

CONFERENCES IN RESEARCH AND PRACTICE IN  
INFORMATION TECHNOLOGY

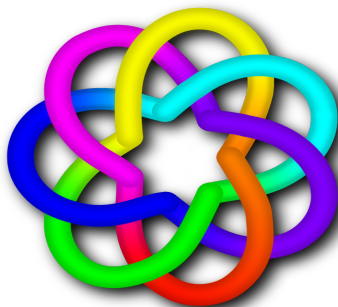
VOLUME 104

# DATABASE TECHNOLOGIES 2010

AUSTRALIAN COMPUTER SCIENCE COMMUNICATIONS, VOLUME 32, NUMBER 3



AUSTRALIAN  
COMPUTER  
SOCIETY



 **CORE**  
*Computing Research & Education*



# DATABASE TECHNOLOGIES 2010

Proceedings of the  
Twenty-First Australasian Database Conference  
(ADC 2010), Brisbane, Australia,  
January 2010

Heng Tao Shen and Athman Bouguettaya, Eds.

Volume 104 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 2010.** Proceedings of the Twenty-First Australasian Database Conference (ADC 2010), Brisbane, Australia, January 2010

**Conferences in Research and Practice in Information Technology, Volume 104.**

Copyright ©2010, Australian Computer Society. Reproduction for academic, not-for-profit purposes permitted provided the copyright text at the foot of the first page of each paper is included.

Editors:

**Heng Tao Shen**

School of Information Technology and Electrical Engineering  
The University of Queensland  
Brisbane St Lucia, QLD 4072,  
Australia  
Email: [shenht@itee.uq.edu.au](mailto:shenht@itee.uq.edu.au)

**Athman Bouguettaya**

CSIRO ICT Centre  
GPO Box 664  
Canberra ACT 2601,  
Australia  
Email: [Athman.Bouguettaya@csiro.au](mailto:Athman.Bouguettaya@csiro.au)

Series Editors:

Vladimir Estivill-Castro, Griffith University, Queensland  
Simeon J. Simoff, University of Western Sydney, NSW

[crpit@scm.uws.edu.au](mailto:crpit@scm.uws.edu.au)

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

Conferences in Research and Practice in Information Technology, Volume 104.  
ISSN 1445-1336.  
ISBN 978-1-920682-85-9.

Printed, December 2009 by UWS Press, Locked Bag 1797, South Penrith DC, NSW 1797, Australia  
Document engineering by Susan Henley, University of Western Sydney  
Cover Design by Matthew Brecknell, Queensland University of Technology  
CD Production by FATS Digital, 318 Montague Road, West End QLD 4101, <http://www.fats.com.au/>

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 Twenty-First Australasian Database Conference (ADC 2010), Brisbane, Australia, January 2010

Preface .....	vii
Programme Committee .....	viii
Organising Committee .....	x
Welcome from the Organising Committee .....	xi
CORE - Computing Research & Education .....	xii
ACSW Conferences and the Australian Computer Science Communications .....	xiii
ACSW and ADC 2010 Sponsors .....	xv

### Invited Papers

A Framework of Data Integration, Knowledge Management and User Behaviour Modelling in Health-care Applications of Diabetes .....	3
<i>Yanchun Zhang, Guandong Xu, Liping Wang and Kerry Bennett</i>	
Contrast Pattern Mining and its Applications .....	5
<i>Kotagiri Ramamohanarao</i>	

### Contributed Papers

How Consistent Are Human Judgments of Whether an Open Resource is Educational Material? ....	9
<i>Michael C. Harris, James A. Thom and Falk Scholer</i>	
Dynamic Framed-Slot ALOHA Anti-Collision using Precise Tag Estimation Scheme .....	19
<i>Prapassara Pupunwiwat and Bela Stantic</i>	
Scalable Online Index Construction with Multi-core CPUs .....	29
<i>Hiroyuki Yamada and Motomichi Toyama</i>	
Recursive Partitioning Method for Trajectory Indexing .....	37
<i>Elizabeth Antoine, Kotagiri Ramamohanarao, Jie Shao and Rui Zhang</i>	
Efficient Best Path Monitoring in Road Networks For Instant Local Traffic Information .....	47
<i>Shuo Shang, Ke Deng and Kai Zheng</i>	
Towards Unifying Advances in Twig Join Algorithms .....	57
<i>Nils Grimsmo and Truls A. Bjorklund</i>	
A Quantum Interpretation of the View-Update Problem .....	67
<i>Christian Flender</i>	
Counting Distinct Objects over Sliding Windows .....	75
<i>Wenjie Zhang, Ying Zhang, Muhammad Aamir Cheema and Xuemin Lin</i>	

Privacy-Aware Access Control in XML Databases .....	85
<i>Anders H. Landberg, J. Wenny Rahayu and Eric Pardede</i>	
Systematic Clustering Method for l-diversity Model .....	93
<i>Md Enamul Kabir, Hua Wang, Elisa Bertino and Yunxiang Chi</i>	
Tuning QoD in Stream Processing Engines .....	103
<i>Mohamed A. Sharaf, Panos K. Chrysanthis and Alexandros Labrinidis</i>	
Building a Dynamic Classifier for Large Text Data Collections .....	113
<i>Pavel Kalinov, Bela Stantic and Abdul Sattar</i>	
EP-based Robust Weighting Scheme for Fuzzy SVMs .....	123
<i>Shaoyi Zhang, Kotagiri Ramamohanarao and James C. Bezdek</i>	
Exploit Keyword Query Semantics and Structure of Data for Effective XML Keyword Search .....	133
<i>Khanh Nguyen and Jinli Cao</i>	
An Analysis of Spreadsheet-Based Services Mashup .....	141
<i>Dat Dac Hoang, Hye-young Paik and Boualem Benatallah</i>	
Optimizing XML Data with View Fragments .....	151
<i>Jun Liu, Mark Roantree and Zohra Bellahsene</i>	
A Real Time Hybrid Pattern Matching Scheme for Stock Time Series.....	161
<i>Zhe Zhang, Jian Jiang, Xiaoyan Liu, Ricky Lau, Huaqing Wang and Rui Zhang</i>	
Distributed Data Clustering in Multi-Dimensional Peer-To-Peer Networks .....	171
<i>Stefano Lodi, Gianluca Moro and Claudio Sartori</i>	
Stock Risk Mining by News .....	179
<i>Qi Pan, Hong Cheng, Di Wu, Jeffrey Xu and Yu Yiping Ke</i>	
<b>Author Index</b> .....	189

## Preface

The series of Australasian Database Conference is an annual forum for exploring novel technical developments and applications of database systems. The 21st Australasian Database Conference, ADC 2010, is held in Brisbane, Australia, as part of Australasian Computer Science Week.

ADC 2010 invited submissions of original contributions in all research areas of databases and its applications. The program committee received forty five submissions of full research papers; each was thoroughly reviewed by at least three PC members or external reviewers. Nineteen papers have been selected for presentation at the conference. In addition, the program committee invited two prominent researchers, Professor Kotagiri Ramamohanarao and Professor Yanchun Zhang for the traditional ADC invited talks.

