality of services and exception handling. Revisiting our previous scenario of a user entering a museum, the following interactions will be triggered: (1) The observer is initialised with quality-of-service rules about location services, (2) The observer detects the low quality of the GPS service when entering the museum, (3) An auxiliary service translates RFID information into GPS data, (4) Preference is given to the higher quality data.

The next challenge facing our event-based SOA is that of sufficient privacy and protection of user-related information. How much information needs to be exchanged between services and how much information can be hidden from the service vendors? These questions will be addressed in future research as more complex modelling and validation are necessary that are, unfortunately, beyond the capabilities of most existing model checking tools.

8 Prototypes

Two prototypes were implemented, one uses Java programming and the other formal modelling in Uppaal.

Basic architecture The basic TIP 3 architecture was implemented using Java. The implemented network is based on a TCP/IP stack. Every component that submits or receives information over the network uses a Connector. To simplify the network access for services and brokers, this connector hides the send and receive implementations. The transported information is an event. This event is wrapped into a message that additionally contains the receiver details, i.e. hostname and host port. A connector provides send and receive methods. The sender and receiver objects are hidden inside and work autonomously. That means, whenever a new message is enqueued for sending, the sender automatically forks a worker process for submitting the contained event to the addressed remote party. Whenever the component, i.e., the service or broker, that owns the connector calls the receive method, a received message is taken from this queue. If the queue is empty the receive method blocks and waits until a new message is enqueued.

This prototype was used to implement and verify the concept of an event-based SOA. It was also used to implement and test caching and pre-fetching in TIP 3.

Advanced architecture We re-implemented the basic architecture using real-time discrete event modelling in Uppaal [2]. Uppaal is a tool-box for the verification of real-time systems. Uppaal uses timed automata to model processes. We extended this basic architecture with the advanced concepts described in the previous section: observers, rules, and auxiliary services. The model was divided into three parts for reasons of clarity and verification. We decided to model the client peer and its services in one model. The server peer and its services are modelled in a second model. The communication between several peers is modelled in a third model. The design and modelling process comprises three interleaving steps. In the first step, the usability requirements should be recognised, as they are needed later during the design and modelling process. During the modelling process, the properties are identified and formulated as verification queries in the second step. The requirements are used to formulate the properties. In the third step, the model is verified using the properties. The verification of the properties often leads to a revision of the model. The different steps often interleave: the analysis of the verification results can lead to a re-engineering of the model or the property, so that the model has to be verified again or the new property has to be verified. Finally the results of the last verification are analysed. The result analysis differentiates two cases: either the property holds under the assumptions upon which the model is based, or it does not hold. If the property does not hold, it is examined and analysed. A property that does not hold highlights weaknesses in the model and should result in a better model or property.

The prototype was used to verify basic properties of advanced architecture and infrastructure. The verification was largely performed using simulation. Our simulations showed that the model functions adequately and without deadlock: services are able to register, i.e., publishers can advertise, subscribers are subscribed to events if their functional conditions are satisfied; publishers can publish events to the broker; the broker filters events to the subscribers; services can deregister, or are deregistered by the observer in case of disconnection. The observer evaluates the services' functional conditions and subscription rules. When a new publisher has advertised to the broker, every subscribers' subscription rules are evaluated and subscriptions accordingly updated. If a registering service's functional conditions are not satisfied, the registration process is stalled until an appropriate advertisement has been published to the broker. Our simulations have shown that only the currently visible service reacts to user input.

9 Summary

In this paper, we proposed an event-based service-oriented architecture for collaboration of mobile context-aware services. All services are loosely coupled and interact via an event-based middle-ware. The information exchange is event-based, i.e., events trigger the submission of information. We illustrated the principle of events message exchange in two versions: a basic architecture that was implemented and practically evaluated, and a formal modelling of an extended architecture to allow for verification and generalizability of observations. We outlined how existing legacy services of a tourist information system can be migrated to the new TIP 3 architecture.

References

[1] Web services architecture. Technical report, World Wide Web Consortium, February 2004.

[2] G. Behrmann, A. David, and K. G. Larsen. A tutorial on uppaal. In *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, page 200236, 2004.

[3] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical report, Hanover, NH, USA, 2000.

[4] C.-Y. Chow, M. F. Mokbel, and X. Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In *GIS '06: Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, pages 171–178, New York, NY, USA, 2006. ACM.

[5] G. Cugola and H.-A. Jacobsen. Using publish/subscribe middleware for mobile systems. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):25–33, 2002.

[6] T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[7] L. Eschner and A. Hinze. Design and formal model of an event-driven and service-oriented architecture for the mobile tourist information system tip. Technical report, University of Waikato, November 2008.

[8] P. T. Eugster, P. A. Felber, and A. marie Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35:114–131, 2003.

[9] D. Frey and G.-C. Roman. Context-aware publish subscribe in mobile ad hoc networks. In A. L. Murphy and J. Vitek, editors, *COORDINATION*, volume 4467 of *Lecture Notes in Computer Science*, pages 37–55. Springer, 2007.

[10] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42, New York, NY, USA, 2003. ACM.

[11] K. Henricksen, J. Indulska, and T. Mcfadden. Middleware for distributed context-aware systems. In *International Symposium on Distributed Objects and Applications (DOA*, pages 846–863. Springer, 2005.

[12] A. Hinze and G. Buchanan. The Challenge of Creating Cooperating Mobile Services: Experiences and Lessons Learned. In *Twenty-Ninth Australasian Computer Science Conference (ACSC 2006)*, Hobart, Australia, Jan. 2006.

[13] A. Hinze and S. Junmanee. Advanced recommendation models for mobile tourist information. In *Federated Int. Conference On The Move to Meaningful Internet: CoopIS*, pages 643–660, 2006.

[14] A. Hinze and A. Voisard. Location- and time-based information delivery in tourism. In *Conference in Advances in Spatial and Temporal Databases (SSTD 2003)*, volume 2750 of *LNCS*, Santorini Island, Greece, July 2003.

[15] T. D. Hodes and Y. H. Katz. Composable ad hoc location-based services for heterogeneous mobile clients. *ACM Wireless Networks*, 5:411–427, 1999.

[16] X. Huang. Travel Planning Map Service – Development of a Java-based Application for Travel Planning. Master's thesis, Dep. of Comp. Sc., University of Waikato, Sept. 2006.

[17] N. Le Sommer. Service Provision in Disconnected Mobile Ad Hoc Networks. In *International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2007)*, pages 125–130, Papeete, French Polynesia (Tahiti), November 2007. IEEE Computer Society Press.

[18] R. Meier and V. Cahill. Steam: Event-based middleware for wireless ad hoc networks. In *Proceedings of The ICDCSW*, pages 639–644, 2002.

[19] A. Slominski, M. Govindaraju, D. Gannon, and R. Bramley. An Extensible and Interoperable Event System Architecture Using SOAP. Technical Report TR549, Department of Computer Science Indiana University Bloomington, IN, U.S.A., 2002.

[20] C.-F. Sorensen, M. Wu, T. Sivaharan, G. S. Blair, P. Okanda, A. Friday, and H. Duran-Limon. A context-aware middleware for applications in mobile ad hoc environments. In *MPAC '04: Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing*, pages 107–110, New York, NY, USA, 2004. ACM.

[21] J. van Gurp, A. Karhinen, and J. Bosch. Mobile service oriented architectures (mosoa). In F. Eliassen and A. Montresor, editors, *Proceedings of DAIS*, volume 4025 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2006.

[22] X. Yang, A. Bouguettaya, B. Medjahed, H. Long, and W. He. Organizing and accessing web services on air. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 33(6):742–757, 2003.

[23] S. S. Yau, F. Karim, Y. Wang, B. Wang, and S. K. S. Gupta. Reconfigurable context-sensitive middleware for pervasive computing. *IEEE Pervasive Computing*, 1(3):33–40, 2002.

Conditional Purpose Based Access Control Model for Privacy Protection

Md Enamul Kabir **Hua Wang**

Department of Mathematics and Computing
University of Southern Queensland,
Toowoomba, Queensland 4350, Australia,
Email: {kabir, wang}@usq.edu.au

Abstract

This paper presents a model for privacy preserving access control which is based on variety of purposes. Conditional purpose is applied along with allowed purpose and prohibited purpose in the model. It allows users using some data for certain purpose with conditions. The structure of conditional purpose based access control model is defined and investigated through a practical paradigm with access purpose and intended purpose. An algorithm is developed to achieve the compliance computation between access purposes and intended purposes. According to this model, more information from data providers can be extracted while at the same time assuring privacy that maximizes the usability of consumers' data. This model extends traditional access control models to a further coverage of privacy preserving in data mining atmosphere. Its interior is a new structure for managing collected data in an effective and trustworthy way. This structure helps enterprises to circulate clear privacy promise, to collect and manage user preferences and consent. The implementation of the idea in the paper shows the flexibility of the model, and finally we provide comparisons of our work to other related work.

Keywords: access control, access purpose, intended purpose, conditional intended purpose, prohibited intended purpose.

1 Introduction and Motivation

Privacy[1] preservation of individuals is a challenging problem in the data-mining environment. Enterprises collect customer's personal identification information along with other attributes during any kinds of marketing systems. It is a natural expectation that the enterprise will use this information for various purposes, this leads to concern that the personal data may be misused. Many enterprises collect, store and use huge amount of personal information. A study conducted by the Federal Trade Commission has shown that 97 percent of websites were collecting at least one type of identifying information such as name, e-mail address, or postal address of customers (Federal Trade Commission 2000). Privacy preservation in data-mining environment has become a great concern both for enterprises and individuals. As individuals are more concern about their privacy, they are becoming more reluctant to carry out their businesses and transactions online, and many organizations are losing a considerable amount of potential profits (Forrester Research 2001). The research shown that on-line commerce was reduced by US$15 billion in 2001 due to individual privacy concerns. These reactions from individuals imitate an altering awareness about how data is managed. Therefore without a clear compromising between individuals and enterprises, data quality and data privacy cannot be achieved and many organizations are seriously thinking about privacy issues of consumers. By demonstrating good privacy practices, many businesses are now trying to build up solid trust to customers, thereby attracting more customers (Baker & Peter 2003). Considering the privacy of customers, enterprise has to develop a secure privacy policy to remove the fear of customers. Thus in an internal management system, a reliable, efficient, effective and secure privacy policy should be established depending on customer's requirements.

A lot of work has been developed in order to protect the privacy of individuals and showed that the notion of purpose should be used as the basis for access control for specifying a privacy policy (Agrawal et al. 2002, Powers et al. 2002, LeFevre et al. 2004, Agrawal et al. 2005, Byun et al. 2005, 2008). According to Yang et al. (2007), a privacy policy ensures that data can only be used for its intended purpose (intended usage of data), and an access purpose (intension for accessing data objects) is compliant with the data's intended purpose. During the last few years, rapid technological developments especially in the field of information technology directed most attention and energy to the privacy protection of Internet users. Unless customers' data is suitably protected, individuals' privacy can be breached revealing their personal information. On the other hand, these collected data sets are the most important tools for a wide range of studies. Again the data that is more protected usually loses data quality. So it is necessary to come up a point where both data quality and data privacy are achieved. Although a significant number of works has been developed in this area (OASIS, Agrawal et al. 2002, Powers et al. 2002, LeFevre et al. 2004, Agrawal et al. 2005, Byun et al. 2005, 2008), research has yet to be done in order to remove the dilemma between data quality and data privacy.

Many privacy policy access control models have been proposed in order to protect the privacy of consumers. Byun et al. (2005, 2008) pointed out that privacy protection cannot be easily achieved by traditional access control models as it focuses on which user is performing which action on which data

[1] Privacy is defined as the right of an individual to decide when, how, and to what extent he/she would like to share his/her information

Table 1: Hypothetical data base illustrating AIP and PIP

name	age	address	income	name_{ip}	age_{ip}	address_{ip}	income_{ip}
Alice	35	21, West St., TBA, QLD 4350	35000	$\langle\{G\},\{\Phi\}\rangle$	$\langle\{\Phi\},\{G\}\rangle$	$\langle\{G\},\{A, S\}\rangle$	$\langle\{G\},\{M\}\rangle$
Bob	29	45, Fay CT., TBA, QLD 4350	23000	$\langle\{G\},\{\Phi\}\rangle$	$\langle\{G\},\{M\}\rangle$	$\langle\{G\},\{A, S\}\rangle$	$\langle\{G\},\{A,M\}\rangle$
Ron	56	20, Anita Dr., TBA, QLD 4350	56000	$\langle\{G\},\{\Phi\}\rangle$	$\langle\{G\},\{M\}\rangle$	$\langle\{G\},\{A, S\}\rangle$	$\langle\{G\},\{A\}\rangle$
Jak	48	25, Wuth St., TBA, QLD 4350	48000	$\langle\{G\},\{\Phi\}\rangle$	$\langle\{G\},\{M\}\rangle$	$\langle\{G\},\{A\}\rangle$	$\langle\{G\},\{A,M\}\rangle$

G={General purpose}, A={Admin purpose}, S={Shipping purpose}, P={Purchase purpose}, M={Marketing purpose}, ip={Intended purpose}=⟨AIP, PIP⟩

object. But a reliable privacy policy are concerned with which data object is used for which purpose. They also suggested that the notion of purpose must play a major role in access control models and that an appropriate metadata model must be developed to support such privacy centric access control models in order to protect data privacy. An approach is developed that is based on intended purposes, which specify the intended usage of data, and access purpose, which specify the purposes for which a given data element is accessed. Usually, during the data collection procedure customers are informed about the purposes of enterprises. Customers then decide whether their information could be used or not for a certain purpose. That means data providers are given an option of their data with certain purposes. If an individual mentions that his/her data could not be used for a certain purpose, then his/her information is not accessible for the purpose. Usually data providers are reluctant to use any part of their information for any purposes and so there is a possibility of losing information. But more information can be extracted from data providers by providing more possible options of using their information. It is possible to protect the privacy of individuals in this model, but there is a shortcoming of information loss. An intended purpose is divided (IP) into two parts: Allowed Intended Purposes (AIP) (explicitly allows to access the data for the particular purpose) and Prohibited Intended Purpose (PIP) (data access for particular purposes are never allowed). In order to recognize the model clearly, suppose that a company uses consumers' data for the purpose of General, Admin, Marketing and Shipping and consider the hypothetical database in Table 1.

In Table 1, the value of Alice's attribute income_{ip} is $\langle\{G\},\{M\}\rangle$, which means that Alice income could be used for General purpose but strictly prohibited to use for Marketing purpose. If we take a query

SELECT name
FROM Table 1
FOR Marketing Purpose

it gives the name of Alice, Bob, Ron, Jak and if we have a query

SELECT name, age
FROM Table 1
FOR Marketing Purpose

it returns nothing because prohibited intended purposes override the allowed intended purposes. This model protects privacy of consumers as it considers customers' requirements but it occurs more information loss. So a natural question arise

" Is it possible to extract information from PIP at least conditionally?"

The answer of this question is achieved in this article by adding a new term conditional purpose in the intended purpose. In order to extract more data and protect data privacy, conditional purpose plays a role in access control models. In this paper, we address this goal by presenting a model of purpose management, which is a fundamental building block on which conditional purpose based access control can be developed. Our proposed model is based on access purpose and intended purpose. Both access purposes and intended purposes are specified with respect to a hierarchical structure that organizes a set of purposes for a given enterprise. A key feature of our proposed model is that it supports conditional purpose and prohibited purpose, thus allowing users to specify that data should be used conditionally or should not be used for a set of purpose.

Observing these challenges and the satisfaction of both enterprises and customers, we need a better model to extract more information from customers with privacy guarantees. To overcome this challenge, we propose a new access control called conditional based access control model. In the access control model it is enable to extract information from PIP by giving conditions called Conditional Intended Purpose (CIP). Our proposed model is helpful for enterprises to establish an ideal privacy policy and to manage data in a sensitive, effective and trustworthy way. It also helps policy makers and the experts in the data-mining environment.

The reminder of this paper is organized as follows. We present a brief overview of privacy related technologies in Section 2. Since purpose is used as the basis of access control, a brief description of the notion of purpose is described in Section 3. In Section 4 we present comprehensive descriptions of our proposed access control model. Section 5 is devoted to compliance check and access control using query modification. We compare our proposed model with the most recent access control models in Section 6. Concluding remarks are included in Section 7.

2 Related Work

This work is related to several topics in the area of privacy preservation in data mining atmosphere.

The most notable technique to protect privacy is the W3C's Platform for Privacy Preferences (P3P) that formally specify privacy policy by service providers (Marchiori 2002). P3P provides a way for a web site to encode its data collection in a machine-readable format known as a P3P policy, which can be compared against a user's privacy preferences Yang et al. (2007). Byun et al. (2008) pointed out that P3P does not provide any functionality to keep promises in the internal privacy practice of enterprise. Thus it can be said that a striking privacy policy with inadequate enforcement mechanism may place the organizations at risk of reputation damage. The concept of Hippocratic database introduced by Agrawal et al. (2002) that amalgamates privacy protection in relational database system. A Hippocratic database includes privacy policies and authorizations that associate with each attribute and each user the usage purpose(s) (Al-Fedaghi 2007). Agrawal et al. (2002) presented a privacy preserving database architecture called Strawman which was based the access control on the notion of purposes, and opened up database-level researchers of privacy protection technologies. After that, purpose based access control introduced by Byun et al. (2005, 2008) and Yang et al. (2007), fine grained access control introduced by Rizvi et al. (2005) and Agrawal et al. (2005) are widely used access control models for privacy protection. In IT system the proposed Enterprise Privacy Authorization Language (EPAL) of IBM is a language for writing enterprise privacy policies to run data handling practices. An EPAL policy defines hierarchies of data-categories, user-categories, and purpose (Byun et al. 2008). A set of actions, obligations, and conditions are also defined by an EPAL policy.

A lot of works (Bertino et al. 1996, Denning et al. 1988, Sandhu & Chen 1991, 1998) provide many valuable insights for designing a fine-grained secure data model. In a multilevel relational database system, every piece of information is classified into a security level, and every user is assigned a security clearance (Byun et al. 2008). LeFevre et al. (2004) proposed an approach to enforcing privacy policy in database setting. This work focus on ensuring limited data disclosure, based on the premise that data providers have control over who is allowed to see their personal data and for what purpose. They introduced two models of cell-level limited disclosure enforcement and suggest an implementation based on query modification techniques.

Byun et al. (2008) present a comprehensive approach for privacy preserving access control model. In their access control model multiple purposes to be associated with each data elements and also support explicit prohibitions. This model is based on the notion of purpose as it plays a central role and is the basic concept on which access decisions are made. According to Byun et al. (2008) a purpose describes the reason(s) for data collection and data access, access purpose is intension for accessing data objects and intended purpose is the specified usages for which the data objects are collected. Massacci et al. (2005) pointed out that most privacy-aware technologies use purpose as a central concept around which privacy protection is built.

All of these works proposed different approaches to protect the privacy of individuals through different models without being considering to extract more information. Our aim is to preserve privacy of individuals as well as extracting more information. With this aim in this paper we propose a model that has significantly improved the work of Byun et al. (2008). It has improved in three different remarkable ways. First, we introduce conditional purpose in addition to explicit prohibitions that make data providers more flexible to give information. Second, the enterprise can publish an ideal privacy policy to manage data in a sensitive, effective and trustworthy way, and third it reduces the information loss as it shows that we can extract more information from data providers.