The ADC PC chairs have also looked at all the accepted papers to select a paper to be awarded the conference's Best Paper. This year's Best Paper award goes to two papers. The first paper that won the award is "Counting Distinct Objects over Sliding Windows" by Wenjie Zhang, Ying Zhang, Muhammad Aamir Cheema and Xuemin Lin. The second paper that won the award is "Stock Risk Mining by News" by Qi Pan, Hong Cheng, Di Wu, Jeffrey Yu and Yiping Ke. 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.

**Heng Tao Shen**  
University of Queensland

**Athman Bouguettaya**  
CSIRO

ADC 2010 Programme Chairs  
January 2010

# Programme Committee and Review Panel

## Program Chairs

Heng Tao Shen (University of Queensland, Australia)  
Athman Bouguettaya (CSIRO, Australia)

## Programme Committee

Annika Hinze (University of Waikato, New Zealand)  
Anthony Tung (National University of Singapore, Singapore)  
Aoying Zhou (East China Normal University, China)  
Bin Cui (Peking University, China)  
Boualem Benatallah (University of New South Wales, Australia)  
Brahim Medjahed (University of Michigan, USA)  
Chen Li (University of California at Irvine, USA)  
Chengfei Liu (Swinburne University of Technology, Australia)  
Christian Bhm (University of Munich, Germany)  
Donghui Zhang (Northeastern University, USA)  
Egemen Tanin (University of Melbourne, Australia)  
Falk Scholer (RMIT, Australia)  
Gill Dobbie (University of Auckland, New Zealand)  
Guoren Wang (Northeast University, China)  
Helen Huang (University of Queensland, Australia)  
Helen Paik (University of New South Wales, Australia)  
Hong Cheng (Chinese University of Hong Kong, Hong Kong)  
Liam O'Brien (NICTA, Australia)  
Jenny Zhang (RMIT, Australia)  
Jialie Shen (Singapore Management University, Singapore)  
Jian Yang (Macquarie University, Australia)  
Jianyong Wang (Tsinghua University, China)  
Jie Shao (University of Melbourne, Australia)  
Jinli Cao (La Trobe University, Australia)  
Kian-Lee Tan (National University of Singapore, Singapore)  
Markus Stumptner (University of South Australia, Australia)  
Miyuki Nakano (Tokyo University, Japan)  
Mourad Ouzzani (Purdue University, USA)  
Quan Sheng (University of Adelaide, Australia)  
Rui Zhang (University of Melbourne, Australia)  
Shan Wang (Renmin University, China)  
Shazia Sadiq (University of Queensland, Australia)  
Shichao Zhang (University of Technology Sydney, Australia)  
Selcuk Candan (Arizona State University, USA)  
Walid Aref (University of Purdue, USA)  
Wei Wang (University of New South Wales, Australia)  
Xiangmin Zhou (CSIRO, Australia)  
Xuan Zhou (CSIRO, Australia)  
Xue Li (University of Queensland, Australia)  
Yoshiharu Ishikawa (Nagoya University, Japan)  
Yu Qi (Rochester Institute of Technology, USA)  
Yu Zheng (Microsoft Research Asia, China)  
Zahir Tari (RMIT, Australia)  
Zaki Malik (Wayne State University, USA)

## **Additional Reviewers**

Ali, Mohammed Eunos  
Asadzadeh, Parvin  
Barukh, Moshe  
Beheshti, Seyed Mehdi Reza  
Behm, Alexander  
Borkar, Vinayak  
Nutanong, Sarana  
Purohit, Vijendra  
Shao, Jie  
Sun, Haiyang  
Talukder, Nilothpal

Tao, Aries  
Thom, James  
Wang, Yi  
Weber, Ingo  
Xiao, Xiangye  
Xie, Hairuo  
Yang, Zhenglu  
Ye, Zhen  
Zheng, Huiyuan  
Zheng, Wenchen  
Zhou, Xuan

# Organising Committee

## Co-Chairs

Dr. Wayne Kelly  
Prof. Mark Looi

## Budget and Facilities

Mr. Malcolm Corney

## Catering and Booklet

Dr. Diane Corney

## Sponsorship and Web

Dr. Tony Sahama

## Senior Advisors

Prof. Colin Fidge  
Prof. Kerry Raymond

## Finance and Travel

Ms. Therese Currell  
Ms. Carol Richter

## Registration

Mr. Matt Williams

## DVD and Signage

Mr. Matthew Brecknell

## Satchels and T-shirts

Ms. Donna Teague

# Welcome from the Organising Committee

On behalf of the Australasian Computer Science Week 2010 (ACSW2010) Organising Committee, we welcome you to this year's event hosted by the Queensland University of Technology (QUT). Striving to be a "University for the Real World" our research and teaching has an applied emphasis. QUT is one of the largest producers of IT graduates in Australia with strong linkages with industry. Our courses and research span an extremely wide range of information technology, everything from traditional computer science, software engineering and information systems, to games and interactive entertainment.

We welcome delegates from over 21 countries, including Australia, New Zealand, USA, Finland, Italy, Japan, China, Brazil, Canada, Germany, Pakistan, Sweden, Austria, Bangladesh, Ireland, Norway, South Africa, Taiwan and Thailand. We trust you will enjoy both the experience of the ACSW 2010 event and also get to explore some of our beautiful city of Brisbane. At Brisbane's heart, beautifully restored sandstone buildings provide a delightful backdrop to the city's glass towers. The inner city clusters around the loops of the Brisbane River, connected to leafy, open-skied suburban communities by riverside bikeways. QUT's Garden's Point campus, the venue for ACSW 2010, is on the fringe of the city's botanical gardens and connected by the Goodwill Bridge to the Southbank tourist precinct.

ACSW2009 consists of the following conferences:

- Australasian Computer Science Conference (ACSC) (Chaired by Bernard Mans and Mark Reynolds)
- Australasian Computing Education Conference (ACE) (Chaired by Tony Clear and John Hamer)
- Australasian Database Conference (ADC) (ADC) (Chaired by Heng Tao Shen and Athman Bouguet-taya)
- Australasian Information Security Conference (AISC) (Chaired by Colin Boyd and Willy Susilo)
- Australasian User Interface Conference (AUIC) (Chaired by Christof Lutteroth and Paul Calder)
- Australasian Symposium on Parallel and Distributed Computing (AusPDC) (Chaired by Jinjun Chen and Rajiv Ranjan)
- Australasian Workshop on Health Informatics and Knowledge Management (HIKM) (Chaired by Anthony Maeder and David Hansen)
- Computing: The Australasian Theory Symposium (CATS) (Chaired by Taso Viglas and Alex Potanin)
- Asia-Pacific Conference on Conceptual Modelling (APCCM) (Chaired by Sebastian Link and Aditya Ghose)
- Australasian Computing Doctoral Consortium (ACDC) (Chaired by David Pearce and Rachel Cardell-Oliver).

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 ACSW2010 including the Organising Committee, the Conference Chairs and Programme Committees, our sponsors, the keynote speakers and the delegates. Special thanks to Justin Zobel from CORE and Alex Potanin (co-chair of ACSW2009) for his extensive advice and assistance. If ACSW2010 is run even half as well as ACSW2009 in Wellington then we will have done well.