3 Purpose, Access Purpose and Intended Purpose

Table 2: Predetermined Intended Purposes

	Group 1	Group 2	Group 3
Name	$\langle\{G\},\{T\},\{\Phi\}\rangle$	$\langle\{G\},\{\Phi\},\{T\}\rangle$	$\langle\{G\},\{\Phi\},\{\Phi\}\rangle$
Address	$\langle\{G\},\{T\},\{\Phi\}\rangle$	$\langle\{G\},\{\Phi\},\{T\}\rangle$	$\langle\{G\},\{\Phi\},\{\Phi\}\rangle$
Phone	$\langle\{G\},\{T\},\{\Phi\}\rangle$	$\langle\{G\},\{\Phi\},\{T\}\rangle$	$\langle\{G\},\{\Phi\},\{\Phi\}\rangle$
Age	$\langle\{G\},\{T\},\{\Phi\}\rangle$	$\langle\{G\},\{\Phi\},\{T\}\rangle$	$\langle\{G\},\{\Phi\},\{\Phi\}\rangle$
Income	$\langle\{G\},\{T\},\{\Phi\}\rangle$	$\langle\{G\},\{\Phi\},\{T\}\rangle$	$\langle\{G\},\{\Phi\},\{\Phi\}\rangle$

Data is collected for certain purpose. For instance, a nation wide demographic survey in Australia, data may be collected to know the socioeconomic and demographic characteristics of all Australians. Each data access also serves a certain purpose. So it is a natural expectation that a privacy policy should concern which data object is used for which purposes. Many authors indicated that purpose is a central part in many privacy preserving access control model.

3.1 Definition of Purpose

For preserving the privacy of customers, each and every data access must obey with the privacy policies on which customers have conditionally or unconditionally agreed. A representative privacy policy for a data element includes purposes, retention, condition and obligation. This means that the particular data element can be conditionally or unconditionally accessed only for the specific purposes with certain conditions. The retention indicates how long the data element can be reserved, and the obligation designates the actions that must be followed after an access to the data element is approved. So purpose is the most interesting thing to researchers as it directly shows how access to data elements has to be controlled. P3P defines purpose as "the reason(s) for data collection and use" and specifies a set of purposes (World Wide Web Consortium). In commercial surroundings purposes normally have a hierarchical associations among them; i.e., generalization and specialization relationships. For instance, a group of purposes such as direct-marketing and third party marketing can be represented by a more general purpose, marketing. We borrow the purpose definition from Byun et al. (2008).

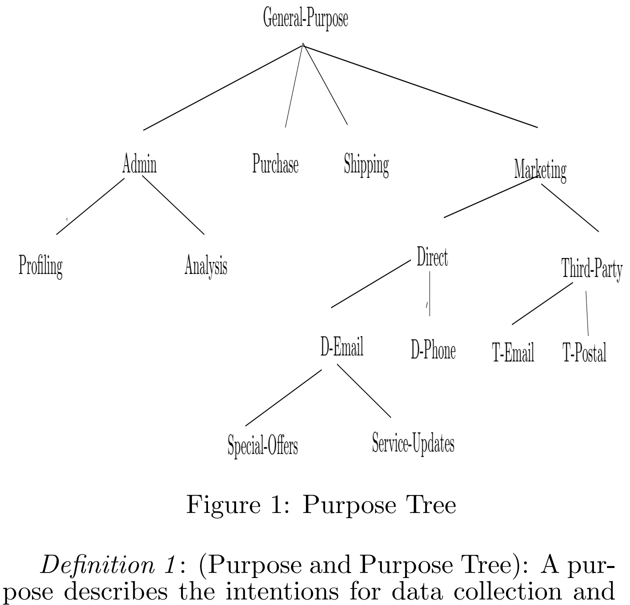


Figure 1: Purpose Tree

Definition 1: (Purpose and Purpose Tree): A purpose describes the intentions for data collection and data access. A set of purposes, denoted as ω, is organized in a tree structure, referred to as Purpose Tree and denoted as Ω, where each node represents a purpose in ω and each edge represents a hierarchical relation between two purposes. Let r_i, r_j, be two purposes in Ω. We say that r_i is an ancestor of r_j (or r_j is a descendent of r_i) if there exists a downward path from r_i to r_j in Ω. Figure 1 is an example of purpose tree, where each node represents a purpose in ω and each edge represents a hierarchical relation between two purposes.

Purposes, depending on their association with objects and subjects, may be called intended purposes or access purposes respectively.

Definition 2 (Access Purpose): An access purpose is intensions for accessing data objects, and it must be determined by system when data access is requested. So access purpose specifies the purpose for which a given data element is accessed.

Definition 3 (Intended Purpose): An intended purpose is the specified usages for which data objects are collected. That is, purpose associated with data and thus regulating data accesses as intended purpose. According to our approach an intended purpose consists of the following three components.

Allowable Intended Purpose (AIP): This means that data providers explicitly allow accessing the data for a particular purpose. For example data providers may consider that his/her information can be used for marketing purpose without any further restrictions.

Conditional Intended Purpose (CIP): This means that data providers allow accessing the data for a particular purpose with some conditions. For example data providers may consider that his/her income information can be used for marketing purpose by hiding his/her personal identification information (e.g. id or name etc.) or his/her income data can be reveal through generalization. or only the first letter of name can be used for marketing purpose.

Prohibited Intended Purpose (PIP): This means that data providers strictly disallow accessing the data for a particular purpose. For example data providers may consider that his/her income information cannot be used for marketing purpose. In that case data provider's income attribute is strictly prohibited to use for marketing purpose. An example of how AIP, CIP and PIP works is illustrated through a hypothetical database in Table 3.

So an intended purpose IP is a tuple $\langle AIP, CIP, PIP\rangle$, where AIP $\subseteq \omega$, CIP$\subseteq \omega$ and PIP$\subseteq \omega$ are three sets of purposes. The set of purposes implied by IP, denoted by IP*, is defined to be AIP$^\downarrow \cup$CIP$^\downarrow$ -PIP$^\updownarrow$, where

R$^\downarrow$, is the set of all nodes that are descendants of nodes in R, including nodes in R themselves,

R$^\uparrow$, is the set of all nodes that are ancestors of nodes in R, including nodes in R themselves, and

R$^\updownarrow$, is the set of all nodes that are either ancestors or descendants of nodes in R, that is, R$^\updownarrow$=R$^\uparrow$ $\cup$ R$^\downarrow$.

The following example explains the definition of AIP, CIP and PIP.

Example 1: Suppose IP= $\langle$\{Admin, Direct\}, \{Third-party\}, \{D-mail\}$\rangle$, then IP* = \{Admin, Profiling, Analysis, D-Phone, Third-party, T-Email$_c$, T-Postal$_c$\}. where subscripts c indicates that customers information can be used for the purpose with some conditions.

Definition 4 (Access Purpose Compliance): Let Ω be a purpose tree. Let IP= $\langle AIP, CIP, PIP\rangle$ and AP be an intended purpose and an access purpose defined over Ω, respectively. AP is said to be compliant with IP according to Ω, denoted as AP$_{\Leftarrow\Omega}$IP, if and only if AP$\in$ IP*.

Example 2: Suppose a company established the following privacy policies:

- We use your information for purchasing purposes, to provide services to you, and to inform you of services that may better meet your needs.
- We will disclose, conditionally disclose or will not disclose your information to third parties according to our privacy requirements.

In Table 2, Group 3 represents customers who have given consent for third-party marketing, Group 1 represents customers who have given consent for third-party marketing by removing personal identification information or via generalization/suppression (conditionally given consent) and Group 2 represents customers who have not given consent for third-party marketing.

4 Conditional purpose based access control

In our model data providers are asked three options for their data usage, permissible, prohibited and conditional permissible usages of each data item. For example, a data provider may select his/her name is permissible for **Admin** purpose, address is not permissible for **Shipping** purpose but income information is conditionally permissible for **Marketing** purpose. That is, data provider does not have any privacy concern over the name when it is used for the purpose of administration, great concern about privacy of the address information (and so does not want to disclose address) when it is used for the purpose of shipping, but his/her income information can be used for marketing purpose with some conditions. Here the term "conditions" means that data provider ready to release his/her certain information for certain purpose by removing his/her name or id or through generalization. This information is then stored in the database along with the collected data, and access to the data is tightly governed according to the data provider's requirements. For using the term condition data providers feel more comfortable to release their data. So according to our approach enterprise can establish an attractive privacy policy and it is possible to extract more information from data providers. Table 4 illustrates some imaginary records and intended purposes stored in a conceptual data base relation. Notice that each data element is stored in three different purposes each of which corresponds to a particular intended purposes.

As discussed before, our design of intended purposes supports permissive, conditions and prohibitive privacy policies. This construction allows more squash and flexible policies in our model. Moreover, by using CIP and PIP, we can assure that data access for particular purposes are allowed with some conditions and never allowed. Note that an access decision is made based on the relationship between the access purpose and the intended purpose of the data. Access is allowed only if the access purpose is included in the implementation of the intended purpose; in that case the access purpose is compliant with the intended purpose. The access is accepted with conditions if the implementation of intended purpose includes the access purpose with conditions; in this case we say that access purpose is conditionally complaint with intended purpose. The access is denied if the implementation of the intended purpose

Table 3: Hypothetical data base illustrating AIP, CIP and PIP

name	age	address	income	name_{ip}	age_{ip}	address_{ip}	income_{ip}
Alice	35	21, West St., TBA, QLD 4350	35000	⟨{G}, {Φ}, {Φ}⟩	⟨{Φ}, {M}, {A}⟩	⟨{G}, {Φ}, {A, S}⟩	⟨{G}, {A}, {M}⟩
Bob	29	45, Fay CT., TBA, QLD 4350	23000	⟨{G}, {Φ}, {Φ}⟩	⟨{G}, {M}, {Φ}⟩	⟨{G}, {M}, {A, S}⟩	⟨{G}, {M}, {A}⟩
Ron	56	20, Anita Dr., TBA, QLD 4350	56000	⟨{G}, {Φ}, {Φ}⟩	⟨{G}, {M}, {Φ}⟩	⟨{G}, {Φ}, {A, S}⟩	⟨{G}, {S}, {A}⟩
Jak	48	25, Wuth St., TBA, QLD 4350	48000	⟨{G}, {Φ}, {Φ}⟩	⟨{G}, {M}, {Φ}⟩	⟨{G}, {M}, {A}⟩	⟨{G}, {M}, {A}⟩

G={General purpose}, A={Admin purpose}, S={Shipping purpose}, P={Purchase purpose}, M={Marketing purpose}, ip={Intended purpose}=⟨AIP, CIP, PIP⟩

Table 4: Fictional records and intended purposes

	name	age	address	income
AIP	Alice	35	21, West St., TBA, QLD 4350	35000
CIP	A	30-40	West St., TBA, QLD 4350	30000-40000
PIP	⋆	⋆	⋆	⋆
AIP	Bob	29	45, Fay CT., TBA, QLD 4350	23000
CIP	B	20-30	Fay CT., TBA, QLD 4350	20000-30000
PIP	⋆	⋆	⋆	⋆
AIP	Ron	56	20, Anita Dr., TBA, QLD 4350	56000
CIP	R	50-60	Anita Dr., TBA, QLD 4350	50000-60000
PIP	⋆	⋆	⋆	⋆
AIP	Jak	48	25 Wuth St., TBA, QLD 4350	48000
CIP	A	50-60	Wuth St., TBA, QLD 4350	40000-50000
PIP	⋆	⋆	⋆	⋆

⋆ means data providers are reluctant of any usage of their data items

does not include the access purpose, in this case access purpose is not complaint with the intended purpose. Suppose in the online marketing system, an enterprise collects name, age, address and income of customers along with other information and the enterprise uses customer's information for the purpose of admin, shipping, purchase and marketing. Consider the hypothetical database in Table 3.

In Table 3, the value of Alice's attribute income_{ip} is ⟨{G},{A},{M}⟩ which means that Alice income could be used for General purpose but strictly prohibited to use for Marketing purpose. It also means that Alice income could be used for Admin purpose by hiding her personal identification information or through generalization. Similarly, Bob, Ron and Jak's income information could be used conditionally for Marketing purposes but their income information is strictly prohibited for Admin purpose.

4.1 Implementation

In our proposed model, users query the database using standard SQL statements. In this article we assume that each query is connected with a specific purpose. The data is accessible to each query varies depending on the data providers agreement and the purpose of the query. For example, any query against Table 3 with any purpose returns a result that is equivalent to the result of the query. As our proposed model directly reflect the information that is allowed, conditionally allowed or prohibited by each data provider, querying against these model does not violate privacy. This model is quite different from the conventional access control model as different sets of data may be returned for the same query depending on the purpose of the query and the data providers' agreements. Thus from the hypothetical database in Table 3, if we take the query

SELECT name, income
FROM Table 3
FOR Marketing Purpose,

then by using Table 4, we get the information in Table 5.

Table 5: Filtering information

Ron	56000
Bob	20000-30000
Jak	40000-50000

We can see from Table 5 that it gives name and income of Ron as he allows to disclose his name and income information for Marketing purpose. It also shows other two incomes via generalization as they conditionally allowed to disclose their income. This clearly shows the utility of using our proposed model. It demonstrates that it can extract more information from data providers.

Theorem 1: Let p, q and r denote the probability that a data provider gives consent of a particular attribute for AIP, PIP and CIP respectively. Assuming that these probabilities remain the same from data provider to data provider. Then the conditional based access control model extracts more information than the model proposed by Byun et al. (2008).

Proof: Let n be the total number of data providers. If p and q are the probabilities that a given data provider gives consent of a particular attribute for AIP and PIP. Then the average numbers of data providers who give consent for AIP is np. That means by using the model of Byun et al. (2008), the average number of data providers who give consent for AIP of a particular attribute is np. If we use our model then the average number of data providers who give consent to disclose their their data for a certain purpose with some conditions is nr. So by using the conditional based access control model total average number of data providers whose information is accessible is $(np + nr)$. Since n and p both are positive so $(np + nr)$ is always greater than np. This means that it is enable to extract more information from customers by using the conditional based access control model.

In our model, the collected data is used for different purposes on the basis of the data providers requirements. For using the CIP, both privacy and usability of data can be achieved as it filters out the values by performing a purpose compliance. By using a hypothetical database and the extracted outcome in Table 5, it shows clearly the data utility and data providers information is protected. It showed by Theorem 1 that our proposed model extracts more information with assuring privacy.

5 Access control

Among the various possible techniques to determine access purpose, in this paper we utilize the method where the users are required to explicitly state their access purposes when they try to access data. That is, users provide an access purpose for each query they issue.

5.1 Compliance Check

Consider the purpose tree in Figure 2 and it encoded into a relation pt-table as shown in Table 6. The first column *p_id* represents the identification number of each purpose node, the second column *p_name* represents the name of each purpose node, and the third column *parent* is used to capture the hierarchical relationships among purpose nodes. The column code is the binary encoding of each purpose. For example, in Table 6 the purpose B is encoded as '0×100' in hexadecimal representation, while the purpose E is encoded as '0×020' in hexadecimal form. The last three columns *aip_code*, *cip_code* and *pip_code* are precalculated encodings of purpose implications. As we know, when a purpose r_i is used as an AIP, it means that every descendant of r_i, including r_i itself is allowed. For example, the purpose D in Figure 2 used as an AIP implies that access is allowed for the purpose of D as well as G, H, I and J. Thus, the *aip_code* of D contains the implied set of D, which is the sum of the encodings of D, G, H, I and J. Note that *aip_code* and *cip_code* of each purpose is same as in the long run both are allowed. The *pip_code* of a particular purpose r_j is computed similarly by summing the encodings of every descendant and ancestor of r_j with the encoding of r_j itself.

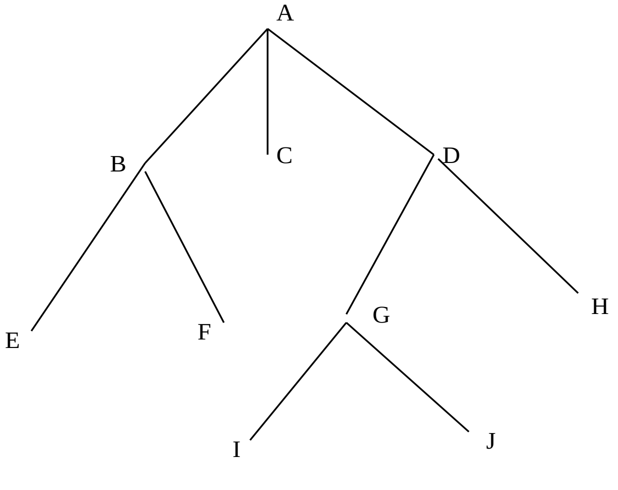


Figure 2: Purpose Tree

An access purpose is compliant with an intended purpose if and only if the access purpose is not prohibited by PIP and it is allowed by both AIP and CIP. Thus, the purpose compliance check can be done with two bitwise AND an operation as follows:

AIP, CIP and PIP, say *ap_code*, *aip_code*, *cip_code* and *pip_code* respectively, the access purpose is compliant with the intended purpose if and only if

$$\begin{aligned}&(\text{ap\_code \& cip\_code})=0 \wedge (\text{ap\_code \& aip\_code}) \neq 0 \vee\\&(\text{ap\_code \& pip\_code})=0 \wedge (\text{ap\_code \& aip\_code}) \neq 0 \vee\\&(\text{ap\_code \& pip\_code})=0 \wedge (\text{ap\_code \& cip\_code}) \neq 0.\end{aligned}$$

where, & is bitwise AND operator, $\wedge$ is logical AND operator and $\vee$ is logical OR operator. Conflicts among the AIP, CIP and the PIP for the same data element are resolved by applying the denial-takes-procedure policy where PIP overrides AIP and CIP, and CIP overrides AIP. The computation for purpose compliance check is illustrated in Table 7.

5.2 Query modification

It is a natural expectation that privacy-preserving access control techniques ensures a query result contains only the data items that are allowed or conditionally allowed or completely prohibited for the access purpose of the query. This expectation is achieved in this paper using query modification Stonebraker & Wong (1974). It is important to notify that query modification provides powerful and flexible controls without requiring any alteration in underling mechanisms and that it is supported in a major commercial Data Base Management System (Oracle Corporation 2002). Our query modification algorithm is outlined in Table 7.

The complexity of our query modification algorithm is in $O(n)$, where n is the number of attributes accessed by a given query. The method Modifying_Query is invoked only if the access purpose of the query is verified to be acceptable by the validate function. If the access purpose is unacceptable, then query is rejected without further being processed. The query modification algorithm checks both the attributes referenced in the projection list and the attributes referenced in predicates. As the attributes in the projection list determine what data items will be included in the result relation of a query, it may seem enough to enforce privacy policy based only on the attributes in the projection list. However, the result of a query also depends on the predicates, and not enforcing privacy constraints on the predicates may introduce inference channels. The abounding algorithm filters out a tuple if any of its elements that are accessed is conditionally allowed or prohibited with respect to the given access purpose. For example, consider a query,

SELECT name, income, address
FROM Table 3
FOR Marketing Purpose.

Suppose there is a customer record of which name is allowed for marketing, income is conditionally allowed for Marketing but the address is prohibited for this purpose. Then our algorithm only excludes address of this record from the query result and income information is visible anonymzing the customer's name or income information reveal via generalisation. So according to our proposed model

Table 6: Pt-table

p_id	p_name	parent	code	aip_code	cip_code	pip_code
1	A	-	0×200	0×3FF	0×3FF	0×3FF
2	B	1	0×100	0×130	0×130	0×330
3	C	1	0×080	0×080	0×080	0×280
4	D	1	0×040	0×04F	0×04F0	0×24F
5	E	2	0×020	0×020	0×020	0×320
6	F	2	0×010	0×010	0×010	0×310
7	G	4	0×008	0×00B	0×00B	0×24B
8	H	4	0×004	0×004	0×004	0×244
9	I	7	0×002	0×002	0×002	0×24A
10	J	7	0×001	0×001	0×001	0×249

income information of this customer is still usable for Marketing purpose instead of excluding other records.