**Dr Wayne Kelly and Professor Mark Looi**

Queensland University of Technology

ACSW2010 Co-Chairs

January, 2010

# CORE - Computing Research & Education

CORE welcomes all delegates to ACSW2010 in Brisbane. CORE, the peak body representing academic computer science in Australia and New Zealand, is responsible for the annual ACSW series of meetings, which are a unique opportunity for our community to network and to discuss research and topics of mutual interest. The original component conferences ACSC, ADC, and CATS, which formed the basis of ACSWin the mid 1990s now share the week with seven other events, which build on the diversity of the Australasian computing community.

In 2010, we have again chosen to feature a small number of plenary speakers from across the discipline: Andy Cockburn, Alon Halevy, and Stephen Kisely. I thank them for their contributions to ACSW2010. I also thank the keynote speakers invited to some of the individual conferences. The efforts of the conference chairs and their program committees have led to strong programs in all the conferences again, thanks. And thanks are particularly due to Wayne Kelly and his colleagues for organising what promises to be a strong event.

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

During 2009, CORE continued to work on journal and conference rankings. A key aim is now to maintain the rankings, which are widely used overseas as well as in Australia. Management of the rankings is a challenging process that needs to balance competing special interests as well as addressing the interests of the community as a whole. ACSW2010 includes a forum on rankings to discuss this process. Also in 2009 CORE proposed a standard for the undergraduate Computer Science curriculum, with the intention that it be used for accreditation of degrees in computer science.

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 2009; in particular, I thank Gill Dobbie, Jenny Edwards, Alan Fekete, Tom Gedeon, Leon Sterling, and the members of the executive and of the curriculum and ranking committees.

**Justin Zobel**

President, CORE  
January, 2010



# 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## Conference Acronyms

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

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

## ACSW and ADC 2010 Sponsors

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



CORE - Computing Research and Education,  
[www.core.edu.au](http://www.core.edu.au)



CEED,  
[www.corptech.com.au](http://www.corptech.com.au)



CSIRO ICT Centre,  
[www.csiro.au/org/ict.html](http://www.csiro.au/org/ict.html)



Queensland University of Technology,  
[www.qut.edu.au](http://www.qut.edu.au)



SAP Research,  
[www.sap.com/about/company/research](http://www.sap.com/about/company/research)



The Commonwealth Scientific and Industrial  
Research Organisation,  
[www.csiro.au](http://www.csiro.au)



AUSTRALIAN  
COMPUTER  
SOCIETY  
Australian Computer Society,  
[www.acs.org.au](http://www.acs.org.au)



THE UNIVERSITY  
OF QUEENSLAND  
AUSTRALIA  
University of Queensland,  
[www.uq.edu.au](http://www.uq.edu.au)



# INVITED PAPERS



# A Framework of Data Integration, Knowledge Management and User Behaviour Modelling in Healthcare Applications of Diabetes

Yanchun Zhang<sup>1</sup>, Guandong Xu<sup>1</sup>, Liping Wang<sup>2</sup> and Kerry Bennett<sup>2</sup>

<sup>1</sup>Centre for Applied Informatics, School of Engineering & Science

<sup>2</sup>Australian Community Centre for Diabetes

Victoria University

PO Box 14428, Vic 8001, Australia

{Yanchun.Zhang, Guandong.Xu, Liping.Wang, Kerry.Bennett}@vu.edu.au

## Abstract

In last few decades, with the advent of database systems and networking technologies, a huge volume of health data and valuable medical knowledge have been electronically available, accessible and processible, especially over the virtual cyberspace – the Web, even from a remote corner in the world. Nowadays the wide deployment of Hospital Information Management Systems (HIMS) and Web based clinical or medical systems, for example, the Medical Director, a generic GP clinical system, have made it possible to record, disseminate and implement the health information and clinical practices easily and globally. And health care and medical service is becoming more data-intensive and evidence-based since electronic health records are used to track individuals' and communities' health information (particularly changes). Australia has a relative advanced health computing and networking infrastructure, including high rate of Internet connectivity throughout health service providers, electronic prescribing records, and many administrative data collection units and services at national or state levels. These highlights substantially motivate and advance the emergence and the progress of data-centric health data and knowledge management research and practice, for example, *Health Informatics*.

In this paper, we aim at addressing the above challenges and difficulties encountered in diabetes health service. We propose a framework of data integration, knowledge management and user behaviour modelling for complementing and improving existing health care and service systems for diabetes. In particular, a comprehensive Web-based and knowledge-driven information system will be established to work as an information and communication hub on diabetes, providing a consolidated research repository, a dynamic online and mobile community and a strong information management support of integrating and sharing information on diabetes. The schematic system structure and functional components of the framework will be presented and demonstrated in this paper respectively.

**Keywords:** Data Integration, Knowledge Management, User Behaviour Modelling, diabetes healthcare service.

Type II diabetes is the fastest growing chronic disease and is the sixth cause of death in Australia. Certain regions, particularly western metropolitan Melbourne has an extremely high diabetes prevalence rate. Health research discovered that the prevalence and development of diabetes have a close relation to social determinant of population. Helping populations with, or at high risk of, diabetes to gain understanding of the disease, to make changes to their lifestyles, and to foster their confidence in accessing or contributing to services – in a manner that is inclusive, appropriate and flexible - is emerging as an urgent call from individuals and communities locally and nationally.

---

Copyright © 2010, Australian Computer Society, Inc. This paper appeared at the Twenty-First Australasian Database Conference (ADC2010), Brisbane, Australia, January 2010. Conferences in Research and Practice in Information Technology, Vol. 104, Heng Tao Shen and Athman Bouguettaya, Eds. Reproduction for academic, not-for-profit purposes permitted provided this text is included.





# Contrast pattern mining and its applications

Kotagiri Ramamohanarao

Department of Computer Science and Software Engineering

The University of Melbourne

Kotagiri@unimelb.edu.au

## Abstract

The ability to distinguish, differentiate and contrast between different data sets is a key objective in data mining. Such ability can assist domain experts to understand their data, and can help in building classification models. This presentation will introduce the principal techniques for contrasting data sets. It will also focus on some important real world application areas that illustrate how mining contrasts is advantageous.

## References

- 1) James Bailey, Thomas Manoukian, Kotagiri Ramamohanarao: Fast Algorithms for Mining Emerging Patterns. PKDD 2002: 39-50.
- 2) J. Bailey and T. Manoukian and K. Ramamohanarao: A Fast Algorithm for Computing Hypergraph Transversals and its Application in Mining Emerging Patterns. Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM). Pages 485-488. Florida, USA, November 2003.
- 3) Jinyan Li, Thomas Manoukian, Guozhu Dong, Kotagiri Ramamohanarao: Incremental Maintenance on the Border of the Space of Emerging Patterns. Data Min. Knowl. Discov. 9(1): 89-116 (2004).
- 4) Hamad Alhammady, Kotagiri Ramamohanarao: The Application of Emerging Patterns for Improving the Quality of Rare-Class Classification. PAKDD 2004: 207-211
- 5) Hamad Alhammady, Kotagiri Ramamohanarao: Using Emerging Patterns and Decision Trees in Rare-Class Classification. ICDM 2004: 315-318
- 6) Hamad Alhammady, Kotagiri Ramamohanarao: Expanding the Training Data Space Using Emerging Patterns and Genetic Methods. SDM 2005
- 7) Hamad Alhammady, Kotagiri Ramamohanarao: Using Emerging Patterns to Construct Weighted Decision Trees. IEEE Trans. Knowl. Data Eng. 18(7): 865-876 (2006).
- 8) Hongjian Fan, Kotagiri Ramamohanarao: An Efficient Single-Scan Algorithm for Mining Essential Jumping Emerging Patterns for Classification. PAKDD 2002: 456-462
- 9) Hongjian Fan, Kotagiri Ramamohanarao: Efficiently Mining Interesting Emerging Patterns. WAIM 2003: 189-201
- 10) Hongjian Fan, Kotagiri Ramamohanarao: Noise Tolerant Classification by Chi Emerging Patterns. PAKDD 2004: 201-206
- 11) Hongjian Fan, Ming Fan, Kotagiri Ramamohanarao, Mengxu Liu: Further Improving Emerging Pattern Based Classifiers Via Bagging. PAKDD 2006: 91-96
- 12) Hongjian Fan, Kotagiri Ramamohanarao: Fast Discovery and the Generalization of Strong Jumping Emerging Patterns for Building Compact and Accurate Classifiers. IEEE Trans. Knowl. Data Eng. 18(6): 721-737 (2006)
- 13) Jinyan Li, Guozhu Dong, Kotagiri Ramamohanarao: Instance-Based Classification by Emerging Patterns. PKDD 2000: 191-200
- 14) Jinyan Li, Guozhu Dong, Kotagiri Ramamohanarao: Making Use of the Most Expressive Jumping Emerging Patterns for Classification. PAKDD 2000: 220-232
- 15) Jinyan Li, Guozhu Dong, Kotagiri Ramamohanarao: Making Use of the Most Expressive Jumping Emerging Patterns for Classification. Knowl. Inf. Syst. 3(2): 131-145 (2001)
- 16) Jinyan Li, Guozhu Dong, Kotagiri Ramamohanarao, Limsoon Wong: DeEPs: A New Instance-Based Lazy Discovery and Classification System. Machine Learning 54(2): 99-124 (2004).
- 17) Jinyan Li, Kotagiri Ramamohanarao, Guozhu Dong: Combining the Strength of Pattern Frequency and Distance for Classification. PAKDD 2001: 455-466
- 18) Kotagiri Ramamohanarao, James Bailey: Discovery of Emerging Patterns and Their Use in Classification. Australian Conference on Artificial Intelligence 2003: 1-12
- 19) Ramamohanarao, K. and Bailey, J. and Fan, H. Efficient Mining of Contrast Patterns and Their Applications to Classification, Third International Conference on Intelligent Sensing and Information Processing, 2005 (39--47).
- 20) Ramamohanarao, K. and Fan, H. Patterns Based Classifiers, World Wide Web 2007: 10(71--83).
- 21) Qun Sun, Xiuzhen Zhang, Kotagiri Ramamohanarao: Noise Tolerance of EP-Based Classifiers. Australian Conference on Artificial Intelligence 2003: 796-806
- 22) Xiuzhen Zhang, Guozhu Dong, Kotagiri Ramamohanarao: Information-Based Classification by Aggregating Emerging Patterns. IDEAL 2000: 48-53
- 23) Zhou Wang, Hongjian Fan, Kotagiri Ramamohanarao: Exploiting Maximal Emerging Patterns for Classification. Australian Conference on Artificial Intelligence 2004: 1062-1068

---

Copyright © 2010, Australian Computer Society, Inc. This paper appeared at the Twenty-First Australasian Database Conference (ADC2010), Brisbane, Australia, January 2010. Conferences in Research and Practice in Information Technology, Vol. 104, Heng Tao Shen and Athman Bouguettaya, Eds. Reproduction for academic, not-for-profit purposes permitted provided this text is included.



# CONTRIBUTED PAPERS



# How Consistent Are Human Judgments of Whether an Open Resource is Educational Material?

Michael C. Harris

James A. Thom

Falk Scholer

School of Computer Science and Information Technology, RMIT University,  
GPO Box 2476, Melbourne, Victoria 3001, Australia.

Email: {michael.c.harris,james.thom,falk.scholer}@rmit.edu.au

## Abstract

Systems that filter web search results to return open educational resources need evaluation. The Cranfield method, which is widely used in information retrieval evaluation, can be used as the basis of a model for evaluating such systems. The Cranfield method requires a collection of resources with associated judgments. In this paper, we describe an experiment to collect judgments of whether open resources are educational material. We show experimentally that judges can agree on what resources are educational material, even in the absence of an educational context, and demonstrate that displaying the query used to retrieve the resource makes a judge less likely to rate a resource as educational.

**Keywords:** Open Educational Resources, Information Retrieval, Inter-rater Agreement

## 1 Introduction

The increase in the use of technology to support learners has led to a radical increase in the amount of digital teaching and learning material. Reusing existing material can increase the productivity of teaching staff, provide academics with more time to spend on value-added activities such as interacting with students, increase the return on investment of creating high-quality resources, and lead to the improvement of the quality of resources (Harris & Beiers 2005). However, the benefits of reuse cannot be realised if appropriate resources cannot be found.

This work explores a key issue related to the construction of collections appropriate for the evaluation of systems that filter web resources to identify learning material. The concept “likely to support learning” is not well defined, so complete agreement between judges rating resources according to that concept is unlikely. We investigate if people can broadly agree on what resources are likely to support learning in the face of this ambiguity.

This paper is organised as follows. We begin in Section 2 by discussing related research. In Section 3 previous work on agreement evaluation is detailed. In Section 4 we describe the user experiment conducted in this research, the results of which are analysed in Section 5. We then discuss what these results suggest about the ability of judges to agree on whether a resource is educational, and what impact this has on an evaluation methodology for systems that filter

educational resources. We conclude by outlining possible future work.

## 2 Related work

Significant time and money has been spent in the private and public sectors developing and maintaining systems designed to manage educational material, often called learning objects. For example, CANARIE<sup>1</sup>, the Canadian Advanced Network and Research for Industry and Education alone has spent almost \$C30 million on online learning projects, with the design and development of e-learning repositories being a major research theme (MacLeod 2005).

These repositories often expose incompatible access interfaces and contain few resources (Neven & Duval 2002). Research on the effective retrieval of educational material has assumed that all resources being searched are learning objects. However, many educational resources are released on the World Wide Web, and clearly there is much more on the Web than just learning material.

When publicly released, these resources are known as Open Educational Resources (OERs) (Wiley 2007), which are generally defined as “technology-enabled, open provision of educational resources for consultation, use and adaptation by a community of users for non-commercial purposes” (UNESCO 2002).

Previous work suggests that when people want to find digital material to support learning, they prefer to use a public search engine, such as Google<sup>2</sup> (Harris & Beiers 2005, Griffiths & Brophy 2005). These users may be more satisfied with search engine results if only resources likely to support learning were presented. To provide satisfactory result sets, resources that are unlikely to support learning should not be present. A filter to detect material that is likely to support learning would therefore be useful.