The following example illustrates how our algorithm modifies queries. This example is a revised version of Byun et al. (2008) where purpose encoding of Marketing is assumed to be ‘0×200’. For the query

```
SELECT name, income
FROM table 3
FOR Marketing Purpose,
```

modified query becomes

```
SELECT name, income
FROM Table 3
WHERE Comp_Check(’0×200’,
name_aip, name_cip, name_pip)
AND Comp_Check(’0×200’,
income_aip, income_cip, income_pip).
```

Table 7: Query Modification Algorithm

```
Comp_Check (ap, aip, cip, pip)
/⋆ This function is required for query modification ⋆/
Returns Boolean
if (ap & cip)=0 and (ap & aip)$\neq$ 0
return True;
else if (ap & pip)=0 and (ap & aip)$\neq$ 0
return True;
else if (ap & pip)=0 and (ap & cip)$\neq$ 0
return True;
else False;

Modifying_Query (Query Q)
Returns a modified privacy-preserving query Q
Let $R_1,\cdots, R_n$ be the relations referenced by Q
Let P be the predicates in WHERE clause of Q
Let $a_1,\cdots,a_m$ be the attributes referenced in both
the projection list and P
Let AP be the access purpose encoding of Q
for each $R_i$ where i=1,.,n do
if (Comp_Check (AP, $R_i$.aip, $R_i$.cip, $R_i$.pip)=False) then
return ILLEGAL-QUERY;
end if;
end for;
return Q without modified P;
```

6 Comparison

There are some related works on privacy preservation. The closest works related to this article are Hippocratic databases (Agrawal et al. 2002) and purpose based access control model (Byun et al. 2008). In this section we will compare our proposed model with these two models.

Agrawal et al. (2002) proposed Hippocratic databases that incorporate privacy protection within relational database system. The proposed technique uses privacy metadata, which consist of privacy policies and privacy authorizations stored in two tables. The authors proposed a strawman design for Hippocratic databases. This design identified the technical challenges and problems in designing such databases. But the authors did not consider the concepts of purpose. By contrast, in our proposed model we investigated more sophisticated concepts of purpose. We used conditional purpose and the association of different purposes with a data element which are not considered in their work.

Byun et al. (2008) provided a comprehensive framework for purpose and data management. They argued that in order to protect data privacy, the notion of purpose must play a major rule in access control model. The authors proposed approach is based on intended purposes, which specify the intended usage of data, and access purposes, which specify the purposes for which a given data element is accessed. They also argued that traditional access control models focus on which user is performing which action on which data objects but privacy policies are concerned with which data object is used for which purposes. The authors proposed purpose based access control model (PBAC) allows multiple purposes to be associated with each data element and also supports explicit prohibitions. Although their proposed model designed on the basis of customers requirements and so does not violate privacy, the main drawback of this model is the information loss. In that model customers are given only two options whether their private data can be used or not for certain purposes instead of giving more possible options. But we strongly believe that by giving more options to customers data extractions can be easily achieved. By contrast, the proposed model in this paper provides three more options that help enterprises to extract more information from customers, assuring privacy. This criteria is achieved theoretically by Theorem 1 in Subsection 4.1. This clearly shows the utility and usability of our proposed model in a effective and trustworthy way.

7 Conclusion

Although privacy preserving desires a secure infrastructure and relies on access control technology, it is not a security problem but it is related to a data management problem. Purposes play a significant role in the field of database management system privacy preserving techniques. In this paper we introduced conditional based access control model for privacy protection in database system that enables enterprise to operate as a reliable keeper of their customers data. The basic concepts of the proposed conditional based access control model are discussed and it has shown the possibility to extract more information from customers by providing a secure privacy policy. The study reveals that this model achieves a better progress than the other access control models in the area of privacy preserving in data mining environment. The utility of the proposed conditional based access control model is illustrated through a practical example. We also discussed the algorithm to achieve the compliance check between access purpose and intended purposes. The effect of the proposed access control can be extremely useful for internal access control within an organization as well as well as information sharing between organizations. The enterprise can use this technology to enforce the privacy promises they make and to enable their customers to maintain control over their data. It would also help researchers, users and the associated people in the area of data mining.

Our proposed approach provides a complete structure for privacy preserving access control model. On the basis of this approach, a significant further work still needs to be done. Our future work includes extending this model in the Role-based Access Control (RBAC), Dynamic Purpose-based Access Control (DPBAC) and in the other access control systems.

References

Agrawal, R., Kiernan, J., Srikant, R. & Xu, Y. (2002), Hippocratic databases, *in* '28th International Conference on Very Large Databases (VLDB)'.

Agrawal, R., Bird, P., Grandison, T., Kiernan, J., Logan, S. & Xu, Y. (2005), Extending relational database systems to automatically enforce privacy policies, *in* 'ICDE', pp.1013-1022.

Al-Fedaghi (2007), Beyond Purpose-based privacy access control, *in* '18th Australian Database Conference (ADC)'.

Barker, S. & Stuckey, P.N. (2003), Flexible access control policy specification with constraint logic programming, *in* 'ACM Transaction on Information and System Security', Vol. 6(4), November.

Bertino, E., Jajodia, S. & Samarati, P. (1996), Database security: Research and practice, *in* 'Information systems'.

Bertino, E., Byun, J.W., & Li, N. (2005), Privacy-Preserving database system, *in* 'FOSAD', pp. 178-206.

Byun, J., Bertino, E. & Li, N. (2005), Purpose based access control of complex data for privacy protection, *in* 'Symposium on Access Control Model And Technologies (SACMAT)'.

Byun, J., Bertino, E. & Li, N. (2008), 'Purpose based access control for privacy protection in relational database systems', *VLDB J* **17**(4), 603–619.

Denning, D., Lunt, T., Schell, R., Shockley, W. & Heckman, M. (1988), The seaview security model, *in* 'The IEEE Symposium on Research in Security and Privacy'.

Federal Trade Commission(2000), Privacy online: Fair information practices in the electronic marketplace: A report to congress, May. Available at www.ftc.gov/reports/privacy2000/privacy2000.pdf.

Forrester Research (2001), Privacy concerns cost e-commerce $15 billion. Technical report, September.

IBM, The Enterprise Privacy Authorization Language (EPAL). Available at www.zurich.ibm.com/security/enterprise-privacy/epal.

LeFevre, K., Agrawal, R., Ercegovac, V., Ramakrishnan, R., Xu, Y. & DeWitt, D. (2004), Disclosure in Hippocratic databases, *in* 'The 30th International Conference on Very Large Databases (VLDB)', August.

Marchiori, M. (2002). The platform for privacy preferences 1.0 (P3P1.0) specification. Technical report, W3C, April.

Massacci, F., Mylopoulos, J. & Zannone, N. (2005), Minimal Disclosure in Hierarchical Hippocratic Databases with Delegation, *in* 'The 10th Europran Symposium on Research in Computer Security', September 12-14.

OASIS, Core and hierarchical role based access control (rbac) profile of xacml v2.0. Available at http://www.oasis-open.org/.

OASIS, Extensible access control markup language (xacml) 2.0. Available at http://www.oasis-open.org/.

OASIS, Privacy policy profile of xacml v2.0. Available at http://www.oasis-open.org/.

Oracle Corporation (2002), The Virtual Private Database in Oracle9iR2: An Oracle Technical White Paper,January, Available at www.oracle.com.

Rizvi, S., Mendelzon, A. O., Sudarshan, S. & Roy, P. (2004), Extending query rewriting techniques for fine-grained access control, *in* 'SIGMOD Conference', pp.551-562.

Powers, C.S., Ashley, P. & Schunter, M. (2002), Privacy promises, access control, and privacy management, *in* 'The 3rd International Symposium on Electronic Commerce'.

Sandhu, R. & Jajodia, S. (1991), Toward a multilevel secure relational data model, *in* 'ACM Transactional Conference on Management of Data (SIGMOD)'.

Sandhu, R. & Chen, F. (1998), The multilevel relational data model, *in* 'ACM Transaction on Information and System Security'.

Stonebraker, M. & Wong, E. (1974), Access control in a relational database management system by query modification, *in* 'ACM CSC-ER Proceedings of the 1974 Annual Conference', January.

World Wide Web Consortium (W3C), Platform for Privacy Preferences (P3P), Available at www.w3.org/P3P.

Yang, N., Barringer, H. & Zhang, N. (2007), A Purpose-Based Access Control Model, *in* 'Third International Symposium on Information Assurance and Security', pp. 143-148.

Information Retrieval in Structured Domains

Vincent W. L. Tam and John Shepherd
School of Computer Science and Engineering
University of New South Wales,
UNSW Sydney, NSW 2052, Australia

vincetam@cse.unsw.edu.au and jas@cse.unsw.edu.au

Abstract

In this work, we investigate utilizing the structure of a website to increase the effectiveness of document retrieval within a structured domain. In particular we examine various methods to combine evidence within the website in order to improve the quality of pages returned.

Keywords: Information retrieval, Structured IR, Passage retrieval

1 Introduction

Information retrieval is a broadly studied topic. Significant research efforts have been focused on document retrieval from World Wide Web. We aim to refine document retrieval within a website by improving the quality of document relevance against queries. We achieve this by taking into consideration the evidence collected from pages that are related to the document under inspection.

Websites are normally organised according to some structure (based on an information architecture) to make it more convenient for users to navigate the site. Often, the URL structure of pages reflects this organisation. These observations raise the issue of whether we can make use of the structure/organisation to improve search. The work in this paper sets out to explore this issue by trying to answer the following questions

1. Do surrounding pages of articles carry useful information to improve the quality of results in ranking documents against queries?
2. How to define the set of related pages for the above purpose and how to define the range of this set?

Our approach to answering these questions was to conduct information retrieval experiments on websites that were known to conform to a well-defined, hierarchical structure. The goal of these experiments was to determine how to use the information in related pages to improve relevance scoring. Such experiments, of course, could prove only that the approach is effective for sites that follow this structure.

2 Related Works

In this section we review several streams of research that motivated our experiment.

2.1 URL Structure

URLs of web pages have already been used to improve retrieval results. Keywords in URLs usually provide hints for information retrieval, and this has been utilized in search engines (and by "search engine optimisers") to enhance rankings of retrieved pages. There are also known uses of URLs as evidence to categorize pages in websites (e.g. Kules, Kustanowitz and Shneiderman 2006, Shih, and Karger 2004) where the pages are organised into hierarchies of subjects within the website. Under this assumption, URLs of web pages provide information on how the pages are categorized. Clearly, not all websites follow such conventions (e.g. many of the increasing number of dynamically-generated websites). However, a sufficient number of websites are organised by URL to make it worthwhile to consider this approach.

2.2 XML Element retrieval

A major stream of research that is related to our work is information retrieval in structured documents. This research focuses mainly on text retrieval from XML documents. XML documents are well-structured articles with tags to define elements within the articles. Information retrieval from XML documents aims to retrieve elements that closely match the queries. This stream of research is inspired by the Initiative for the Evaluation of XML retrieval (INEX). Our work differs from XML retrieval in that our targeted documents are individual pages within a website instead of elements contained in articles. Elements contained in articles have a well-defined unit (the article) to draw information about the context of the elements from (e.g. Kimelfeld, Kovacs, Sagiv and Yahv 2007). On the other hand, there is no clear boundary for this part-whole relationship for web pages. The range and number of documents to be included as related pages is not well-defined. To identify such boundaries was part of our research objective. A second difference between our work and XML retrieval is that queries in XML retrieval can specify the context of the desired results via Xpath. This is the case if the schema of the XML documents is known beforehand (e.g. Beigbeder 2007, Carpineto, Romano and Caracciolo 2007). Besides, the element tags of XML documents carry additional information for retrieval in the form of element attributes and element names. This helps in defining the function of the elements (e.g. Abstract, summary, title). In our setting, such information is not available to assist with document retrieval.

2.3 Passage Retrieval

Another stream of research that is similar to our work is passage retrieval. This research focuses mainly on enhancing search results by returning passages within documents instead of the whole documents, or enhancing document retrieval by collecting evidence of relevance from individual passages (Wilkinson 1994). It has been shown that combining evidence from this part-whole relationship helps in returning more relevant passages during retrieval (Callan 1994, Sigurbjörnsson, Kamps and Rijke 2004). Our research was inspired by this finding, and adopts a similar approach to draw evidence from surrounding pages to enhance retrieval results.

2.4 Structures among web pages

Previous works on document retrieval have also explored the use of hyperlink structure. PageRank (Brin and Page 1998) and HITS (Kleinberg 1999) are two of the most widely used algorithms in this category. This work differs from our research in that it focuses mainly on the association between different websites. PageRank and HITS assigned a score of authority (together with a hub in the case of HITS) to websites. They calculate relevance of documents by incorporating this score during retrieval. This is not appropriate in our setting of retrieving pages from a single website. If we used such schemes, retrieval results would always be biased towards authority pages/page-groups, regardless of what users put in as queries. In particular we do not want to focus on ranking pages by their global importance within the site.

In addition, search engines like Google (Brin and Page 1998) utilize information propagation to enhance retrieval results. In particular, search engines propagate words from link anchors to their target pages and these anchor words play an important role during retrieval (Glover, Tsioutsiouliklis, Lawrence, Pennock and Flake 2002). This is similar to our proposal of using extra information from other pages, but instead we are looking at words from pages that surround the target page rather than words that refer to this page.

2.5 Vector space model and Cosine similarity

The vector space model (Salton 1971) and the cosine similarity algorithm are widely used to rank documents against queries. Our experiments calculated relevance based on these models. We extended the scoring mechanism of documents against queries by combining relevance score of surrounding pages returned by these models. Since the focus of this research was to examine the effectiveness of combining evidence collected from related pages in a structured domain, we did not in particular examine and compare the uses of other models e.g. BM25 (Robertson, Walker, Jones, Hancock-Beaulieu and Gatford 1994), nor any other approaches of adopting the vector space model. Given the popularity of such models we believe that this provided a reasonable and understandable platform to perform such tests.

3 Method

3.1 Structure of a website

We first describe the relationships among documents by using the URL of each page to construct a relationship graph.

Exploring page relationships by URLs is comparatively cheap in processing and is readily available. Websites often use a directory/folder hierarchy to reflect the organisation of information in the site, and this structure is reflected in the URLs. Our approach to determine relatedness of pages attempts to exploit this by considering that if a page was included in a folder it is likely that this page has a similar "context" to other pages under the same folder. Sub-folders typically contain documents which specialise the context of their parent folder. Users of such sites often exploit the hierarchical structure as a basis for navigating through the site. Figure 1 shows a scenario with a hierarchical collection of folders, along with their corresponding URLs.

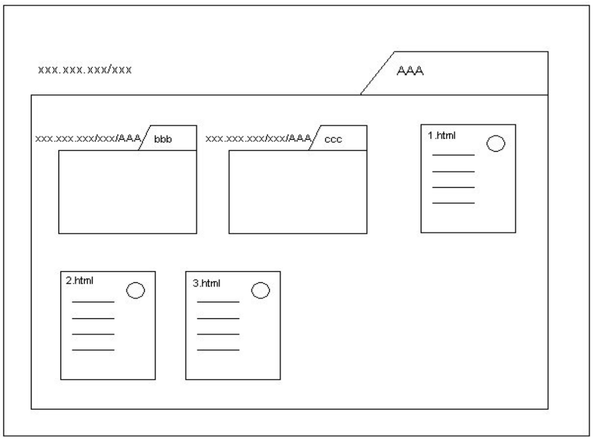


Figure 1: A folder-subfolder relationship scenario. Folders "bbb" and "ccc" are sub-folders in "AAA". The URLs of the folders reflect this hierarchy.

The first step is to establish an ancestor/descendent relationship among web pages, based on their URLs. If D is a page in a web-site, the URL(D) denotes its web address. The URLs of index pages (e.g. `index.html`) are normalised by removing the page component and treating them as a directory name. A page D_1 is defined to be an ancestor of another page D_2 if URL(D_1) is a prefix of URL(D_2), i.e.

$$\text{URL}(D_2) = \text{URL}(D_1)\ /*$$

Based on the ancestor/descendent relationship, we introduce a distance function Dist(D_1,D_2) which measures the number of pages *d* separating the two pages along their URL paths. Dist(D_1,D_2) is defined as follows:

Dist(D_1, D_2) = 0, if D_1 and D_2 is not related.
Dist(D_1, D_2) = d, if D_2 is a descendent of D_1
Dist(D_1, D_2) = -d, if D_1 is a descendent of D_2

3.2 Relevance of Documents

We adopt the vector space model and cosine similarity to calculate the similarity *sim(q,d)* of a document against a query:

$$sim(q,d) = \frac{\sum (w_{t.d} \cdot w_{t.q})}{\sqrt{\sum w_{t.d}{}^2 \times \sum w_{t.q}{}^2}}$$

where $w_{t.d}$ is the tfidf score of a term *w* in the document *d*, given by the following formula:

$$w_{t.d} = tf \times \ln(\frac{N}{n})$$

tf = term frequency of the term w in the document
N = total number of indexed document
n = total number of documents that contain w.

Documents are ranked by their *sim(q,d)* score.

3.3 Implementation

We store every term in the index. Terms are stemmed using the Porter Stemming algorithm (Rijsbergen, Robertson and Porter 1980) and stored with their term frequencies and tfidf scores. The sum of square of terms' tfidf for each page (the $w_{t.d}{}^2$ part in sim(q,d)) was also calculated and stored in the index for more efficient retrieval.

To obtain the distance measure Dist(D_1,D_2), we compare page URLs retrieved from the index at runtime. The URLs stored in the index are pre-processed as follows: removing duplicate URLs, removing redundant '/' characters, removing index page filenames, appending '/' to the end of URLs for index pages. Dist(D_1,D_2) is then calculated by counting the difference in the number of '/' characters in the URLs if one of their URLs is a prefix of the other.

3.4 Evidence from other documents

We combine the relevance score of surrounding documents with the initial score obtained from cosine similarity. We introduce a variable, factor λ, to adjust the relative weights of the surrounding evidence score and the initial similarity score. The λ factor has a value between 0 and 1; its use is shown below.

We also introduce a variable *rLimit* to adjust the definition of "related pages". To be precise, *rLimit* is the maximum distance allowed between two pages for them to be treated as related pages. For example, if *rLimit* is set to 1 we only considered pages that are one document away from the one we are looking at along the path of URL.

3.4.1 First Approach

Our first attempt accumulated the relevance score directly from related documents. The relevance score for document *d* with respect to query *q* is given by:

$$F_1(d) = (1-\lambda) \times sim(q,d) + \sum_{d_n \in R} sim(q,d_n) \times \sigma$$

$$\sigma = \lambda^{\log_2(|dist(d,d_n)|)}$$

where *R* is the set of documents with dist(d, d_n) ≠ 0, and |dist(d, d_n)| <= *rLimit.*

Notice that in this formula, we take the absolute value of dist(d,d_n) in defining the set R. We therefore do not explicitly distinguish between ancestor pages and descendent pages. The factor σ is introduced to account for the fact that the further away a document is from *d*, the smaller the effect it should be affecting *d*.

3.4.2 Second Approach

The first approach we adopted is a simple framework to combine scores of surrounding documents. There were two shortcomings in this attempt. Firstly it did not take into account sibling documents in the structure we introduced. Secondly there existed a bias to pages that had a lot of related pages. These were usually pages that were indices to folders that contained many child pages. To account for these we introduce our second set of formulae for document relevance. Note that the score is computed in two stages as described below.

Stage I

$$F_2"(d) = (1-\lambda) \times sim(q,d) + \sum_{d_n \in R} sim(q,d_n) \times \sigma$$

$$\sigma = \lambda^{\log_2(|dist(d,d_n)|)}$$

where *R* is the set of documents with dist(d, d_n) < 0, and |dist(d, d_n)| <= *rLimit.*

Stage II (final score used)

$$F_2(d) = (1-\lambda) \times F_2"(d) + \sum_{d_n \in R} F_2"(d_n) \times \sigma$$

$$\sigma = \lambda^{\log_2(dist(d,d_n))}$$

where *R* is the set of documents with dist(d, d_n) > 0, and dist(d, d_n) <= *rLimit.*

We split the scoring process into two stages. In the first stage we accumulate cosine similarity scores from descendents of pages. In the second run, we do the reverse, accumulating similarity scores returned in the first run from ancestors. The motivation for doing so was that in our setting in defining website structure, a document could have more than one child page, while each document belongs to a single parent page. In other words, a document belongs to one and only one folder, but a folder usually consists of more than one document. Therefore splitting the accumulation of score into two stages ensures that the child pages can share the biased scores from their parents returned in the first run.

Another advantage of this formula is that it captures the effect of both the parent-child relationship and the effects of sibling pages in the folder. This is because in obtaining the results during the first stage, the parent's score has been affected by all of its children. Therefore in the second run, when we combine evidence collected from parent pages with their updated similarity scores, the effect of sibling pages is propagated to every child page of the parent. Figure 2 shows an example of this.