To ensure effectiveness, filters should be systematically evaluated. This evaluation can be carried out by assessing filter performance on a dataset where each resource has been labelled according to whether it should be retained or filtered out. This labelled dataset is called a *ground truth* or *gold standard*. This paper examines the establishment of a ground truth for evaluating systems that filter web resources for educational material.

The notion of a ground truth is also used in information retrieval (IR) system evaluation, and this suggests a useful starting point for developing an evaluation methodology for learning material filtering. The ground truth for IR systems evaluation is typically constructed by having relevance judgments assigned to resources by human judges. Systems are

Copyright ©2010, Australian Computer Society, Inc. This paper appeared at the Twenty-First Australasian Database Conference (ADC2010), Brisbane, Australia, January 2010. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 104, Heng Tao Shen and Athman Bouguettaya, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

<sup>1</sup><http://www.canarie.ca>

<sup>2</sup><http://www.google.com>

measured based on their ability to approximate the human-assigned relevance judgments. This is known as the *Cranfield method* after experiments carried out at Cranfield University in the 1960s (Cleverdon 1967).

The Cranfield method requires a collection of documents, a set of queries, and a set of relevance judgments linking the documents and the queries (Hildreth 2001). It is the standard approach used in information retrieval evaluation, and is used for evaluation in, for example, the Text REtrieval Conference (TREC) (Buckley & Voorhees 2005) and the Cross Language Evaluation Forum (CLEF) (Braschler & Peters 2002). This evaluation methodology has also been adapted for use in other retrieval evaluation domains, such as for XML retrieval evaluation (Kazai et al. 2003).

We propose that, using the Cranfield method as a model, systems that filter learning material can be evaluated based on their ability to select those resources that have been categorised by human judges as educational.

Experiments based upon the Cranfield method make the assumption that relevance is a property of resources in relation to a query, independent of the user. Under this assumption, the user and the context of retrieval is completely represented by the query (Saracevic 2007). While there has been debate about the validity of this assumption, it has been a useful starting point for IR experiments in general (Buckley & Voorhees 2005). We make a similar simplifying assumption, that a resource can be judged educational or not, independent of the specific educational context. Though context obviously plays an important role in education, we believe making the assumption of context independence is appropriate for the development of an evaluation methodology for retrieving educational resources. Our experiment explores this assumption.

In IR systems, the ground truth is constructed by assessing relevance, a concept which is complex and multi-dimensional (Saracevic 2007). For filtering educational resources, the ground truth should have assigned judgments of whether resources are educational or likely to support learning, which is also a complex concept. We investigate whether people can broadly agree on what resources are likely to support learning in the face of this complexity.

The level of agreement between raters will assist in deciding how many judgments are needed for each resource to establish an accurate ground truth. The use of a single assessor to judge the relevance of each resource has been a criticism of experiments based on the Cranfield method (Harter 1996). However, it is common in IR experiments to use a test collection where each resource has been assigned a relevance assessment by a single assessor, and this methodology has been shown to be adequate on small collections (Burgin 1992).

We further examine whether displaying the query used to retrieve a resource influences judgments and affects agreement.

Collaborative recommendation systems also use ratings that users assign to resources (Adomavicius & Tuzhilin 2005). In collaborative recommendation, ratings are collected from users in production systems and used to suggest resources to other users. Also related are information filtering systems, which find or remove resources from an incoming stream of data based on user profiles (Belkin & Croft 1992). However, these systems differ from judgments in IR evaluation experiments, and to what we propose in this paper, in that we collect ratings based on independent assessments of items; that is, the ratings have no relationship to users' general profiles.

### 3 Agreement evaluation methodology

In this section we provide some background on measures of agreement. Alongside our discussion of agreement measures, we present a worked example of each method, in the domain of judging educational material.

#### 3.1 Overlap

The usefulness of larger test collections with single assessments was experimentally supported in relation to the TREC collections by Voorhees, who showed that, despite variability of individual relevance assessments, the relative ranking of systems is stable (Voorhees 1998). In this work, TREC collections were reassessed by additional judges, and the level of agreement between all judges was measured using *overlap* (Voorhees 1998).

Overlap is the mean of the size of the intersection of positive ratings divided by the size of the union of positive ratings for each resource. That is, the average of the number of times both judges rated the resource relevant (for our purposes, rated the resource educational) divided by the number of times either judge rated the resource relevant. Voorhees reports the overlap measures between each of the three judges, and overlap across the three judges. However, the value of this overlap calculated across all judges will decrease as the number of judges increases, as a single dissenting judge counts as disagreement. For this reason, mean pairwise overlap is a more useful measure.

We use the example data from Table 1 to illustrate all agreement measures. Let there be  $J$  judges and  $R$  resources. For our example we use three judges ( $J = 3$ ) each rating five resources ( $R = 5$ ). A value of 0 represents a judgment that a resource was not educational, and a value of 1 represents a judgment that a resource was educational.

Table 1: Example ratings of three judges on five resources

Judges	Resources				
	1	2	3	4	5
1	0	0	1	1	1
2	0	0	0	1	1
3	0	1	1	0	1

For our example, pairwise overlap can be calculated as follows. We have three judges, and three pairwise comparisons; judge 1 with judge 2, judge 1 with judge 3, and judge 2 with judge 3. Consider judges 1 and 2; they agree that resources 4 and 5 are educational, so the intersection is 2. However, judge 1 also rated resource 3 as educational, so the union is 3. Overlap for these two judges is therefore  $\frac{2}{3} = 0.667$ . Mean pairwise overlap is the average overlap across all pairs of judges, as shown in Table 2.

Table 2: Example mean pairwise overlap

Judges	intersect	union	pairwise overlap
1 & 2	2	3	0.667
1 & 3	2	4	0.500
2 & 3	1	4	0.250
mean pairwise overlap			0.472

### 3.2 Raw agreement

A further simple measure commonly used is *raw agreement*, which is the proportion of observed agreement to possible agreement. Uebersax says, “Much neglected, raw agreement indices are important descriptive statistics. They have unique common-sense value. A study that reports only simple agreement rates can be very useful; a study that omits them but reports complex statistics may fail to inform readers at a practical level.” (Uebersax 2008)

In the context of assessments of whether resources are OERs, if a random resource is selected from a test collection, and we select a random rater who has judged the resource an OER, what is the probability that another random judge will agree? If the proportion of negative and positive judgments differ greatly, overall agreement will be biased towards the dominant judgments (Kundel & Polansky 2003). This often happens in relevance judgments, where there are likely to be far fewer documents that are relevant to a query than irrelevant documents. It is therefore important that the levels of positive and negative agreement are reported separately.

We derive the measure for raw agreement, limiting our discussion to the binary case, which we use in our experiment below. See Uebersax (2008) for calculation of raw agreement with an arbitrary number of categories.

We calculate raw agreement as follows. Let  $p_r$  be the number of times resource  $r$  was positively rated and  $n_r$  be the number of times resource  $r$  was negatively rated. To illustrate, we can see from Table 1 that resource 5 has three positive judgments, giving  $p_5 = 3$ , and resource 2 has two negative judgments, giving  $n_2 = 2$ . The number of times a rating was given to each resource in our example is shown in Table 3.