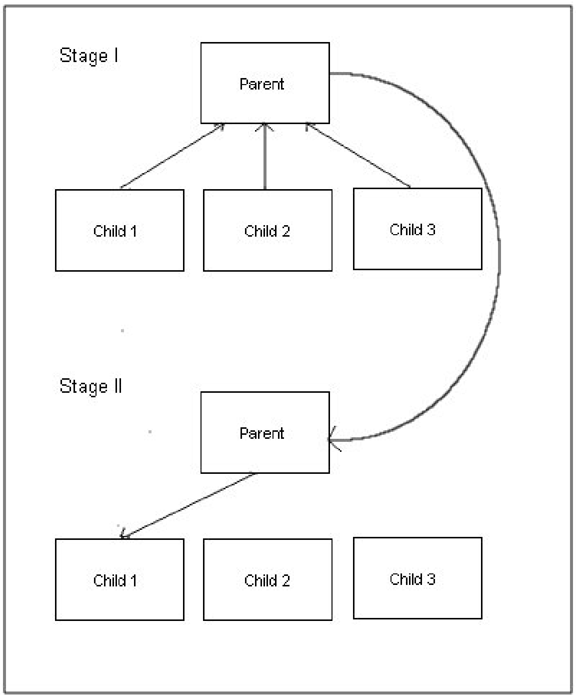


Figure 2: An example illustrating how evidence from sibling pages is propagated to "Child 1".

3.4.3 Third Approach

The first two attempts in our experiments accumulated the *sim(q,d)* score of surrounding documents as evidence to refine the relevance of the documents during retrieval. In our third formula we also wanted to take into account the length of each document explicitly. Not only does this give us a closer approximation to the original cosine similarity approach, but it also accounts for the bias to pages with more children mentioned above, without the need to split the process into two runs. This is done by dividing the sum of the dot product for each related page, after taking the factor λ into account, by the length of all documents that have been included in the calculation:

$$F_3(d) = \frac{\sum (w_{t.d} \cdot w_{t.q}) \times \sigma + \sum_R \sum_{d_n} \frac{w_{t.d_n} \cdot w_{t.q}}{\alpha}}{\sqrt{\left[\sum w_{t.d}{}^2 + \sum_R \sum_{d_n} \left(\frac{w_{t.d_n}}{\alpha}\right)^2\right] \times \sum w_{t.q}{}^2}}$$

$$\sigma = (1 - \lambda)$$

$$\alpha = \log_2(|dist(d, d_n)|)$$

where *R* is the set of documents with dist(d, d_n) ≠0, and |dist(d, d_n)| <= *rLimit*.

Notice that this formula is similar to the cosine similarity function, the difference being that we have factored down the effect of related documents by their distance from the inspecting document. The formula also treats every document individually in calculating the vector length instead of adding all the vectors before calculating the length. Given the sparsely distributed vector space property of text in a retrieval system we believe this is a reasonable approximation, and avoids the need to re-calculate variable document length at runtime. As mentioned in section 3.3, the sum of squares of $w_{t.d}$ was pre-calculated and stored in the index and therefore was readily available.

4 Experiments

We tested our algorithms on the websites of computing courses at the University of New South Wales. Our sample websites contained on average more than 1500 pages each. A set of queries was derived from questions that were asked on the course forums, so should represent realistic requests for information from the course websites.

In the evaluation we manually determined if a retrieved page was relevant, relevant but not helpful, or irrelevant from a number of subjects. We then assigned a score of 2, 1 and 0 respectively to the retrieved documents and took the average from the manual scores. We evaluated the effectiveness of different approaches by picking the first five ranked documents for each approach and calculating the total relevance score for these documents. The final result of each run was expressed in a precision ratio calculated from the abovementioned method. The tests were carried out by varying the factor λ in each formula. We also obtained two runs for each formula by setting *rLimit* to 1 and to infinity respectively.

The larger λ is, the more we weight evidence collected from related documents. When *rLimit* was equal to 1, the set of related pages were limited to pages that are directly related along the URL path. In other words, only those that surround the pages were used. On the other hand with *rLimit* set to infinity we took into consideration all pages along the path, with the effect of pages being factored by the distance from the inspecting page as described in our formulae. We compared each run with the baseline method of cosine similarity.

5 Results and Discussion

The precision ratios for our example queries are presented as follows.

Query	Baseline	F_1	F_1	F_1
		(λ =0.25)	(λ =0.5)	(λ =0.75)
1	0.3	0.2	0.2	0.2
2	0.3	0.4	0.4	0.4
3	0.3	0.3	0.1	0.1
4	0.7	0.4	0.3	0.3
5	0.2	0.4	0.4	0.4
6	0.1	0.3	0.3	0.3
7	0.1	0.3	0.3	0.3
8	0.2	0.2	0.2	0.2
9	0.3	0.1	0.3	0.4
10	0.3	0.4	0.2	0.1
11	0.2	0.2	0.1	0.2
12	0	0.1	0	0.1
13	0.2	0.3	0.3	0.1
14	0.6	0.8	0.8	0.8
15	0.4	0.3	0.2	0
Average	**0.28**	**0.31**	**0.27**	**0.26**

Table 1: F1 with *rLimit* = 1

Query	Baseline	F_1	F_1	F_1
		(λ =0.25)	(λ =0.5)	(λ =0.75)
1	0.3	0.2	0.2	0.2
2	0.3	0.3	0.3	0.3
3	0.3	0.1	0.1	0.1
4	0.7	0.4	0.3	0.3
5	0.2	0.4	0.4	0.4
6	0.1	0.3	0.3	0.3
7	0.1	0.2	0.2	0.2
8	0.2	0	0.2	0.2
9	0.3	0.1	0.3	0.4
10	0.3	0.2	0.1	0.1
11	0.2	0.1	0.1	0.1
12	0	0.1	0	0.1
13	0.2	0.1	0.1	0.1
14	0.6	0.8	0.8	0.8
15	0.4	0	0	0
Average	**0.28**	**0.22**	**0.23**	**0.24**

Table 2: F1 with no *rLimit*

Table 1 and Table 2 compare the results of retrieval using Formula 1 against the baseline method, which was cosine similarity. While table 1 showed the ability to obtain a similar result to the baseline method, table 2 showed inferior results. As we had earlier point out, Formula 1 would bias to pages that contained many child pages and we found that this was exactly the case when we looked in details into the pages returned by the algorithm. Nevertheless, by setting *rLimit* = 1 the results had been better than that without *rLimit*. In particular we obtained better results than the baseline when *rLimit* = 1 and λ = 0.25. This showed that under certain circumstances, we did benefit from collecting context evidence from surrounding documents. The implication of having better results with $\lambda < 0.5$ was that while we took into account of related documents, we shouldn't forget the importance of the initial similarity score of the documents themselves. In other words, the initial score of documents should still be the major deciding factor, yet we could benefit from taking into account other evidence with comparatively less weight.

Query	Baseline	F_2	F_2	F_2
		(λ =0.25)	(λ =0.5)	(λ =0.75)
1	0.3	0.6	0.6	0.6
2	0.3	0.4	0.3	0.3
3	0.3	0.2	0.2	0.2
4	0.7	0.9	0.8	0.8
5	0.2	0.4	0.2	0.2
6	0.1	0.3	0.1	0.1
7	0.1	0.3	0.2	0.2
8	0.2	0	0	0
9	0.3	0.5	0.6	0.6
10	0.3	0.6	0.3	0.3
11	0.2	0.1	0.4	0.3
12	0	0	0	0
13	0.2	0.3	0	0
14	0.6	0.8	0.6	0.6
15	0.4	0.2	0	0
Average	**0.28**	**0.37**	**0.29**	**0.28**

Table 3: F2 with *rLimit* = 1

Query	Baseline	F_2	F_2	F_2
		(λ =0.25)	(λ =0.5)	(λ =0.75)
1	0.3	0.6	0.6	0.6
2	0.3	0.3	0.4	0.3
3	0.3	0.2	0.2	0.2
4	0.7	0.9	0.8	0.8
5	0.2	0.4	0.2	0.2
6	0.1	0.3	0.1	0.1
7	0.1	0.1	0.2	0.2
8	0.2	0	0	0
9	0.3	0.5	0.6	0.6
10	0.3	0.2	0.2	0.3
11	0.2	0.3	0.4	0.3
12	0	0	0	0
13	0.2	0.1	0	0
14	0.6	0.8	0.6	0.6
15	0.4	0.2	0.2	0.2
Average	**0.28**	**0.33**	**0.30**	**0.29**

Table 4: F2 with no *rLimit*

Table 3 and Table 4 compare the results of retrieval using Formula 2 against the baseline method. This formula was designed with the aim to alleviate the bias towards the parent-child relationship presented with our proposed way to define website structure. In addition to that, the formula also took advantage of capturing the effect of sibling pages. The effect of this was obvious. As seen in the tables, we succeeded in improving retrieval results from that of Formula 1 in all cases.

From Table 4 we see that we obtained a much improved results when comparing the results to Table 2. With no *rLimit*, the bias in F1 we mentioned above would propagate along the path to every level in the hierarchy and therefore deteriorate document relevance. Although we had factored down the effect of pages that are further apart by the σ factor, the accumulation of scores from a group of pages might have too large an effect and therefore pages were still heavily affected. On the other hand having alleviated the bias with F2 we could see from Table 4 that the retrieval quality benefited from taking context along the path. The best result was obtained when λ = 0.25.

Similarly, from table 3, we had the best retrieval results when λ = 0.25. When comparing Table 3 and Table 4 we found improvement in results when *rLimit* = 1 and λ = 0.25. This is similar to the findings with Formula 1.

Query	Baseline	F_3 (λ =0.25)	F_3 (λ =0.5)	F_3 (λ =0.75)
1	0.3	0.4	0.4	0.2
2	0.3	0.4	0.4	0.5
3	0.3	0.2	0.2	0.2
4	0.7	0.4	0.4	0.4
5	0.2	0.2	0.2	0.1
6	0.1	0.3	0.3	0.3
7	0.1	0.3	0.4	0.2
8	0.2	0	0	0
9	0.3	0.1	0.2	0.2
10	0.3	0.7	0.7	0.5
11	0.2	0.2	0.2	0.2
12	0	0	0.1	0.1
13	0.2	0.3	0.3	0.3
14	0.6	0.6	0.6	0.6
15	0.4	0.4	0.4	0.4
Average	**0.28**	**0.30**	**0.32**	**0.28**

Table 5: F3 with *rLimit* = 1

Table 5 and Table 6 show the results of applying Formula 3 to the retrieval. While we managed to obtain slightly better results with the case when setting *rLimit* = 1, the results with no *rLimit* is less obvious. Formula 3 incorporated distance into the calculation of documents scores. We obtained the best results with λ = 0.5 and *rLimit* = 1. When comparing the approach of Formula 3 with the other formulae, Formula 3 was designed so that we did not have to worry about the effect of having very many related documents, which was the cause of the poor result cases when using Formula 1. In comparing results of Formula 1 and Formula 3 we found that this has been successful. However Formula 2 is superior to Formula 3. We believe that splitting the process into two runs, not only alleviates the problem of having too many related documents, but also more effectively takes into account the similarity score of sibling pages.

Query	Baseline	F_3 (λ=0.25)	F_3 (λ =0.5)	F_3 (λ =0.75)
1	0.3	0.4	0.4	0.2
2	0.3	0.3	0.3	0.4
3	0.3	0.2	0.2	0.2
4	0.7	0.4	0.4	0.4
5	0.2	0.2	0.2	0.1
6	0.1	0.3	0.2	0.2
7	0.1	0.2	0.2	0.2
8	0.2	0	0	0
9	0.3	0.1	0.2	0.2
10	0.3	0.6	0.6	0.5
11	0.2	0.1	0.1	0.2
12	0	0	0.1	0.1
13	0.2	0.3	0.3	0.3
14	0.6	0.6	0.6	0.6
15	0.4	0.2	0.4	0.4
Average	**0.28**	**0.26**	**0.28**	**0.27**

Table 6: F2 with no *rLimit*

To sum up, the best results were obtained from Table 3 when Formula 2 was used, with λ = 0.25 and *rLimit*=1. The worst results were obtained when we used Formula 1 without setting *rLimit*. This reinforced our belief that the use of evidence without distinguishing pages as ancestors or descendents would be inferior in our setting because there is a 1-to-m relationship, whereas a parent could have more than one child page and therefore the accumulation of evidences from them resulted in bias to these pages. Our two attempts to solve the bias, namely by splitting the runs into two separate single direction accumulation of score along the paths of related pages so as to allow the other pages to share this bias, and to take document length into account, have been successful. Apart from Formula 1, our results suggest that retrieval is more effective when context from related pages is taken into consideration.

In any case, we observed that we always obtained better results if used only the immediate surrounding pages as related pages (i.e. *rLimit* = 1). This suggests an answer to our second research question of "how to define the ranges of related pages to assist in improving the results of information retrieval?" We believe the reason for this was that the further away an ancestor is from a page, the more general context it has (and thus is less directly relevant to the query). On the other hand the further away a descendent is from a page, the more specific context it carries (and this context may be irrelevant to the query). Nonetheless in all of our approaches we observe that retrieval results do benefit from taking surrounding context into calculation.

Another point that is worth noting is that the first two formulae we used depend on the cosine similarity that we are comparing to. In other words, these aimed to improve the retrieval results among a pool of already retrieved documents. This can be observed by the fact that the individual score of each run on each query did not vary much. If the cosine similarity method was not able to draw the relevant documents set from the index then Formula 1 and Formula 2 can at most improve slightly on the results but wouldn't had a much better results returned. On the other hand Formula 3 is less dependent on the original results set given by cosine similarity and we therefore observed cases that either improve much or vice versa. Nevertheless, from our observation, with the right settings of environment variable λ and *rLimit*, the retrieval results were improved.

6 Conclusion and Future works

In this work, we have conducted preliminary experiments to show that for websites where the underlying domain structure is reflected in the URLs of the documents, retrieval results can be improved by taking into account evidence collected from related articles. We utilized the URLs in order to explore the hierarchy of the website and draw related pages from this hierarchy. We also showed that it is most effective when only immediately surrounding documents were used instead of taking into account every document along the related path. Although we are yet to perform further experiments on approaches other than re-ranking the documents returned by cosine similarity, our experiment has shown that it is worthwhile to draw evidence of context from other documents in the website.

Having gained encouraging results from our tests, the next stage is to perform more tests (more queries, more webistes) to provide a stronger base of evidence for such effects. In addition we aim to improve our formula in calculating similarities of documents and queries. In particular, as mentioned in the previous section, Formula 1 and Formula 2 rely on the initial scores obtained by cosine similarity. We therefore would be interested to look for better algorithms, so that our retrieval system would not only to make improvement based on the cosine similarity score, but also look for relevant documents that might be missed by it.

In addition we would try on retrieval algorithm other than cosine similarity to test on the effect of our approach to draw evidence from related page. We could then combine various scoring methods to enhance retrieval results further.

Besides, we would like to perform tests to see if the order of modifying scores according to the hierarchy is important. We have already carried out similar tests with Formula 2, in which we split the formula into two runs. In future research, we would examine the order of applying score to ancestors and descendents in each run.

Finally apart from using URLs as hints to categorize pages, we would also like to examine the use of other structure, including linkages among pages within a website, to test the effectiveness of drawing evidence from related pages that utilizes other structure within the website. This might enable us to deal with websites, such as Wikis and CMSs, where the domain structure of the site is not directly reflected in the URL structure.

7 References

Beigbeder, M. (2007): Structured content-only information retrieval using term proximity and propagation of title terms. In Proceedings of INEX 2006, page 200-212.

Brin, S. and Page, L (1998): The anatomy of a large-scale hypertextual web search engine. In Proceedings of the 7th International World Wide Web Conference, pages 107-117.

Callan, J. (1994): Passage-level evidence in document retrieval. In Proceedings of the 17th ACM-SIGIR Conference on Research and Development in information retrieval, pages 302-310.

Carpineto, C., Romano, G., Caracciolo, C. (2007): Information Theoretic Retrieval with structured Queries and Documents. In Proceedings of INEX 2006, pages 178-184.

Glover, E., Tsioutsiouliklis, K., Lawerence, S., Pennock, D. M. and Flake G. W. (2002): Using web structure for classifying and describing web pages. In Proceedings of the 11th international conference on World Wide Web. Pages 562-569.

INEX: Initiative for the Evaluation of XML Retrieval, 2007

Kimelfed, B., Kovacs, E., Sagiv, Y. and Yahav, D. (2007): Using Langauge models and the HITS Algorithm for XML Retrieval, Proceedings of INEX 2006, pages 253-360.

Kleinberg, J (1999): Authoritative sources in a hyperlinked environment. Journal of the ACM, 46:604-632.

Kules B., Kustanowitz J. and Shneiderman, B. (2006): Categorizing web search results into meaningful and stable categories using fast-feature techniques. In Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries.

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. and Gatford, Mike. (1994): Okapi at TREC-3. NIST Special Publication 500-226, Overview of the Third Text Retrieval Conference (TREC-3).

Salton, G. (1971): The SMART Retrieval System – Experiments in Automatic Document Processing, Prentice-Hall, Inc., Upper Saddle River, NJ, 1971.

Shih, L. K. and Karger D. R. (2004): Using URLs and table layout for web classification tasks, In Proceedings of the 13th international conference on World Wide Web, pages 193-202.

Sigurbjörnsson, B., Kamps, J. and Rijke, M. (2004): An Element-Based Approach to XML Retrieval. In INEX 2003 Workshop Proceedings, pages 19-26.

Wilkinson, R. (1994): Effective retrieval of structured documents. In Proceedings of the 17th ACM-SIGIR Conference on Research and Development in information retrieval, pages 311-317.

Van Rijsbergen, C. J., Robertson, S. E. and Porter, M. F. (1980): New models in probabilistic information retrieval. London: British Library. (British Library Research and Development Report, no. 5587).

CSC: Supporting Queries on Compressed Cached XML

Stefan Böttcher, Rita Hartel
University of Paderborn,
Computer Science, Fürstenallee 11,
33102 Paderborn, Germany
+49 52 51 60 66 62, +49 5251 60 66 12
stb@uni-paderborn.de, rst@uni-paderborn.de

Abstract

Whenever a client frequently has to retrieve, to query and to locally transform large parts of a huge XML document that is stored on a remote web information server, data exchange from the server to the client may become a serious bottleneck that simply limits scaling of the amount of information that can be processed locally on the client by a client-based application.

We present Compressed Structure Caching (CSC) as a solution that reduces the amount of data exchange by a combination of the following techniques: compression of the XML document's structure, client-side caching of the structure and of already received XML content, inference and optimized loading of the content needed on the client to answer a given query.

We provide a performance evaluation that demonstrates that our approach significantly reduces the amount of data exchange from server to client.

Keywords: XML, Caching, Compression.

1 Introduction

1.1 Motivation

XML has become a standard data exchange format in many information sources e.g. in the web, and XPath has become a key standard for context sensitive search in huge XML documents.

We consider scenarios, in which client applications need to process large fragments of huge XML documents that are provided on remote web servers, and where the data exchange from the server to the client is a bottleneck. These scenarios require techniques that minimize data transfer between the server-side XML information source and the client submitting queries. In order to reduce the data transfer, two techniques are possible: compression of exchanged data, and caching and reuse of previous query results. Both techniques have been investigated independently of each other, but are challenging to combine.

Our approach combines both techniques, i.e., it is based on caching and reusing compressed XML information which the client has previously downloaded from the server. The problem considered in this paper is how to reduce the amount of XML data exchange between server and client by intelligent server-side XML fragment compression and by a client-side caching and integration strategy for compressed XML data, such that XPath queries can be executed on cached compressed data on the client.

1.2 Limitations of related approaches

Related approaches follow two different directions called query shipping and data shipping. *Query shipping* means that each client C that cannot answer a given query Qi from its cache sends Qi to the information server that executes Qi on the information source IS, i.e. computes R=Qi(IS) and returns the result R to C. The returned result R is stored in the cache and can be used to answer a second query Qk if a compensation query Qc exists such that Qk applied to the information source IS returns the same answer as applying Qc to R. More formally, the returned result R can be used to answer a second query Qk if it can be proved that Qk(IS) = Qc(R). The proof techniques suggested by e.g. (Balmin et al. 2004), (Mandhani and Suciu 2005), (Xu and Ozsoyoglu 2005) aim at showing that Qk is equivalent to Qc o Qi, which is sufficient to prove that Qk(IS) = Qc(Qi(IS)) = Qc(R).

Unfortunately, the compensation query approaches are applicable to very small subsets of XPath only. Even worse, it can be shown that already for very small subsets of XPath the search for compensation queries is NP hard.

As a consequence most queries can not profit from the cache when using query shipping and the same XML fragments may be shipped as a part of an answer again and again.

Data shipping means that all the data needed to answer a query Qi is shipped to the client such that Qi can be answered locally. For example, approaches like (Böttcher and Türling 2004), (Koch, Scherzinger and Schmidt 2008), and (Marian and Siméon 2003) have been developed to compute the so called *read set* of a query, which is an easily computable superset of the data that has to be accessed to answer the query.

For the purpose of reducing data transfer, the data shipping approach has the following advantages: A client can determine very fast which part of the data needed to answer the query can be read from its cache, and in the long run, the data shipping approach to query processing increases the number of possible cache hits. Although data shipping ensures a greater amount of cache-hits than query shipping, a huge amount of data must be transferred for queries the read-set of which is very large, e.g. queries that count data or queries that involve a long search on the XML structure. In other words, the disadvantage of data shipping is an increase of transferred data that is not needed to answer a query.

Regarding the advantages and disadvantages of query shipping and data shipping, our goal is to combine the advantages of both query processing approaches. Although there is a trade-off between additional data transfer and number of cache-hits, we show that our approach results in a significant reduction of overall data transfer between the web-information source and the client.

1.3 Problem description

The problem investigated is how to improve web information caching in such a way that the overall data exchange needed between the web information source and the client is reduced for arbitrary XPath queries. This includes XPath queries which have to search or to navigate in a large part of the structure of an XML document and XPath queries that use axes beyond the limited sub-classes that have been investigated for compensation queries.

1.4 Contributions

This paper proposes a novel approach to caching of XML web information, called Compressed Structure Caching (CSC) that combines the following properties:

1. It separates a huge XML document that is provided on a web information server into its constituent parts: its tree structure and its values of text constants and of attributes.
2. It extracts and compresses the structure of the XML document to a compressed tree (CT) including the element names and attribute names and transfers the CT to the web clients that want to work on the XML document.
3. The client can decide based on the CT whether or not he has enough knowledge to answer the query on its own, i.e., without sending the query to the server and retrieving the query results from the server.
4. Web clients submit queries to the server which infers from each client query which text and attribute values are required on the client to answer the query. The required values are compressed and sent to the server.
5. Finally, the client caches the CT plus the compressed values and evaluates its queries on these compressed structures.

We have implemented and comparatively evaluated our system (CSC) with two other approaches, i.e. querying uncompressed data, and querying compressed data that is not kept in the cache. Our results show that SCS clearly outperforms the two other approaches.

1.5 Paper organization

The remainder of this paper is organized as follows. In Section 2, we explain the key ideas of the general solution and show the system architecture. Section 3 describes a specific solution instance for which we have done the performance evaluation and the evaluation results. Section 4 describes related work and Section 5 contains a summary and the conclusions.

2 Key ideas of the CSC solution

We first give an overview of the system architecture and thereafter describe the key ideas and design decisions of our implementation of CSC.

2.1 System architecture

The overall architecture of our CSC system is shown in Figure 1.

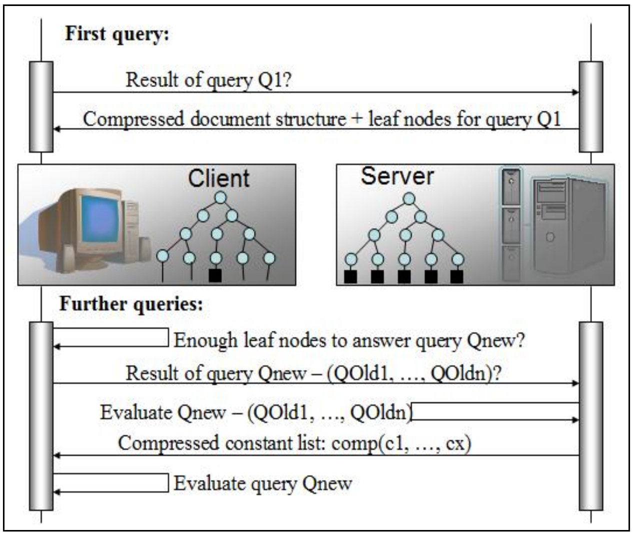

First query:
Result of query Q1?
Compressed document structure + leaf nodes for query Q1
Client
Server
Further queries:
Enough leaf nodes to answer query Qnew?
Result of query Qnew – (QOld1, …, QOldn)?
Evaluate Qnew – (QOld1, …, QOldn)
Compressed constant list: comp(c1, …, cx)
Evaluate query Qnew

Figure 1. Overview of the system architecture

The server that provides the huge XML information source separates the XML tree structure from the values of the XML leaf nodes, i.e. texts and attributes, and compresses the XML structure without the XML leaf node values separately.

When a client submits its first query to the server, the server transfers the highly compressed XML structure to the client. Additionally, for each query, the server collects all the XML document's leaf nodes in document order that are needed for query evaluation, but that are not contained in the client's cache. The server compresses this XML leaf node constant list and sends it to the client. Finally, the client evaluates the query.

Before the client sends further queries to the server, it checks whether the query can be answered locally or requires more XML leaf node data. Only if data is missing, in the client's cache, the client submits the query to the server.

2.2 Key idea of CSC

The key idea of the implementation is that client and server use the same XPath evaluator EV with one exception: the implementation of the access to constant values. The main requirement to EV is that the program code accessing constant values is isolated, e.g. it is done in a function *getValue(XPathQuery, currentContextNode)* for which a server implementation exists that differs from the client implementation. Client implementation and server implementation communicate via the constant exchange buffer explained below. All other operations of the XPath evaluation EV are implemented only once, and are used in an identical fashion on the server side and on the client side.

Client and server use the same XPath evaluator, in our case an evaluator based on a reduced instruction set RIS. RIS consists of the following operations:

- fc: Returns the first-child of the current context node ccn.
- ns: Returns the next-sibling of the current context node ccn.
- label: Returns the label of the current context node ccn if ccn is an element or an attribute node.
- parent: Returns the parent of the current context node ccn.
- node type: Returns the node type (i.e., either element, attribute, or text node) of the current context node ccn.
- getValue: Returns the text value of the current context node ccn if ccn represents a text node or the text value of an attribute.

Other operations of CoreXPath are reduced to these operations using rewrite rules as presented in (Böttcher and Steinmetz 2007a).

The main goal of the constant exchange is to send only those constants from the server to the client that are really needed. This will save most of the data exchange, because the constants are much more difficult to compress than the structure.

The key idea is to use the same XPath evaluator on the server and on the client side. The server's evaluator only picks those constants from the document that are needed on the client side. These constants are packed together, are compressed using string compression (e.g. in our case bzip2), and finally are submitted to the client and decompressed. As the evaluation order of XML nodes on the client side and on the server side are identical, the constant order picked by the server is identical to the constant order required by the client. Therefore, the server simply writes picked constants sequentially into the buffer that is compressed, submitted to the client and decompressed at the client side. And the client simply reads the constants from the decompressed buffer one by one.

2.3 Separation of structure and text constants

Due to the semi-structured nature of an XML document, the document structure contains a lot of redundancies, while the text data does contain fewer redundancies. Therefore, the document structure alone can be compressed much better than text and attribute values alone or the XML document as a whole combining structure and constants. For example, the overall compression ratio achieved for different XML documents including structure plus text data reaches compression ratios of 10% up to 30%, i.e., it reduces the document size by a factor of 3.3 up to 10. However, the compression ratio achieved for structure only of the same XML documents is between 0.3% and 10%, i.e., it reduces the document structure size by a factor of 10 up to 330. Furthermore, nearly all XPath queries access significantly more inner XML document nodes which are part of the XML structure than they access leaf nodes which contain the text or attribute values.

Therefore, we propose to separate the structure from the text constants and compress both parts separately. Compression approaches that perform such a separation of structure and constants and that support queries and even modification on the generated compressed XML at the same time are e.g., BSBC (Böttcher, Hartel and Heinzemann 2008) and DTD subtraction (Böttcher, Steinmetz and Klein 2007). In general, any compressor that separates structure from text constants and that support queries and modification on the generated compressed XML can be integrated in our approach to Compressed Structure Caching (CSC) as well.

2.4 Caching the complete structure, but constants only on demand

Due to the strong compression achievable for structure, it is much more likely that the complete structure can be kept in the cache than that the complete XML document or huge fragments of it can be stored in cache. Therefore, one of the key ideas of CSC is to keep the complete compressed document structure within the client's cache.

However, the constants are loaded into the client's cache on demand, i.e., only when they are needed in order to answer a query. The benefit of this idea is that the inner XML document nodes, i.e. the huge majority of nodes needed for query processing is already available in the cache, whereas only a small minority of nodes, i.e. the leaf nodes really accessed, have to be loaded if not already present in cache.

As the complete document structure and a subset of the text constants are known on the client, the XPath evaluation can be started at the client side in order to determine, whether or not text constants or attribute values are missing. That means that the client can decide based on its XML cache, whether or not it contains already all the data needed to answer the queries.

In contrast, other approaches like the proof techniques suggested by e.g. (Balmin et al. 2004), (Mandhani and Suciu 2005), (Xu and Ozsoyoglu 2005) aim at showing that a so called compensation query applied to the cache returns the same results as the original query applied to the server's database.

Unfortunately, the compensation query approaches are applicable to very small subsets of XPath only and even worse, it can be shown that already for very small subsets of XPath the search for compensation queries is NP hard.

2.5 Pointer-less identification of constants

Pointers from the XML structure into a compressed text or attribute value constant buffer may speed-up query processing, but the addition of pointers to the compressed structure of an XML document will significantly blowup size of the compressed structure, usually by more than 100%. In comparison, a pointer-less technique to address the relevant constants significantly reduces the space needed for the compressed XML structure.

Therefore, CSC uses a constant identification technique that avoids the need for pointers from the compressed structure to the compressed constant values. This constant identification technique saves cache space to store larger XML structures and thus reduces the data transfer from the server to the client.

The key idea used by CSC is the following.

As the XPath evaluation can be started on the document structure, the evaluation order gives an implicit order of the needed text constants. When the server uses the same evaluation order as the client, the server can send compressed text constants in this evaluation order, and the client receives the constant in the evaluation order needed. Therefore no explicit pointers or identification of the constants is needed - neither to transfer constants, nor to insert the constants in the cache, nor to use the constants on the client side. This pointer-less access to the needed constants results in a significant reduction of transferred data.

Note that other caching techniques, e.g. (Böttcher and Türling 2004) use an additional addressing or identification schema for XML nodes, as e.g. the ORDPATH numbering scheme (O'Neil et al. 2004), in order to integrate additional data from the server into the client's cache. This address information requires cache memory and has to be transferred, both being avoided by our CSC approach.

2.6 Constant identification on the server side

On the server side, a modified XPath evaluator not only evaluates the query, but simultaneously collects all the constants of accessed XML leaf nodes in evaluation order, which in our special case is XML document order. Form this collection of accessed XML leaf node constants, all the constants that are already known to the client's cache are deleted during the same scan through the XML document by a technique described in Section 2.7. Thereby, the result of this server side query evaluation is a list of constant values of accessed XML leaf elements in evaluation order, except the values of those leaf elements that are already stored in the client's cache. This list of constants is transferred to the client without any additional identification information, as this information is implicitly known to the client because of the document structure and the evaluation order.

2.7 Constant usage on the client side

The server and the client have to agree on a common XPath evaluator to ensure that the evaluation order of constants is the same on server and on client.

Before sending the query to the client, the client tries to evaluate the query on its cache. During evaluation it will either realize, that no constant data is missing. In this case, no data has to be transferred between server and client. If on the other hand, the client realizes, that its cache does not store all the leaf node values needed to answer a query independently of the server, the client sends the XPath query string to the server. There it is evaluated by a modified XPath evaluator that collects only those values of texts and attributes that are accessed, but not contained in the client's cache in order to answer the query. The values collected by the XPath evaluator are compressed and sent to the client. After having received the list of constants, the client can continue the query evaluation. Whenever a constant value is missing, the client consumes the next constant value from the list, stores it on the current position within the compressed document and uses it for client evaluation. As client and server have agreed on the same evaluation order, this ensures, that the next constant value received from the server is the next constant value the client needs.

This allows a compression and decompression technique of string constants where the server simply pipes in strings into the compressed stream to the client and the client simply extracts them one by one.

2.8 How to avoid the transferal of leaf nodes that are stored in the client's cache

In order to prevent the server from sending constants to the client that are already stored in the client's cache, either the server has to know, which constants are stored within the client's cache, or the client has to tell the server which constants are still stored in the cache.

To avoid sending the same XML leaf constants several times, we have adopted and slightly modified an approach of (Böttcher and Türling 2004). Each query string submitted from the client to the server is stored in a client-specific query list on the server. The server compares this query list with the IDs of old queries to compute the list of *cached queries*, i.e. those queries, the results of which are still stored in the client's cache. The list of cached queries and the actual query are combined to compute the missing constants. Here *missing constants* denote the constants needed to answer the actual query on the client-side that are not yet contained in the cache, i.e. which are the cache-misses and have to be sent from the server to the client. To avoid reading the XML source multiple times, we use a streaming-based approach to read the whole XML document in a single pass only based on (Olteanu et al. 2002) and (Böttcher and Steinmetz 2007a), and apply multiple queries, i.e. the actual query and the cached queries in parallel on this stream. This provides an easy way to compute the leaf nodes accessed by the actual queries and the leaf nodes accessed by the cached previous queries in a single run on the compressed XML file. Only those leaf nodes that are not yet present in the client's cache are collected in document order, are compressed, and are then sent to the client.

2.9 How the client embeds the received constants

CSC has to adapt its pointer-less identification of constants to the situation that some, but not all constants are already present in the client's cache. This means that for each leaf node visited by the client's XPath evaluator, the evaluator has to know whether the constant is stored in the cache or is contained in the stream of constants retrieved from the server.

This can be achieved by using one bit for each leaf node in the cached structure telling the XPath evaluator whether the constant has to be read from the client's cache or from the stream of constants provided by the server.

3 Performance evaluation

Our performance evaluation was done with BSBC (Böttcher, Hartel and Heinzemann 2008) as structure compression tool and Bzip2 as text compressor.

3.1 Summary of BSBC

In our performance evaluation, we have used BSBC on both, the client and the server, as the compressor that generates queryable and updateable compressed XML data and as the tool that performs queries on BSBC compressed XML data.

BSBC is an XML compressor based on element name encoding on the one hand, and on sharing of common sub-trees on the other hand. The BSBC XML compressor separates the XML constants from the XML element names and attribute names and from the nesting of start tags and end tags, i.e. the compressed document structure of an XML document consists of the following parts:

i. A bit stream representing the tree structure of the element nesting in the XML tree, without storing any label information. In the bit-stream, each start-tag is represented by a '1'-bit and each end-tag is represented by a '0'-bit.

ii. Inverted element lists, containing a mapping of element and attribute names to '1'-bit positions within the bit stream.

iii. A so called DAG pointer list. The DAG pointer list represents the shared sub-trees, and it consists of a list of pointers from a parent element to the repeated sub-tree occurring previously within the document structure.

In addition to the document structure, BSBC stores the constants, i.e., the text and attribute values in form of separate constant containers based on the parent element name and compresses each constant container using the generic compressor BZip2.

3.2 Performance evaluation environment

We have implemented Compressed Structure Caching using Java 1.5. We have evaluated Compressed Structure Caching on a dataset created by the XMark Benchmark (Schmidt et al. 2002) using creation factors from 0.0001 to 0.1 and yielding document sizes from 34 kB (factor 0.0001) up to 11.3 MB (factor 0.1).

We compared three different models

- CSC: Compressed Structure Caching described in this paper that initially transfers the whole structure compressed by BSBC. For each query, only the missing constants of XML leaf nodes are transferred, and they are transferred in a compressed way.
- Compression: The query results are computed on the server and the results are only compressed by BSBC before transferring them to the client, i.e. caching is not used.
- Direct: The query results are computed on the server and are transferred to the client in an uncompressed way.

3.3 Performance results

In order to compare the three models, we have executed the queries shown in Table 1 sequentially and have measured the total data volume transferred from server to the client.

ID	Query
q0	/site/people/person[phone or homepage]/name
q1	/site/people/person[descendant::watches]
q2	/site/regions/europe/item/name
q3	/site/people/person/address/city
q4	/site/open_auctions/open_auction/bidder
q5	/site/closed_auctions
q6	/site/people
q7	/site/open_auctions
q8	/site/categories
q9	/site
q10	/site/regions/europe

Table 1. Queries used to evaluate the proposed approach

Figures 2 and 3 show the transferred data volume for the documents created with factors 0.0001 and 0.1. A comparison of both figures demonstrates that after a certain amount of queries, Compressed Structure Caching (CSC) transfers less data than the compressed model that transfers itself less data than the direct model. The bigger the document on the server is, the earlier this effect can be realized: For the document XMark 0.0001, the structure cache transfers less data than the compressed model only from query q9 on, whereas it transfers less data from query q6 up for document XMark 0.1.

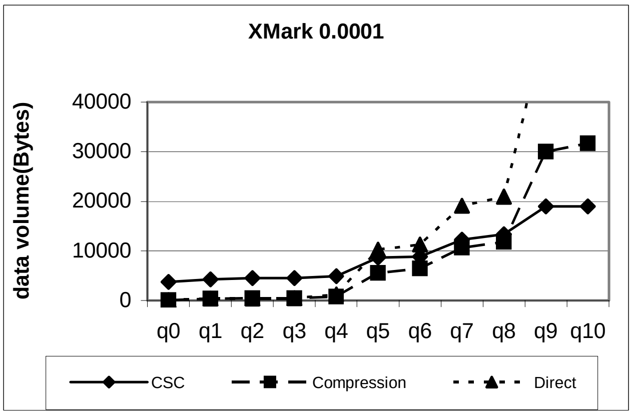


Figure 2. Transferred data volume for XMark 0.0001

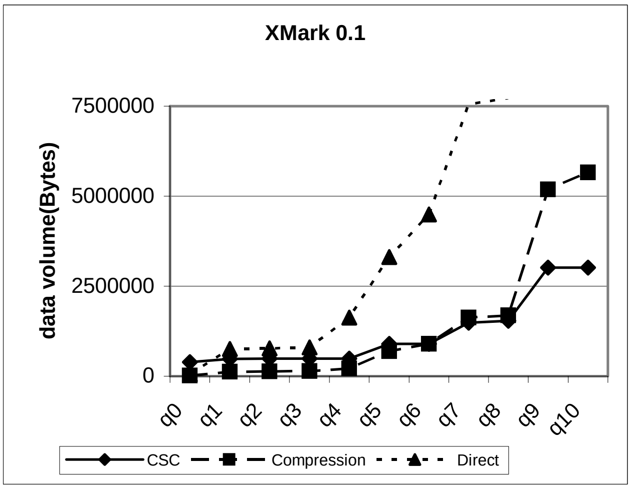


Figure 3. Transferred data volume for XMark 0.1

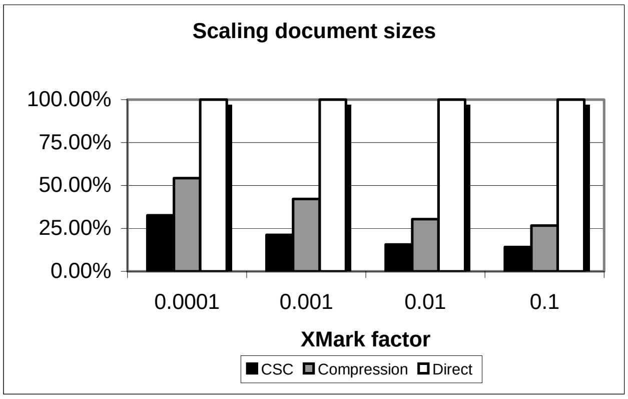


Figure 4. Transferred data volume for different document sizes

This is mainly due to the relative size of the compressed structure which is initially transferred from server to client, and which is 10% of the document size for XMark 0.0001, but only 3% of the document size for XMark 0.1.