Table 3: Number of positive and negative ratings

Rating	Resources				
	1	2	3	4	5
n	3	2	1	1	0
p	0	1	2	2	3

There are  $p_r$  positive ratings for resource  $r$ . If we take one judge who rated resource  $r$  positively, there are  $p_r - 1$  judges that agree. Raw agreements are bi-directional, so total positive agreement is calculated by  $p_r(p_r - 1)$ . Negative agreement is calculated in the same way, taking negative ratings instead of positive ratings.

From the example judgments, take resource 4 from Table 3. We see that  $p_4 = 2$ , meaning that 2 judges said resource 4 is educational. Thus, the total number of agreements that resource 4 is educational is  $p_4(p_4 - 1) = 2(1) = 2$

Therefore, the observed agreement across all  $R$  resources can be expressed as follows, with  $A_{obs}^-$  representing observed agreement on negative judgments and  $A_{obs}^+$  observed agreement on positive judgments.

$$A_{obs}^- = \sum_{r=1}^R n_r(n_r - 1)$$

$$A_{obs}^+ = \sum_{r=1}^R p_r(p_r - 1)$$

The number of possible agreements,  $A_{poss}^-$  for negative agreement and  $A_{poss}^+$  for positive agreement, can be calculated similarly, but instead of taking the

number of judges that agreed with the original judge, we take the number of judges that *could* have agreed. Since a judge cannot agree with themselves, this means possible agreement is one fewer than the total number of judges, or one fewer than the total number of ratings,  $(p_r + n_r - 1)$ , and thus positive agreement for a resource is  $p_r(p_r + n_r - 1)$ .

$$A_{poss}^- = \sum_{r=1}^R n_r(n_r + p_r - 1)$$

$$A_{poss}^+ = \sum_{r=1}^R p_r(n_r + p_r - 1)$$

The observed and possible agreement for our example are shown in Table 4.

Table 4: Observed and possible agreement for each resource

	Resources					Total
	1	2	3	4	5	
$A_{obs}^-$	6	2	0	0	0	8
$A_{poss}^-$	6	4	2	2	0	14
$A_{obs}^+$	0	0	2	2	6	10
$A_{poss}^+$	0	2	4	4	6	16

Therefore, we can calculate specific agreement for both the positive ( $A^+$ ) and negative ( $A^-$ ) cases to be the proportion of observed agreement to possible agreement.

$$A^- = \frac{A_{obs}^-}{A_{poss}^-}$$

$$A^+ = \frac{A_{obs}^+}{A_{poss}^+}$$

For our example data, this means we have  $A^- = \frac{8}{14} = 0.571$  and  $A^+ = \frac{10}{16} = 0.625$ .

Overall agreement can similarly be calculated by dividing the observed agreement from both positive and negative judgments by the number of possible agreements.

$$A = \frac{A_{obs}^- + A_{obs}^+}{\sum_{r=1}^R (n_r + p_r)(n_r + p_r - 1)}$$

Of course,  $n_r + p_r$  is constant, the number of judgments made on a resource, and therefore the number of judges. We defined the number of judges earlier as  $J$ , so the overall agreement can simplified to the following.

$$A = \frac{A_{obs}^- + A_{obs}^+}{R \cdot J \cdot (J - 1)}$$

Using our example data, for overall agreement we have  $A = \frac{8+10}{(5)(3)(2)} = \frac{18}{30} = 0.6$ .

### 3.3 Kappa

The disadvantages of the overlap and raw agreement measures are that they are not corrected for chance, and it is not possible to estimate a confidence interval (Kundel & Polansky 2003). The index  $\kappa$  has been developed to address these issues; Cohen’s  $\kappa$  for two raters (Cohen 1960) and Fleiss’  $\kappa$  for multiple raters (Fleiss 1971).

In calculating Fleiss'  $\kappa$  we ask the question, given that we have some set of observed judgments, what agreement would we expect by chance? The proportion of agreement expected by chance can be represented as  $\bar{P}_e$ . If we take this value away from perfect agreement, we have the best agreement possible,  $1 - \bar{P}_e$ . If we take the chance agreement away from what was observed, which we can call  $\bar{P}_o$ , and divide it by the best possible agreement, we have the proportion of agreement that is not due to chance.

Therefore,  $\kappa$  can be defined as follows, with a value of 1 indicating complete agreement, and a value less than 0 representing agreement less than would be expected by chance.

$$\kappa = \frac{\bar{P}_o - \bar{P}_e}{1 - \bar{P}_e}$$

It is then possible to calculate the standard error, and a confidence interval. Calculating  $\kappa$  for our example data, we have  $\kappa = 0.196$  and  $p = 0.447$ . Therefore, though there is some agreement above chance, our example data does not show statistically significant agreement.

The table developed by Landis and Koch (Landis & Koch 1977) is sometimes used as a way to interpret values of  $\kappa$ . However, the levels chosen are arbitrary, are not applicable across experiments (Sim & Wright 2003) and can lead to unreliable conclusions (Gwet 2001). For these reasons, we do not report our results in relation to the Landis and Koch table.

While  $\kappa$  is used as a measure of agreement, it is not a test of the effect of classifying resources using two methods. We may observe significant agreement both when judging resources with a query visible and with the query not visible, but we want to be able to compare the levels of agreement. For this we use Fisher's exact test (Agresti 1992), which tests the null hypothesis that there is no difference in the proportions that raters assign resources to different categories under each condition.

In the next section, we describe our experiment design. As explained in this section, we take the Cranfield method as a starting point for our work.

## 4 Experiment design

Evaluation of systems that filter e-learning material differs from evaluation of relevance using the Cranfield method in that it seeks to collect classifications of resources according to a concept (supporting learning) as opposed to drawing a relationship (relevance) between a query and a document. To investigate how such classifications should be collected, we conducted a user experiment. The experiment investigates whether human judges can agree on what resources are likely to support learning, and whether visibility of the query used to retrieve the resource has an effect on judgments.

Eight judges were recruited for the experiment. Participants were acquaintances of the first author, from diverse backgrounds, and all had some experience with using web browsers and web interfaces.

A total of 20 resources were judged by the eight judges under one of two conditions: the query used to retrieve the resource being visible ( $q$ ) or not visible ( $q'$ ). Each judge viewed ten resources under condition  $q$  and ten under  $q'$ . A latin squares design was used to control for ordering effects.

### 4.1 Resource selection

The Flexible Learning Toolboxes are a collection of OERs managed by e-Works<sup>3</sup> that comply with the Sharable Content Object Reference Model (SCORM). A log of queries submitted to the live repository of the e-Works collection was obtained, containing 21139 queries, 7764 of them unique. Queries were drawn at random from the unique queries. If it was judged improbable that submitting a particular query to a search engine would return educational resources, that query was discarded. For example, the queries "rte2606a" and "a\*" were discarded. A total of 20 queries were selected for our experiments, and these are shown in the query terms column of Table 5.

While the queries were originally used for seeking resources from an educational repository, the resources used for our experiment were retrieved from a search across the entire web, with each query being submitted to the Yahoo! Search API<sup>4</sup>. Alongside each selected query, Table 5 shows the resources used in our experiment, which were selected as follows.