We can take a closer look on this effect in Figure 4: When using the compressed model without caching, the data volume that has to be exchanged can be reduced to the size of 55% to 26%, compared to the data volume exchanged with the Direct strategy using neither caching nor compression. The reduction of the exchanged data volume is better for larger XML documents.

When using the compressed model with caching, the data volume that has to be exchanged can be reduced even significantly better, i.e. to the size of 33% to 14%, compared to the data volume exchanged with the Direct strategy using neither caching nor compression. Again, the reduction of the exchanged data volume is better for larger XML documents.

To summarize, it can be seen, that even for relative small document sizes and few queries, by using Compressed Structure Caching, the data volume can be reduced up to 13% compared to the direct model and up to 50% compared to using the compressed model.

4 Related work

Although both, web data caching and XML compression, contribute to a reduction of the data transfer from server to client, the fields of web data caching and XML compression have mostly been investigated independently of each other.

There has been a lot of work in XML compression, some of which does not support query processing on compressed data, e.g. (Liefke and Suciu 2000), and most of which support querying compressed data, but not querying cached compressed data, e.g. (Buneman, Grohe and Koch 2003), (Busatto, Lohrey and Maneth 2005), (Cheng and Ng 2004), (Ng et al. 2006), (Zhang, Kacholia and Özsu 2004).

Contributions to the field of caching range from concepts of querying and maintaining incomplete data, e.g. (Abiteboul, Segourin and Vianu 2001), over caching strategies for mobile web clients, e.g. (Böttcher and Türling 2004), to caching strategies based on frequently accessed tree patterns, e.g. (Yang, Lee and Hsu 2003). In comparison, our approach allows for XPath queries using filters and comparisons with constants even on compressed cached XML.

Different approaches have been suggested for checking whether an XML cache can be used for answering an XPath query. On the one hand, there are contributions, e.g. (Balmin et al.2004), (Mandhani and Suciu 2005), (Xu and Ozsoyoglu 2005), that propose to compute a compensation query. These approaches can also be used on compressed XML data, but they are NP-hard already for very small sub-classes of XPath. On the other hand, containment tests and intersection tests for tree pattern queries have been proposed, and could in principle be used for deciding whether a given XPath query can be executed on the cached data locally. However, such intersection tests and containment test are NP-hard for rather small subsets of XPath expressions (Benedikt, Fan and Geerts 2005), (Hidders 2003). In comparison, our approach uses a fast difference computation that can be done within a single scan through the compressed XML file.

In comparison to all other approaches, our technique is to the best of our knowledge the only strategy that combines the following advantages: it caches the relatively small compressed XML structure and supports XPath queries on it, transfers only constants that are really needed for query evaluation and uses a pointer-less transfer format, and it uses an intelligent server strategy to identify those leaf nodes not yet stored in the client's cache.

5 Summary and Conclusions

Whenever data exchange with XML-based information sources is a bottleneck, it is important to reduce the amount of exchanged XML data. Our CSC approach combines two reduction techniques for exchanged data, i.e. caching and XML compression. Furthermore, CSC supports XPath query evaluation on cached compressed XML data, and is not limited to a small XPath subset like tree pattern queries. Additionally, CSC takes advantage of caching the small compressed XML structure and provides an intelligent technique for transferring only those XML leaf node constants from the XML information server to the client that are really needed for query evaluation and that are not yet stored in the client's cache.

Finally, we have provided a performance evaluation that shows that a significant reduction in data exchange can be achieved by CSC.

An interesting extension of our research is to consider modification of compressed data on the client side or the server side. Modification of compressed XML data without complete decompression has been solved in (Böttcher and Steinmetz 2007b), and it seems to be promising to apply this to our compressed XML cache. Furthermore, cache updates and consistency between an XML server and an XML cache have been solved for uncompressed XML (Böttcher 2006).

Therefore, we consider it a promising challenge to combine all three aspects caching, compression and consistent updates in future work.

As XPath is used in other important XML standards like XSLT and XQuery, we consider it a challenging research topic to enhance the results presented here in such a way that they are applicable to XQuery or XSLT as well.

6 References

Abiteboul, S., Segourin, L. and Vianu, V. (2001): Representing and querying XML with incomplete information. *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems,* Santa Barbara, California, USA, ACM Press.

Balmin, A., Özcan, F., Beyer, K.S., Cochrane, R. and Pirahesh, H. (2004): A Framework for Using Materialized XPath Views in XML Query Processing. *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases*, Toronto, Canada, 60-71, Morgan Kaufmann.

Benedikt, M., Fan, W. and Geerts, F. (2005): XPath satisfiability in the presence of DTDs. *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems,* New York, NY, USA, 25-36, ACM Press.

Böttcher, S., Hartel, R. and Heinzemann, C. (2008): BSBC: Towards a succinct data format for XML streams. *Proceedings of the Fourth International Conference on Web Information Systems and Technologies*, Funchal, Portugal, 13-21, INSTICC Press.

Böttcher, S. and Steinmetz, R. (2007a): Evaluating XPath Queries on XML Data Streams. *Data Management. Data, Data Everywhere, 24th British National Conference on Databases,* Glasgow, UK, 101-113, Springer.

Böttcher, S. and Steinmetz R. (2007b): Data Management for Mobile Ajax Web 2.0 Applications. *Database and Expert Systems Applications, 18th International Conference, DEXA 2007,* Regensburg, Germany, 424-433, Springer.

Böttcher, S., Steinmetz, R. and Klein, N. (2007): XML index compression by DTD subtraction. *Proceedings of the Ninth International Conference on Enterprise Information Systems*, Funchal, Madeira, Portugal, 86-94.

Böttcher, S (2006): Cache Consistency in Mobile XML Databases. *Advances in Web-Age Information Management*, Hong Kong, China, 300-312, Springer.

Böttcher, S. and Türling, A. (2004): Caching XML Data on Mobile Web Clients. *Proceedings of the International Conference on Internet Computing, IC '04*, Las Vegas, Nevada, USA, 150-156, CSREA Press.

Buneman, P., Grohe, M. and Koch, C. (2003): Path Queries on Compressed XML. *Proceedings of 29th International Conference on Very Large Data Bases*, Berlin, Germany, 141-152, Morgan Kaufmann.

Busatto, G., Lohrey, M. and Maneth, S. (2005): Efficient Memory Representation of XML Dokuments, *Database Programming Languages, 10th International Symposium,* Trondheim, Norway, 199-216, Springer.

Cheney, J. (2001): Compressing XML with multiplexed hierarchical models. *Proceedings of the 2001 IEEE Data Compression Conference (DCC 2001),* Snowbird, Utah, USA, 163-172, IEEE Computer Society.

Cheng, J. and Ng, W. (2004): XQzip: Querying Compressed XML Using Structural Indexing. *Advances in Database Technology - EDBT 2004, 9th International Conference on Extending Database Technology,* Heraklion, Crete, Greece, 219-236, Springer.

Hidders, J. (2003): Satisfiability of XPath expressions. *Database Programming Languages, 9th International Workshop, DBPL 2003*, Potsdam, Germany, 21-36, Springer.

Koch, C., Scherzinger, S. and Schmidt M. (2008): XML Prefiltering as a String Matching Problem. *Proceedings of the 24th International Conference on Data Engineering,* Cancun, Mexico, 626-635, IEEE.

Liefke, H. and Suciu, D. (2000): XMill: An Efficient Compressor for XML Data, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, Dallas, Texas, USA, 153-164, ACM.

Ng, W., Lam, W.Y., Wood, P.T. and Levene M. (2006): XCQ: A quer14ziable XML compression system. *Knowledge and Information Systems*, **10**(4):421-452.

Mandhani, B. and Suciu, D. (2005). Query caching and view selection for XML databases. *Proceedings of the 31st international conference on Very large data bases*, Trondheim, Norway, 469-480, ACM.

Marian, A. and Siméon, J. (2003): Projecting XML Documents. *Proceedings of 29th International Conference on Very Large Data Bases,* Berlin, Germany, 213-224, Morgan Kaufmann.

Olteanu, D., Meuss, H., Furche, T. and Bry, F. (2002): XPath: Looking Forward. *XML-Based Data Management and Multimedia Engineering – EDBT*

2002 Workshops, Prague, Czech Republic, 109-127, Springer.

O'Neil, P.E., O'Neil, E.J., Pal, S., Cseri, I., Schaller, G. and Westbury, N.(2004): ORDPATHs: Insert-Friendly XML Node Labels. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Paris, France, 903-908, ACM.

Schmidt, A., Waas, F., Kersten, M., Carey, M., Manolescu, I. and Busse, R. (2002): XMark: A benchmark for XML data management. *Proceedings of 28th International Conference on Very Large Data Bases*, Hong Kong, China, 974-985, Morgan Kaufmann.

Xu, W. and Ozsoyoglu, Z.M. (2005): Rewriting XPath queries using materialized views. *Proceedings of the 31st international conference on Very large data bases,* Trondheim, Norway, 121-132, ACM.

Yang, L.H., Lee, M.-L. and Hsu, W. (2003): Efficient mining of XML query patterns for caching. *Proceedings of 29th International Conference on Very Large Data Bases*, Berlin, Germany, 69-80, Morgan Kaufmann.

Zhang, N., Kacholia, V. and Özsu, M.T. (2004): A Succinct Physical Storage Scheme for Efficient Evaluation of Path Queries in XML. *Proceedings of the 20th International Conference on Data Engineering*, Boston, MA, USA, 54-65, IEEE Computer Society.

Ranking-Constrained Keyword Sequence Extraction from Web Documents

Dingyi Chen[1] **Xue Li**[1] **Jing Liu**[1,2] **Xia Chen**[1]

[1] School of Information Technology and Electrical Engineering
The University of Queensland,
Brisbane, Qld 4072, Australia,
Email: xueli@itee.uq.edu.au

[2] School of Computer Science and Electronic Engineering
Xidian University,
Xi'an, 710071, China,
Email: neouma@163.com

Abstract

Given a large volume of Web documents, we consider problem of finding the shortest keyword sequences for each of the documents such that a keyword sequence can be rendered to a given search engine, then the corresponding Web document can be identified and is ranked at the first place within the results. We call this system as an Inverse Search Engine (ISE). Whenever a shortest keyword sequence is found for a given Web document, the corresponding document can be returned as the first document by the given search engine. The resulting keyword sequence is search-engine dependent. The ISE therefore can be used as a tool to manage Web content in terms of the extracted shortest keyword sequences. In this way, a traditional keyword extraction process is constrained by the document ranking method adopted by a search engine. The significance is that the whole Web-searchable documents on the World Wide Web can then be partitioned according to their keyword phrases. This paper discusses the design and implementation of the proposed ISE. Four evaluation measures are proposed and are used to show the effectiveness and efficiency of our approach. The experiment results set up a test benchmark for further researches.

1 Introduction

Search engine eg., Google, Yahoo, or Live Search, helps user find Web pages on a given subject using keywords. Knowing the right keywords, a user is able to locate relevant Web resources in a short time. However, as Kleinberg points out [1], search engines are not able to provide direct answers if user only knows what he/she wants, but does not know the right keywords to search. This would raise an interesting question: How can we extract a sequence of keywords from a Web document, so that once user knows this keyword sequence, he/she would be able to locate the document immediately? Thus the Web search problem could become a problem of approximating or mapping the keywords specified by user into the keyword sequences that can represent documents uniquely. This kind of ranking-constrained keyword extraction process is termed inverse search.

Inverse search as a common psychological process can be used for users to remember the Web pages that they have visited. For example, a tourist might wish to find the best place to watch insects that emit lights at night in Australia. Neither knowing proper terms of the insects nor the place where those insects inhabit, he/she tried "firefly Australia", but none of the search results is about the insects, because this kind of insects is usually referred as "glowworm" or more specifically, *Arachnocampa.* After many trials, he/she finally learnt that glowworm can be seen in Springbrook National Park near Gold Coast and would want to use a few words to represent these relevant pages. Indeed, the exact URLs of these pages can be recorded as bookmarks in a Web browser. However, long URL addresses are generally difficult to memorise or to speak out, therefore they are not suitable for oral communications. Instead of memorising the whole URL, the tourist can just refer to the keywords "glowworm" and "Springbrook" in case his/her friends are also interested. To an on-line advertiser, this Web site could be uniquely identified and advertised by buying these two words (or the word *Arachnocampa*) from the search engine. So when user searches for these words, the associated Web page will be returned at the first place in the search results. On the other hand, a Web page may be uniquely identified by extracting the features of its content and making an index on it.

An *Inverse Search Engine (ISE)* accepts a Web document as input and returns a shortest sequence of keywords that can be used to uniquely identify this document through a search engine. That is, after querying on the keyword sequence, the given Web document should be returned as the first search result by the search engine. In the rest of this paper, the term *target page* refers to the given input Web document, and the *shortest keyword sequence (shortest KS)* refers to a minimum (in keyword counting) ordered-list of terms that can make the target page ranked top in the search results. We use the terms of 'Web document' and 'Web page' interchangeably when the discussion is focused on their content.

For a keyword sequence being shortest, we define the following three characteristics:

- **Minimum number of words in sequence** — A minimum number of keywords that are extracted from a web document. The words are ordered and used as a query on the Web.

This project is supported by Australian ARC Discovery Project DP0558879.

- **Uniquely identifying the document on the Web** — When a search engine uses this keyword sequence, it will locate it uniquely and rank it as the first one in the search result.
- **Search-engine dependent** — Since different search engines may have different document ranking methods, for a given Web document, we are interested in finding the keyword sequence that can make the given document be ranked at the top of the search results by a given search engine. So, different search engines may have different keyword sequences used as queries to make the given document being ranked top.

It should be pointed out that although Page Rank [2] affects the ranking of Web pages of keyword-based search results, our ISE as a content-based approach is considered to be independent from the URLs that link the given Web page to others.

Typical search engines treat a search query as a sequence of keywords. This is because the order of search keywords may indicate either the relevant importance of keywords or the occurring order of keywords. For example, a query of "search engine" is different from "engine search", for the former is looking for an information system to obtain lists of references matched with specific criteria; while the latter might be finding a Web site that compares mechanical engines.

Keyword sequence is useful in many fields. In scientific publishing, authors are required to provide a list of words which point out the main topics of the paper for searching and indexing purposes. The important topic usually appears first in a keyword sequence.

Automatic keyword extraction from documents has been implemented in a few systems. For example, Microsoft Word can generate keywords from given documents. The extraction techniques suggested by [11, 10] can induce keyword-generating rules from the existing document/keyword pairs. Indeed, the keywords generated by those methods reflect the content of target documents to some extent. However, those extracted keywords cannot be used in a ranking process that can bring a Web-deployed document to the top position in a search. This, however, could be the most desired feature to Web surfers. On the other hand, most of those keyword-extraction methods are based on the supervised learning that prefers to work with a high-quality training set, which in many circumstances, is not available.

The main idea of this paper is to combine the keyword extraction with a document ranking process. In order to test the keyword sequence extracted by the system, a search engine will be used to feed with the extracted keyword sequence and to obtain a list of Web pages that the given Web page is included and ranked high. To this end, the implementation of ISE faces a twofold-problem. Firstly, many search engines limit their total number of daily accesses for an automatic client application. Yahoo allows 5,000 queries per day, while Google and Live Search (successor of MSN Search) only allow 1,000. So the number of queries made by ISE should be as small as possible. Secondly, search engines also limit the character-length of queries for security and performance reasons. For example, Google allows at most 2048 characters and MSN allows only 250[1]. Thus, the ISE-generated keyword sequence should contain as fewer words as possible.

To find out the shortest KS, there is always a naive way by using a brute-force method to exhaustively search for the solution. However, this will be computationally very expensive and infeasible. We propose a heuristic method to discover the shortest KS. Our method consists of three stages: embracing, expanding, and eliminating. Firstly the target page is embraced by a seed, i.e., an initial candidate KS that ranks target page in a work range such as top 100, or a larger number to be decided experimentally. Then the candidate KS is extended in order to improve the ranking of the page in that search engine. Finally, the terms in the candidate KS is reordered and surplus terms are eliminated. In this case, the result of ISE is search engine dependent, that is, the shortest KS that tops the target page in Google does not necessarily top the target page in Yahoo.

We have developed a framework of four measures for the evaluation of the effectiveness and efficiency of ISE. These measures are: (1) the *success rate* that is a count of the shortest KSs that can be obtained from different Web pages; (2) the *top-one rate* which tells the percentage of the obtained shortest KSs that actually top the target pages; (3) the *shortness* that reflects whether the ISE is KS-size efficient; and (4) the *impoliteness* that indicates whether the ISE sends too many queries to bother a search engine.

Keyword sequence extracted by ISE has three advantages. (1) Keyword sequence can be used as a digest of corresponding Web content. It can be a phrase that captures the topic of document. It can also be used as a query to get the target page from the Web. In this case, the best representative keywords become the best query words. (2) The training to the keyword extraction process is now performed by a search engine that provides feedback through its ranking process. So ISE does not need to collect a large volume of training data set for keyword extraction. (3) Keyword sequence as a shorthand of Web document can be used to index the Web content so to improve the search engine efficiency. It can be used by online advertisement or other Web-based applications where the key phrases are uniquely associated with certain services or functions.

This paper is organised as follows. Section 2 addresses the influential related work. Section 3 explains our proposed approach. Section 4 illustrates our experimental results. Section 5 provides the conclusions.

2 Related Work

There are two types of keyword-extraction approaches: (1) domain-dependent methods that are based on the supervised machine-learning models and require large training corpora, and (2) domain-independent methods that do not require training corpora.

A keyword is a meaningful term that has some importance in a document. It can be identified using the term frequency (TF) [8]. The intuition is that the important concepts are likely to be referred to more times than others. However, this might not be true in the situation that the terms are frequent in all documents that have the similar content. In this case, the documents cannot be differentiated from each other. As an alternative, we can rank the candidate keywords based on the inverse document frequency (IDF) [9].

Frank *et al.*, [3] introduced an automatic keyword extraction algorithm namely KEA, based on a domain-specific machine learning model. It employs lexical and information retrieval methods to identify candidate keyphrase from document. It calculates the feature values for each candidate and uses Naive Bayes machine to predict the overall probability of

[1] From the HTML source code of http://www.google.com and http://www.msn.com

keyphrases. The features used in the algorithm include $TF \times IDF$ and the positions of their first occurrences.

Kelleher *et al.*, [5] enhanced the KEA by introducing a feature of "Semantic Ratio (SR)" which makes KEA adapted to Web corpus. The SR of a phrase is calculated by dividing the number of occurrences of the phrase in the current document by the number of times it occurs in all documents directly hyperlinked to that document. The idea is based on the assumption that the semantics connection between Web documents is measurable by counting the neighbours of a Web document (in a way similar to the Page Rank) and that the subject matter (identified by the keyphrases) of the document is therefore in some way related to their content. They concluded that the hyperlink information can be used to improve the effectiveness of automatic keyphrase extraction by 50%.

Yih *et al.*, [10] demonstrated that by using extra features, such as the $TF \times IDF$ vectors, the meta data in Web pages, and the query-frequency information from Microsoft MSN query logs, their learning algorithm can substantially outperform KEA.

In the context of extracting keywords from Web documents, it is impossible to collect a large enough training data set for all possible types of Web content. So the domain-independent keyword extraction which does not require training corpus is considered in our approach.