For the first ten queries, resources returned at rank position one were selected for judging. Call this set of resources  $R_A$ , resources 1 to 10 from Table 5.

To ensure that the collection contained an adequate proportion of resources for which a positive judgment was probable, resources returned using the second ten queries were judged by one of the experimenters, in rank order, according to the same criteria that would ultimately be used by the participants. For each query, the highest ranked resource judged likely to support learning was added to the collection. These judgments were made without reference to the search query used to retrieve the resources. Call these resources  $R_B$ , resources 11 to 20 from Table 5.

### 4.2 Presentation and user interface

Five resources from  $R_A$  and five from  $R_B$  were combined and their order randomised to form the first pool,  $P_1$ . The remaining resources were combined and their order randomised to form  $P_2$ . The judgment pools contained the following resources.

$$P_1 = [4, 11, 15, 5, 3, 13, 2, 12, 1, 14]$$

$$P_2 = [19, 6, 16, 18, 10, 8, 17, 20, 9, 7]$$

Resources were presented to judges in four ways, as described below.

- Group 1)  $P_1$  with the query followed by  $P_2$  without the query.
- Group 2)  $P_1$  without the query followed by  $P_2$  with the query.
- Group 3)  $P_2$  with the query followed by  $P_1$  without the query.
- Group 4)  $P_2$  without the query followed by  $P_1$  with the query.

For example, Group 1 were presented the resources from  $P_1$  with the original search query displayed, and then the resources from  $P_2$  were presented without the original search query displayed. The ten resources in  $P_1$  and  $P_2$  were always presented in the same order.

Two judges were randomly assigned to each group. Resources were presented sequentially via a web interface. Judges were asked to classify resources as likely or unlikely to support learning. Specifically,

<sup>3</sup><http://www.eworks.edu.au>

<sup>4</sup><http://developer.yahoo.com/search/web/>



Table 5: Resources in collection.

resource	query terms	rank	URL (http://)
$R_A$			
1	Communicate with colleagues and clients in an office environment	1	www.visualdataallc.com/clients.aspx
2	food safety practices	1	www.ehow.com/how_2075133.practice-food-safety.html
3	cash flow	1	en.wikipedia.org/wiki/Cash_flow
4	Maintain equipment for activities	1	www.govliquidation.com/list/c7484/lna/1.html
5	safe beauty	1	www.safeinternetshops.com/beauty.htm
6	search internet	1	kaftos.com
7	software development	1	en.wikipedia.org/wiki/Software_development
8	lifting safely	1	www.smarter.com/---se--qq-Lifting+Safely.html
9	cash budget	1	www.investopedia.com/terms/c/cashbudget.asp
10	Process customer complaints	1	www.bcuc.com/Complaint.aspx
$R_B$			
11	costing ingredients	4	www.ces.ncsu.edu/depts/poulsci/techmanuals/ingredient_sampling.html
12	prepare cook and serve food	3	www.cfsan.fda.gov/~dms/hret2-2.html
13	computing skills	3	www.cnn.com/TECH/computing/9806/03/autism.idg
14	treat weeds	1	www.blm.gov/weeds/FAQs/FAQs.htm
15	library skills	31	www.rock-hill.k12.sc.us/schools/high/sphs/Media/libraryskills.htm#Mod1
16	(plan and conduct and meetings)	3	www.azskillsusa.org/Teachers/meetings.htm
17	administer projects	8	sais-jhu.edu/cmtoolkit/issues/evaluation/index.html
18	Coordinate implementation of customer service strategies	50	en.wikipedia.org/wiki/Customer_relationship_management
19	write simple documents	1	nadoka.vipnet.org:8080/doc/user/08_is1.htm
20	Arts administration	3	en.wikipedia.org/wiki/Arts_administration

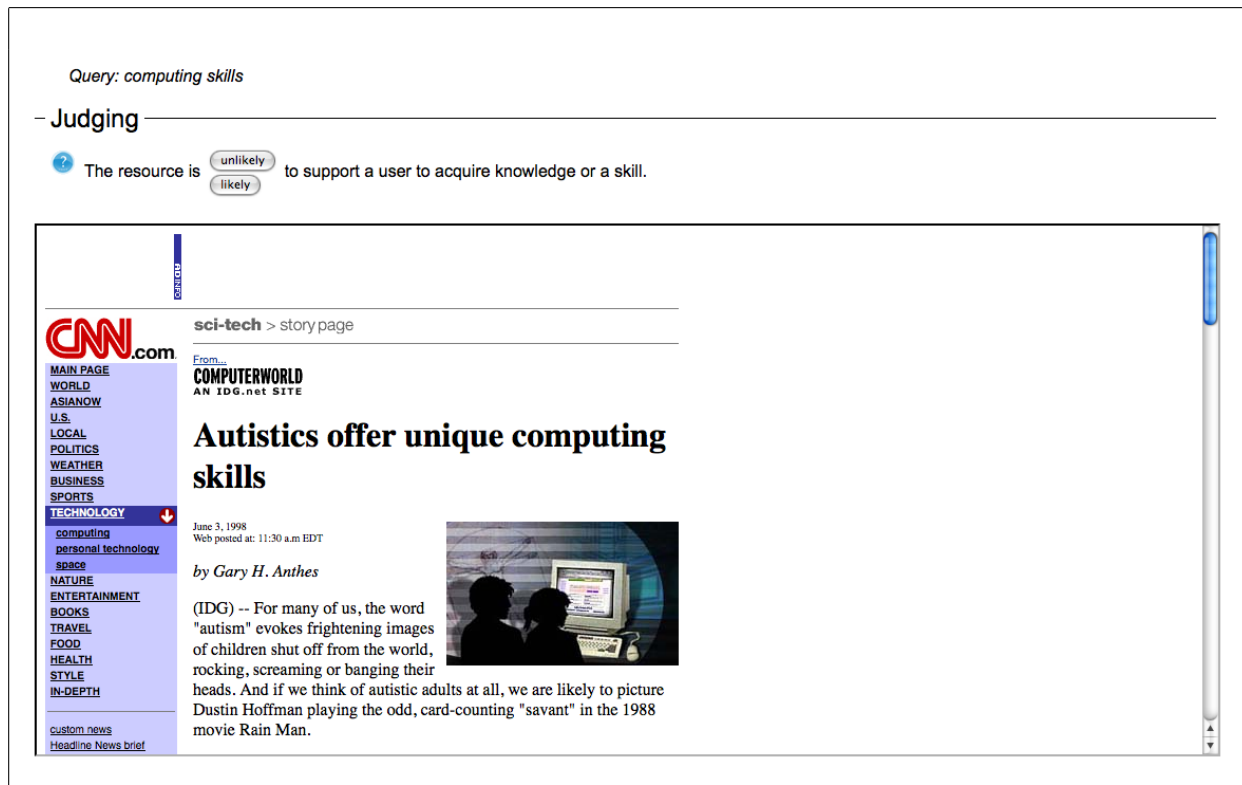


Figure 1: Judgment interface with query.

they were presented with the statement, “The resource is likely/unlikely to support a user to acquire knowledge or a skill,” where the words “likely” and “unlikely” were buttons that recorded the judgment. The judgment interface with the query visible is shown in Figure 1.