Matsu and Ishizuka [6] proposed a keyword extraction algorithm from a single document without using any training corpus. The method firstly extracts a set of frequent terms from the given document. Then a set of co-occurrences between a term and a set of frequent terms is generated. If the probability distribution of a co-occurrence between term t and a set of frequent terms is biased to a particular subset of frequent terms, then term t is believed to be a keyword. In this case, the degree of bias of the co-occurrence between a term and a set of frequent terms is regarded as an indicator of the term importance.

Our work shares a similar idea from the Implicit Query System [4] and the Robust Hyperlinks [7]. The Implicit Query System can automatically generate query words or phrases from an email and send them to an Internet search engine in order to find documents that are relevant to the email. Their method can extract the special features from emails, such as the words used in subject line. The system also uses query logs from the Microsoft MSN Search to avoid picking up the words or phrases that would never be queried by real-life users. In this way, the system can dramatically reduce the total number of candidate queries. Their method uses a training data set with a logistic regression training process.

The Robust Hyperlinks is another keyword extraction method that extracts text signatures from Web pages. These signatures serve as search queries to locate Web pages once the hyperlinks fail. In [7], text signatures are acquired through the TF-IDF vectors. This approach can be very effective for Web directory maintenance and digital libraries, because the total number of documents is known in advance. However, it is difficult, if not impossible, to provide a sufficient large collection of Web pages to produce IDF based on the unlimited number of Web documents.

In our work, the keyword learning is not based on a large collection of documents, but on the feedback given by a search engine in its ranking process. For a given keyword sequence extracted from a document, the higher ranking of the document has, the better that the keyword sequence represents the document. The system can learn directly from the feedback supplied by search engine to find out the best query words or phrases for a given Web page without training on a corpus. The next section discusses the design of Inverse Search Engine (ISE).

3 Inverse Search Engine (ISE)

A query rendered to a search engine is in the form of a text string (keyword sequence) k. A sorted list of Web documents D will be returned as the search result of k. The relationship between search result list D and the search engine function SE can be expressed as:

$$D = [d_1, d_2, \ldots d_i, d_n] \leftarrow SE(k) \tag{1}$$

where the i-th result is referred as d_i and n is the maximum search range, which defines the maximum number of returned Web documents.

In contrast to SE , an inverse search engine (ISE) accepts a target page $\check{d}$ as input and returns the shortest keyword sequence $\check{k}$ which makes the target page $\check{d}$ be ranked at the top of the results of a search engine. In other words, the problem is: Given a target document $\check{d}$ and a search engine $SE(k)$, find a *target keyword sequence* $\check{k}$, which is the shortest keyword sequence that makes the target document $\check{d}$ be ranked at the first ($d_1 = \check{d}$) of the search results, as shown in Formula 2:

$$\begin{array}{ll} \check{k} \leftarrow ISE(\check{d}), & \text{such that} \\ Rank(\check{k}, \check{d}) = 1 & \end{array} \tag{2}$$

where

$$Rank(k,d) = \begin{cases} i & \text{if } d \in D,\, d_i = d \\ -1 & \text{if } d \notin D. \end{cases} \tag{3}$$

The ISE architecture is shown in Figure 1. As we can see that the input of ISE is a Web document $\check{d}$. The final output of ISE is the shortest keyword sequence $\check{k}$. During the process, an initial keyword sequence k is fed into a Search Engine then a list of documents D is returned from the Internet with $\check{d}$ is one of them and ranked high. This process is repeated until $\check{d}$ is ranked at the top. After a further process that eliminates any superfluous words, k becomes the final output $\check{k}$.

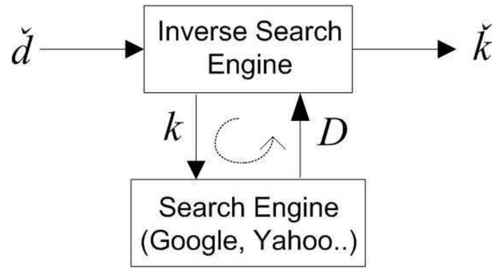


Figure 1: ISE Architecture

There are two naive methods that can exhaustively search for the resulting shortest KS. One is a bottom-up approach namely incremental brute-force. This approach tries all the possible word permutation from uni-gram to n-gram until the shortest keyword sequence is found. The other is a top-down approach namely shrinking brute-force. This approach starts from the keyword sequence that represents the whole content of the target page. Then, the system iteratively removes the "less informative" words from the candidate keyword sequence until the target page falls from its first rank in the search results. Clearly, exhaustive search for the shortest keyword sequence is expensive. The both brute-force methods therefore, are not desirable due to their time complexity.

We consider a heuristic approach to determining an initial candidate KS, so that the search results will contain the target page. The system then makes the target page progressively ranked higher by refining the candidate KS. Our ISE strategy consists of three stages: *embracing*, *expanding* and *eliminating*. At embracing stage, ISE finds an initial candidate KS, whose results contain the target page; then at expanding stage, the target page is topped by adding the words that can highly differentiate the target page from others; and finally at eliminating stage, the candidate KS is shrunk to a minimal size, and yet it can still make the target page ranked at the first place in the search results.

3.1 Embracing Stage

The objective of embracing stage is to determine a *seed* KS that would qualify the target page entering into a ranking process. Two simple heuristic strategies are used for the initial seed generation: *from title* and *from single important words*.

Intuitively, seed contains words that describe the target page. Such important words can be found in the HTML keyword fields, URL anchor text, and *emphasising HTML tags* such as <title>, headings (<h1> or <h2>), bold/italic (<b>/<i>) tags, or the text in larger fonts. Seed might be among terms in emphasised text fragments because those terms imply either their importance or relevance. The pseudo code of embracing strategy "From_Single_Word" is listed in Algorithm 1. The input of the algorithm is the target page $\check{d}$; while the out of the algorithm is a seed s and a list of important terms T.

Algorithm 1 Embracing stage with strategy "From_Single_Word".

```
function EMBRACE_{FromSingleWord}($\check{d}$)
    $\check{d} \mapsto T$   ▷ Terms extracted from $\check{d}$ is stored in term list $T$.
    for $\forall t : t \in T$ do
        if $t$ appears between emphasising tags then
            $t.w \leftarrow 10$   ▷ Emphasised terms weight: 10
        else
            $t.w \leftarrow 1$   ▷ Normal terms weight: 1
        end if
    end for
    Sort $T$ from "heavy" to "light"
    $s \leftarrow \emptyset$
    for $\tau \leftarrow 1$ to $\ell$ do   ▷ $\tau$: length of seed.
        for $\theta \leftarrow 1$ to $\Theta$ do
            Set the candidate seed $\hat{s}$ by choosing $\tau$ terms ($[t_\theta, t_{\theta+1}, \ldots, t_{\theta+\tau-1}]$) from T
            $D \leftarrow SE(\hat{s})$
            if $1 \geq Rank(\hat{s}, \check{d}) < Rank(s, \check{d})$ then
                $s \leftarrow \hat{s}$
            end if
        end for
        if $s \neq \emptyset$ then
            return ($s$,$T$)   ▷ Seed found.
        end if
    end for
    return ($s$,$T$)   ▷ $s = \emptyset$, Seed not found.
end function
```

To hedge against exhaustive search, three constant control parameters are applied to ISE algorithms. They are the maximum search range ϵ, which limits the number of maximum returned results; the maximum trial keyword sequence Θ, which limits the number of trials for different size; and the maximum keyword sequence length ℓ. If no seed can be found in the iteration, we consider this stage fail and will not proceed further.

The title of Web page is usually important because it is either a summary of Web page content, or the relative paths from the home Web page. It is possible to locate the target page by merely using its titles as the initial keyword sequence. Though the HTML specification does not mention the limitation of the title length, in practice, titles are seldom longer than one hundred characters because the title bar of Web browser windows usually does not have a sufficient space to display long titles. For the Web pages with no title, the first sentence within the HTML typesetting/emphasising tags can be treated as a title. If such feature still does not exist, then we use the first sentence of body text. The pseudo code of embracing strategy "From_Title" is listed in Algorithm 2. The input of the algorithm is the target page $\check{d}$ (as well as the title of the target page $d.t$). The output of the algorithm is a seed s and a list of important terms T.

Algorithm 2 Embracing stage with strategy "From_Title".

```
function EMBRACE_{FromTitle}($\check{d}$)
    $\check{d} \mapsto T$   ▷ Terms extracted from $\check{d}$ is stored in term list $T$.
    for $\forall t : t \in T$ do
        if $t$ appears between emphasising tags then
            $t.w \leftarrow 10$   ▷ Emphasised terms weight: 10
        else
            $t.w \leftarrow 1$   ▷ Normal terms weight: 1
        end if
    end for
    Sort $T$ from "heavy" to "light"
    $\hat{s} \leftarrow \check{d}.t$
    $s \leftarrow \emptyset$
    if $Rank(\hat{s}, \check{d}) >= 1$ then
        $s \leftarrow T$
        return ($s$,$T$)   ▷ Seed found.
    end if
    for $\tau \leftarrow 1$ to $\ell -$ length of $\check{d}.t$ do   ▷ $\tau$: length of seed.
        for $\theta \leftarrow 1$ to $\Theta$ do
            Append $\tau$ terms ($[t_\theta, t_{\theta+1}, \ldots, t_{\theta+\tau-1}]$) from T to candidate seed $\hat{s}$
            $D \leftarrow SE(\hat{s})$
            if $1 \geq Rank(\hat{s}, d) < Rank(s, d)$ then
                $s \leftarrow \hat{s}$
            end if
        end for
        if $s \neq \emptyset$ then
            return ($s$,$T$)   ▷ Seed found.
        end if
    end for
    return ($s$,$T$)   ▷ $s = \emptyset$, Seed not found.
end function
```

To choose a seed for it being more readable, popular, or user acceptable, the weight for each term in the word list can be multiplied by a word importance function $M(t)$ when an importance word list T is available, where t is a word. The higher return value of the function indicates the more likely the word is to be chosen for seeding. For example, 'glowworm' should have a higher weight than 'Arachnocampa', as 'glowworm' is a word easier to remember. A typical implementation for word importance is the word frequencies, except for the words in stopword list which should return 0. School teachers may like to set the return values of coarse words to 0 to prevent students from viewing inappropriate Web pages. The important word list is re-sorted after the weight of words in the list are updated. However, there is no such a universal word importance function currently available, we treat each word equally in our current design and the tests of the effectiveness of word importance function is not included in our experiments.

3.2 Expanding Stage

The objective of expanding stage is to find a candidate KS that makes the target page be ranked at the top of the search results. The seed derived at embracing stage is used as the initial candidate KS for the expanding. New terms from the important term list T shall be added to candidate KS until the candidate KS tops the target page.

The current selection method of appending new terms is based on the inverse document frequencies (IDFs) of those terms appeared within the documents of set D. The terms with low document frequencies (DFs) are more likely to improve the rank of the target page. The main reasons are that (1) they narrow down the range of documents to search, (2) they provide the search engine with indexes of all terms, and (3) they return only the documents that contain the keywords in query.

Document frequency is calculated based on Formula 4:

$$DF(t) = |d : d \ni t, d \in D - \check{d}| \quad (4)$$

After DF of each term in the target page is calculated, the term with the lowest DF is appended to the candidate KS. If the rank of the target page improves after appending term t, the term appending is repeated until the candidate KS tops the target page. Otherwise the second lowest DF term is appended and tested, and so forth.

During the process of expanding, a phenomenon called *Search Engine Shading* might occur such that none of the terms in $T_{\check{d}}$ would improve the rank of the target page at the expanding stage. The Search Engine Shading refers that the target document $\check{d}$ is "shaded" by other documents whose ranks are ahead of $\check{d}$ and whose terms form the supersets of that of the target document. For instance, two documents, d_1: "Cats are better than dogs" and d_2: "Cats are not better than dogs". are accessible for a search engine. The term sets of d_1 and d_2 are: { 'Cats', 'are', 'better', 'than', 'dogs'} and { 'Cats', 'are', 'better', 'than', 'dogs', 'not' } . That means $T_{d_1} \subseteq T_{d_2}$. If d_2 is ranked ahead of d_1, there might be no way to rank d_1 ahead of d_2 in the process. So when encountering Search Engine Shading, we need to relax the constraints and to return the candidate KS in order to make the target page ranking higher.

The pseudo code of expanding stage is listed in Algorithm 3. The input parameters of the algorithm are the target page $\check{d}$, the term list T, and the seed s from embracing stage. The algorithm returns a candidate KS k.

Search Engine Shading can be detected if the candidate KS k cannot top the target document when the algorithm stops.

Algorithm 3 Expanding stage.

```
function EXPANDING($\check{d}$,$T$,$s$)
    $k \leftarrow \hat{k} \leftarrow s$
    for $t : t \in T$ do $\hat{k} \leftarrow \hat{k} + t$
        if $Rank(\hat{k}, \check{d}) = 1$ then
            $k \leftarrow \hat{k}$
            return $k$
        else if  then$1 < Rank(\hat{k}, \check{d}) < Rank(s, \check{d})$
            $k \leftarrow \hat{k}$
            return $k$ ←Expanding($\check{d}$,$T$,$\hat{k}$)
        end if
    end for
    return $k$                    ▷ Search Engine Shading encountered.
end function
```

3.3 Eliminating Stage

The objective of eliminating stage is to delete superfluous terms in the candidate KS (denoted as k in the algorithm). A naive way to discover superfluous terms is to test every permutation of terms in k. However the time complexity of this method is $\sum_{i=0}^{|k|} P_i^{|k|}$, which is not desirable for a middle sized candidate KS. For instance, there could have 9,864,100 permutations for a 10-word candidate KS, and every permutation would issue a query to search engine.

In order to reduce the number of tests, we start elimination from the leading terms in KS until the elimination would make the target page fall from its top rank, or the terms in the candidate KS are all tested. The pseudo code of elimination process is listed in Algorithm 4. The input parameters of the algorithm are the target page $\check{d}$ and the candidate KS k from expanding stage. The algorithm returns the shortest KS $\check{k}$.

Algorithm 4 Eliminating stage.

```
function ELIMINATION($\check{d}$,$k$)
    $\check{k} \leftarrow k$.
    Let $T_k$ be the list of terms in $k$.
    for $t : t \in T_k$ do $k' \leftarrow$ Elimination($\check{d}$,$\check{k}$)
        if $1 \leq Rank(k', \check{d}) \leq Rank(\check{k}, \check{d})$ and $k'$ is shorter than $\check{k}$ then
            $\check{k} \leftarrow k'$               ▷ Shorter KS found.
        end if
    end for
    return $\check{k}$                   ▷ the shortest KS found.
end function
```

There are still some cases that the found KS is not the shortest. Other than Search Engine Shading, a possible reason of this problem is that there is a limit of the query length for the search engine efficiency and its security control. The process of finding the shortest KS may also fail if the character-length of KS is longer than the length-limit of a query.

3.4 Limitations

The proposed approach would not work in following circumstances [7]:

Non-indexed Web page: Embracing stage will fail if the target page is not searchable by a search engine.

Dynamic Web page: Page content is subject to constant changes. These pages include service pages, dynamic HTML and search result pages from other search engines or from a database.

Non-textual resource: If the target page does not contain textual information, then it is impossible for embracing stage to extract keyword.

Modern mainstream search engines have many advanced search functions such as phrase search, in-link, regular express, and non-negative search. However, these innovative search methods are hardly applicable to digital libraries or on intranets. For a wider applicability of ISE the above advanced features are not considered in the design of ISE.

4 Experiments

The experiments are designed to evaluate the effectiveness and efficiency of our ISE implementation. The ISE effectiveness is considered as whether a shortest KS can be successfully obtained. And if it is able to top the given target document. While the ISE efficiency is considered as whether the number of queries is reasonably small in order to obtain a shortest KS. It should be pointed out that the actual runtime analysis is not suitable here for ISE efficiency evaluation, as the runtime of ISE can heavily rely on the network traffic, which may not be always stable.

The effectiveness and efficiency of embracing strategies “FromSingleWord” and “FromTitle” are examined and compared with different search engines. The data set preparation, the evaluation methodology, and the results of experiments are shown in the rest of this section.

4.1 Experimental Data

The ISE implementation is tested with two major search engines: Live Search and Yahoo. We have also performed the experiments with Google, however, after we finished the experiments, we found out that Google’s terms of service [2] states that the automatic querying clients are not allowed without separate permission. Since our application for the permission has not yet been granted by Google, we cannot provide our experimental results on Google in this paper. Though the results are consistent to the other two search engines.

To verify our proposed approach, a shortest KS is required for every target page. But only a brute-force search can guarantee to found such shortest KSs. Here we provide *pseudo shortest keyword sequences* for each target page in test. A pseudo shortest KS is generated from randomly concatenated terms which were selected from the titles of category and sub-category of a very large and well-known project namely, Open Directory Project (ODP) [3]. In this way their titles and category/subcategory can form a unique identification of their corresponding Web documents. Therefore the criteria of the effectiveness are set objectively. All Web documents of the Open Directory Project can be potentially used as target pages and used as the initial search result regarding to the given pseudo shortest KSs. In other words, a pseudo shortest KS can guarantee that it will top the target page, even it is not always the actual one that could be found by a brute-force approach.

Since the evaluation sets from search engine vendors are not available yet, a pseudo shortest KS is a necessary devil. Indeed, the web pages indexed by pseudo shortest KSs tend to rank higher, but evaluation cannot proceed without knowing the existence of the shortest KS. Pseudo shortest KS also ensures the target pages be searchable. Establishing an experimental search engine seems plausible, however, a nearly enterprise-level search engine is required, otherwise the conclusions might be misleading because a shortest KS can be found in a few steps, not to mention an ad-hoc ISE function need to be developed for the experimental search engine.

For each search engine, 100 pseudo shortest KSs have been automatically generated in 10 different sizes (from one-word to 10-word). After excluding the non-HTML files (e.g., PDF or Microsoft Word files), 96 Web documents are used as the testing target pages for Yahoo and 99 for Live Search.

Other types of files are to be supported in future, as non-HTML files such as pdf, postscript, Microsoft Word documents, and Powerpoint slides, are now all searchable in modern search engines. Different parsers should be used to handle those formats.

4.2 Evaluation Methodology

The effectiveness of ISE is measured by the success rate and top-one rate. The success rate (SR) shows whether ISE can successfully get the shortest KSs. It is defined as:

$$SR = \frac{N_{\check{k}}}{N} \tag{5}$$

where $N_{\check{k}}$ is the number of the shortest KSs found, and N is the number of the target pages. The top-one rate (TR) shows whether a obtained shortest KS can top the corresponding target page. It is defined as:

$$TR = \frac{T_{\check{k}}}{N_{\check{k}}} \tag{6}$$

where $T_{\check{k}}$ is the number of obtained shortest KSs that top the target pages. The top-one rate is also inversely proportional to the occurrence of the Search Engine Shading problem. An ISE implementation is considered effective if both the SR and TR are in high percentage (i.e., close to 100 percent).

The efficiency of ISE is measured by the average shortness and average impoliteness. The shortness (SH) shows whether the obtained shortest KSs are shorter than the pseudo shortest KSs. It is defined as:

$$SH = \frac{|k|}{|\check{k}|} \tag{7}$$

where $|k|$ and $|\check{k}|$ are the numbers of terms in the pseudo shortest KS and in the shortest KS found. ISE is efficient if the computed shortness is greater than 1. The impoliteness (IP) shows whether ISE is polite to the search engine, that is, ISE is impolite when it sends a large number of queries to a search engine. IP is an important measure because it implies not only the inefficiency of ISE, but also the extra network traffic and search engine load. Moreover, some search engines may have daily query-quota for their client applications. The impoliteness is defined as:

$$IP = \log_{10} q \tag{8}$$

where q is number of queries sent to search engine to obtain a shortest KS. For a single automatic client like our ISE implementation, the daily query-quota stated by most search engines is 1000. Therefore, our ISE implementation is deemed to be acceptable if the impoliteness score is below 3 (i.e., the magnitude of the query-quota).

4.3 Results

4.3.1 Effectiveness Evaluation

The effectiveness of ISE with different embracing strategies is compared in this section. Table 1

[2] http://www.google.com/accounts/TOS

[3] see http://dmoz.org/. “The Open Directory is the most widely distributed data base of Web content classified by humans. The Open Directory powers the core directory services for the Web’s largest and most popular search engines and portals, including Netscape Search, AOL Search, Google, Lycos, HotBot, DirectHit, and hundreds of others.”

Table 1: Success Rate of Embracing Strategies

	FromSingleWord	FromTitle
Live Search	94.95 %	98.99 %
Yahoo	90.63 %	89.58 %

Table 2: Top-one Rate of Embracing Strategies.

	FromSingleWord	FromTitle
Live Search	100.00 %	100.00 %
Yahoo	100.00 %	100.00 %

presents the success rate (SR) of ISE with various embracing strategies in Live Search and Yahoo. High success rates suggest that both the seed and shortest KSs can be found in most cases.

Table 2 presents the top-one rate (TR) of ISE with various embracing strategies in Live Search and Yahoo. The results are perfect. These results show that whenever a shortest KS is found, it tops the given target page. In other words, if a seed can be obtained at embracing stage, then the shortest KS based on the seed will top the target page. Results in Table 2 also show that there is no occurrence of Search Engine Shading. However, we cannot safely conclude that Search Engine Shading will never happen, because we have only tested a limited number of target pages (i.e., 96 for Yahoo and 99 for Live Search).

4.3.2 Efficiency Evaluation

The efficiency of ISE of different embracing strategies are compared in this section. Figures 2 and 3 illustrate the comparisons of the sizes of candidate KSs (a seed is also a candidate KS) at every stage. In these figures, the solid and shading bars on the left represent the sizes of the candidate KSs of the "FromSingleWord" strategy, while bars on the right are for the "FromTitle" strategy at embracing stage. In order to compare the sizes among that of pseudo shortest KSs and candidate KSs, the sizes of pseudo shortest KSs are shown as the hollow wide bars on the left. A shorter KS size suggests a better KS-size efficiency. From these figures, it can be seen that the sizes of candidate KSs are seldom larger than that of the pseudo shortest KSs. This implies that most of the ISE-generated shortest KSs are not very long. We also observed that the sizes of the final shortest KSs are almost identical to their corresponding seeds in "FromSingleWord" strategy, while the sizes of the final shortest KSs are slightly shorter than that of their corresponding seeds in "FromTitle" strategy.

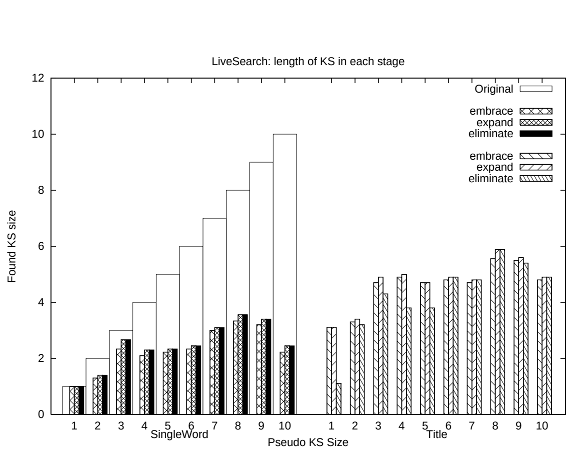


Figure 2: Live Search: Size of Candidate KS at Each Stage.

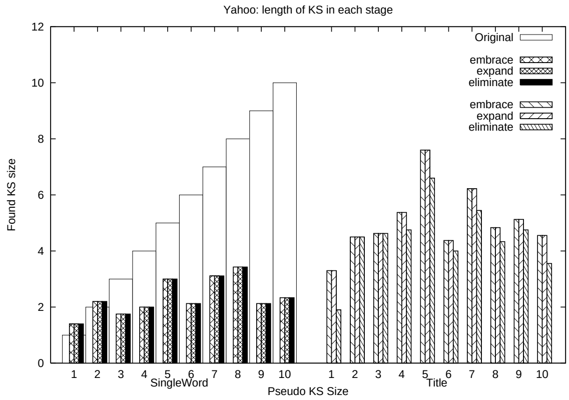


Figure 3: Yahoo: Size of Candidate KS at Each Stage.

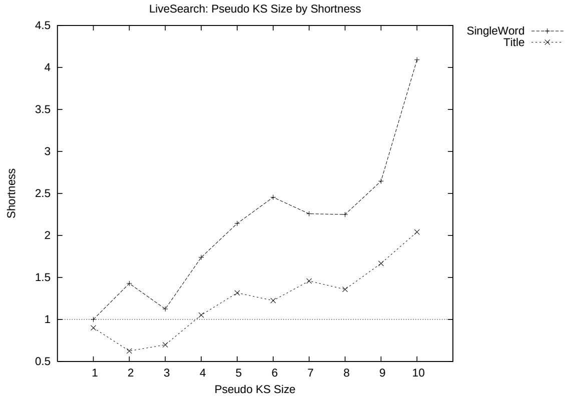


Figure 4: Live Search: Shortness Comparison between Embracing Strategies.

The shortness analysis is shown in Figures 4 and 5, and Table 3. A greater-than-one shortness score indicates that ISE is KS-size efficient. From these results, it can be seen that the both of our embracing strategies are KS-size efficient. Moreover, the embracing strategy "FromSingleWord" is better than "FromTitle" in terms of KS-size efficiency.

Figures 6 and 7 compare the numbers of queries required to obtain candidate KSs at every stage. In these figures, the solid and shading bars on the left show the numbers of queries required to obtain the candidate KSs for the embracing strategy "FromSingleWord", while the bars on the right are for the embracing strategy "FromTitle". A small number of

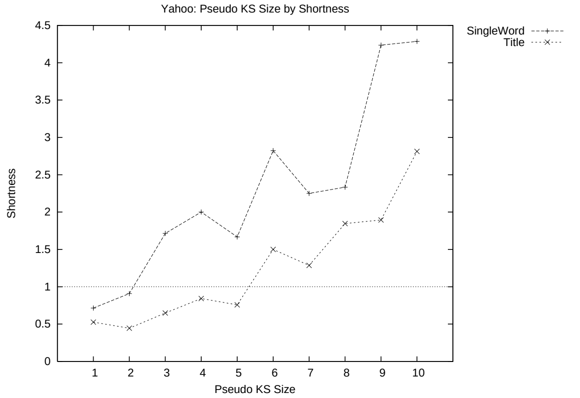


Figure 5: Yahoo: Shortness Comparison between Embracing Strategies.

Table 3: Contingency Table for Average Shortness of Search Engines.

	FromSingleWord	**FromTitle**
Live Search	2.39	1.88
Yahoo	2.72	1.72

Table 4: Average Impoliteness of Search Engines.

	FromSingleWord	**FromTitle**
Live Search	1.59	1.27
Yahoo	1.26	1.01

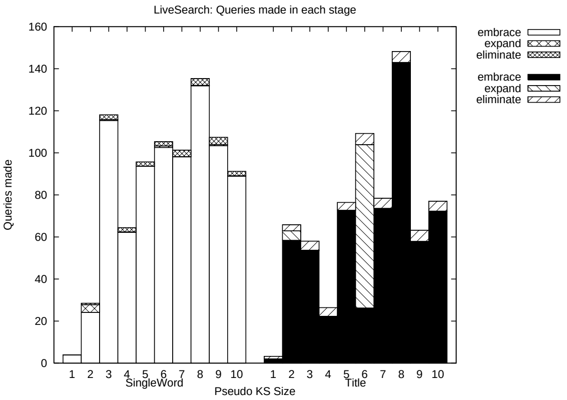


Figure 6: Live Search: Pseudo Shortest KS Size by Queries.

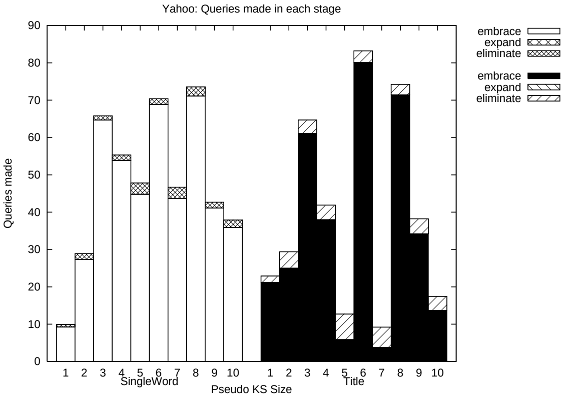


Figure 7: Yahoo: Pseudo Shortest KS Size by Queries.

queries suggests a better query efficiency. In these figures, most queries are issued at embracing stage, followed by eliminating stage, and in most cases, the expanding stage is rarely performed. This implies that the seeds could have already topped the target pages in nearly all cases. Figures 6 and 7 also show that the most shortest KSs can be obtained within 1000 queries, which is below the daily query-quota of the most search engines.

The impoliteness analysis is shown in Figures 8 and 9, and Table 4. A small score of impoliteness indicates that ISE is query-efficient. From these results, the embracing strategy "FromTitle" is more query-efficient than "FromSingleWord" in most cases.

So far no sign of the Search Engine Shading occurred in our experiments. As there exists a shortest KS for each target Web page in our experiments, we have not encountered the circumstances that Search Engine Shading would occur.

To sum up, our ISE implementation is effective and efficient, as the success rates and top-one rates are demonstrated successfully high (> 89%), the shortness scores are large (> 1), and the impoliteness scores are below the threshold (i.e., below the magnitude of the query-quota). Our experiments have effectively set up a benchmark for testing all other future ISE implementations and strategies. The experiment results also show that the embracing stage (seed generation) is the most important stage among the three, as the other two stages make relatively smaller contributions towards the final shortest KS. Thus, finding the better embracing strategies becomes a key to significantly improve the ISE effectiveness and efficiency.

5 Conclusions

Traditional keyword extraction algorithms do not consider the ranking of documents when keyword is extracted. This paper has defined a new type of system namely, Inverse Search Engine (ISE) to extract the shortest sequence of keywords from a Web page such that the keyword sequence can be used in a Web search and the Web page will be ranked as the first result by the given search engine. A number of challenges are addressed and the algorithms are proposed. The main contribution of this paper is the idea of the ranking-constrained keyword sequence extraction as well as the construction of ISE that can be used to discover the shortest keyword sequence for the unique identification of Web documents. The proposed three-stage algorithms are tested for their effectiveness and efficiency. An evaluation framework is proposed and the significant experiment results are demonstrated. Our experiment results can be regarded as a new benchmark for the further research in this direction.

One important issue still remains: 'shortest' is defined as the only one factor. In the real world, metrics such as the length of the keyword and popularity of the keyword would factor in. For example, one would prefer "insect lights Australia" over "Arachnocampa Springbrook". The shortest keyword sequence extracted from Web documents can be useful in many situations, such as Web indexing, content summary, Web advertisement. However, this usage should not prevent users from choosing words that they would prefer or know about, to search for the Web pages that they want.

Further work will focus on above issues and the popularity of keywords will be considered. The system performance regarding the scalability and supporting different file types will also be considered. In this case, the so-called *Search Engine Shading* would occur. The further experiments on the *Search Engine Shading* phenomenon will be given. As the initial embracing strategies can influence the effectiveness and efficiency significantly, a better embracing strategy might be designed to improve the ISE performance.

The idea of design and implementation of ISE has also raised another broad issue that if the whole Web-searchable documents on the World Wide Web can be partitioned into individuals according to their keyword phrases, the certain combination of keyword phrases could be 'owned' by the Web document authors.

References

[1] J. Battelle. John battelle's searchblog: Being jon kleinberg. Battellemedia.com (http://battellemedia.com/archives/000304.php), February 2004.

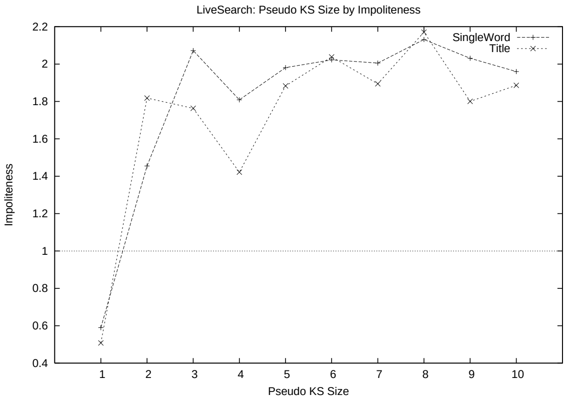


Figure 8: Live Search: Impoliteness Comparison between Embracing Strategies.

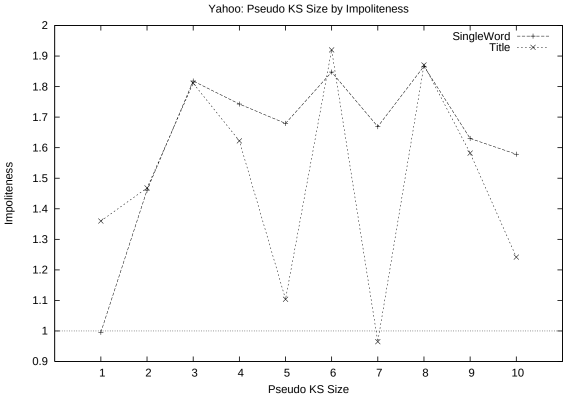


Figure 9: Yahoo: Impoliteness Comparison between Embracing Strategies.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[3] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. Domain-specific keyphrase extraction. In *IJCAI 1999: Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 668–673, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[4] J. Goodman and V. R. Carvalho. Implicit queries for email. In *CEAS 2005: 2nd Conference on Email and Anti-Spam*, 2005.

[5] D. Kelleher and S. Luz. Automatic hypertext keyphrase detection. In *IJCAI 2005: Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1608–1609, 2005.

[6] Y. Matsuo and M. Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(1):157–169, 2004.

[7] T. Phelps and R. Wilensky. *Robust Hyperlinks Cost Just Five Words Each.* University of California, Berkeley, Computer Science Division, 2000.

[8] C. Salton, G.and Yang. On the specification of term values in automatic indexing. *Journal of Document*, 29(4):351–372, Dec. 1973.

[9] K. Sparck Jones. A statistical interpretation of term specificity and its application to retrieval. *Journal of Document*, 28(1):11–20, March 1972.

[10] W. tau Yih, J. Goodman, and V. R. Carvalho. Finding advertising keywords on web pages. In *WWW 2006: Proceedings of the 15th international conference on World Wide Web*, pages 213–222, New York, NY, USA, 2006. ACM Press.

[11] P. D. Turney. Learning algorithms for keyphrase extraction. *IR: Information Retrieval*, 2(4):303–336, 2000.

Author Index

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 67 - Conceptual Modelling 2007
Edited by John F. Roddick and Annika Hinze. January, 2007. 978-1-920682-48-4.
Proc. Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM2007), Ballarat, Victoria, Australia, January 2007.

Volume 68 - ACSW Frontiers 2007
Edited by Ljiljana Brankovic, Paul Coddington, John F. Roddick, Chris Steketee, Jim Warren and Andrew Wendelborn. January, 2007. 978-1-920682-49-1.
Proc. ACSW Workshops - The Australasian Information Security Workshop: Privacy Enhancing Systems (AISW), the Australasian Symposium on Grid Computing and Research (AUSGRID), and the Australasian Workshop on Health Knowledge Management and Discovery (HKMD), Ballarat, Victoria, Australia, January 2007.

Volume 69 - Safety Critical Systems and Software 2006
Edited by Tony Cant. February, 2007. 978-1-920682-50-7.
Proc. 11th Australian Conference on Safety Critical Systems and Software, August 2006, Melbourne, Australia.

Volume 70 - Data Mining and Analytics 2007
Edited by Peter Christen, Paul Kennedy, Jiuyong Li, Inna Kolyshkina and Graham Williams. December, 2007. 978-1-920682-51-4.
Proc. 6th Australasian Data Mining Conference (AusDM 2007), Gold Coast, Australia. December 2007.

Volume 72 - Advances in Ontologies 2006
Edited by Mehmet Orgun and Thomas Meyer. December, 2006. 978-1-920682-53-8.
Proc. Australasian Ontology Workshop (AOW 2006), Hobart, Australia, December 2006.

Volume 73 - Intelligent Systems for Bioinformatics 2006
Edited by Mikael Boden and Timothy Bailey. December, 2006. 978-1-920682-54-5.
Proc. AI 2006 Workshop on Intelligent Systems for Bioinformatics (WISB-2006), Hobart, Australia, December 2006.

Volume 74 - Computer Science 2008
Edited by Gillian Dobbie and Bernard Mans. January, 2008. 978-1-920682-55-2.
Proc. 31st Australasian Computer Science Conference (ACSC2008), Wollongong, NSW, Australia, January 2008.

Volume 75 - Database Technologies 2008
Edited by Alan Fekete and Xuemin Lin. January, 2008. 978-1-920682-56-9.
Proc. 19th Australasian Database Conference (ADC2008), Wollongong, NSW, Australia, January 2008.

Volume 76 - User Interfaces 2008
Edited by Beryl Plimmer and Gerald Weber. January, 2008. 978-1-920682-57-6.
Proc. 9th Australasian User Interface Conference (AUIC2008), Wollongong, NSW, Australia, January 2008.

Volume 77 - Theory of Computing 2008
Edited by James Harland and Prabhu Manyem. January, 2008. 978-1-920682-58-3.
Proc. 14th Computing: The Australasian Theory Symposium (CATS2008), Wollongong, NSW, Australia, January 2008.

Volume 78 - Computing Education 2008
Edited by Simon and Margaret Hamilton. January, 2008. 978-1-920682-59-0.
Proc. 10th Australasian Computing Education Conference (ACE2008), Wollongong, NSW, Australia, January 2008.

Volume 79 - Conceptual Modelling 2008
Edited by Annika Hinze and Markus Kirchberg. January, 2008. 978-1-920682-60-6.
Proc. 5th Asia-Pacific Conference on Conceptual Modelling (APCCM2008), Wollongong, NSW, Australia, January 2008.

Volume 80 - Health Data and Knowledge Management 2008
Edited by James R. Warren, Ping Yu, John Yearwood and Jon D. Patrick. January, 2008. 978-1-920682-61-3.
Proc. Australasian Workshop on Health Data and Knowledge Management (HDKM 2008), Wollongong, NSW, Australia, January 2008.

Volume 81 - Information Security 2008
Edited by Ljiljana Brankovic and Mirka Miller. January, 2008. 978-1-920682-62-0.
Proc. Australasian Information Security Conference (AISC 2008), Wollongong, NSW, Australia, January 2008.

Volume 82 - Grid Computing and e-Research
Edited by Wayne Kelly and Paul Roe. January, 2008. 978-1-920682-63-7.
Proc. Australasian Workshop on Grid Computing and e-Research (AusGrid 2008), Wollongong, NSW, Australia, January 2008.

Volume 83 - Challenges in Conceptual Modelling
Edited by John Grundy, Sven Hartmann, Alberto H.F. Laender, Leszek Maciaszek and John F. Roddick. December, 2007. 978-1-920682-64-4.
Contains the tutorials, posters, panels and industrial contributions to the 26th International Conference on Conceptual Modeling - ER 2007.

Volume 84 - Artificial Intelligence and Data Mining 2007
Edited by Kok-Leong Ong, Wenyuan Li and Junbin Gao. December, 2007. 978-1-920682-65-1.
Proc. 2nd International Workshop on Integrating AI and Data Mining (AIDM 2007), Gold Coast, Australia. December 2007.

Volume 86 - Safety Critical Systems and Software 2007
Edited by Tony Cant. December, 2007. 978-1-920682-67-5.
Proc. 12th Australian Conference on Safety Critical Systems and Software, August 2006, Adelaide, Australia.

Volume 87 - Data Mining and Analytics 2008
Edited by John F. Roddick, Jiuyong Li, Peter Christen and Paul Kennedy. November, 2008. 978-1-920682-68-2.
Proc. 7th Australasian Data Mining Conference (AusDM 2008), Adelaide, Australia. December 2008.

Volume 90 - Advances in Ontologies
Edited by Thomas Meyer and Mehmet Orgun. September, 2008. 978-1-920682-71-2.
Proc. Knowledge Representation Ontology Workshop (KROW 2008), Sydney, September 2008.