An HTML iframe element was used to embed single page resources in the judgment interface. All links within the resources were disabled, and raters evaluated the resources without reference to other web pages.

The resource displayed in Figure 1 is one of the resources used in our experiment, resource 13 from Table 5. The judgment interface without the query visible was identical, save for the removal of the query. Overall, judgments for this resource were split, with four judges believing it was educational material and four believing it was not. When the query used to retrieve the resource was not displayed, three of the four judges who assessed this resource said it was educational material. However, when judges could see the query, only one of the four assessors judged the resource as educational material.

Each participant answered several questions after each judgment, and after they completed all judgments, including being asked for comments about the resource they had just judged.

## 5 Analysis

This section reports on the results of our experimental evaluation of rater agreement, and provides analysis of these results. To give a general picture of the judgments, the number of times a resource was judged educational is shown in Figure 2. For our analysis, we begin by investigating rater agreement across all judgments regardless of other factors. Second, we discuss the impact of query visibility on rater agreement. We then report on the influence of resource type, that is, whether the resource was included in the judgment pool as a first ranked

Table 6: General agreement

Overlap	0.595
Negative	0.633
Positive	0.749
Overall	0.702
$\kappa$	0.382 ( $p < 0.001$ )

resource, or as a resource that was pre-selected to be a likely educational resource. Finally, we conclude by providing a discussion of comments that raters made after judging each resource.

### 5.1 General rater agreement

The frequency with which a number of raters judged resources as educational is shown in Figure 3. The leftmost bar represents the number of resources that no rater judged as educational, and the rightmost bar represents the number of times that all raters judged a resource educational.

We see a bimodal distribution, with higher frequencies at the extremes. This is as expected if there is a high level of agreement.

The agreement measures between the eight judges observed across all resources are presented in Table 6. All measures suggest a high level of agreement, and the value of  $\kappa$  is highly significant. The calculated mean pairwise overlap measure between the eight judges is 0.595, compared with the mean pairwise overlap measure between three assessors of 0.447 shown in Voorhees’ work that justified the use of a single judge in relevance assessments.

### 5.2 Query visibility

When building a collection for assessing systems using the Cranfield method, an assessor makes a judgment about the relevance of a document to a query. The query is therefore central to the process,

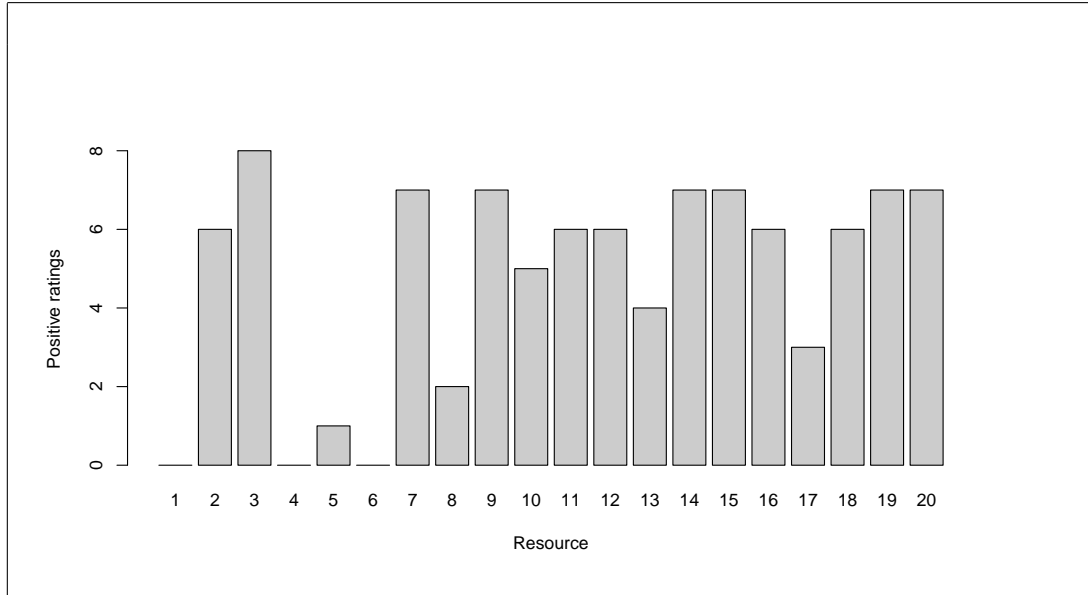


Figure 2: Positive OER judgments by resource and query visibility.

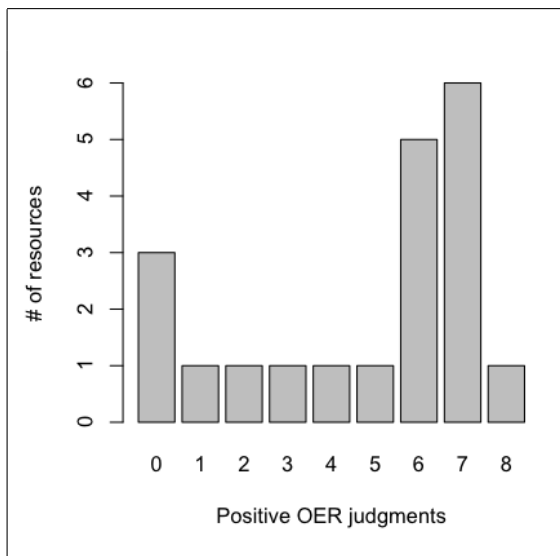


Figure 3: Overall frequency of positive OER judgments.

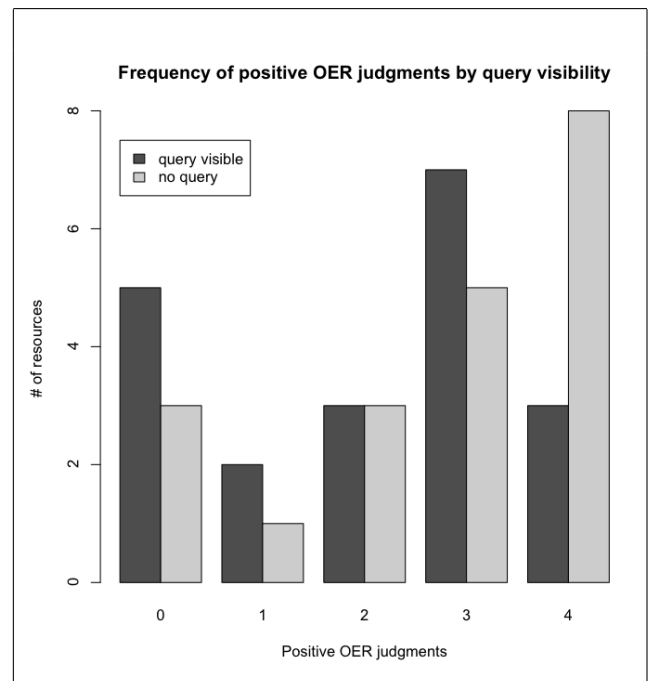


Figure 4: Frequency of positive OER judgments by query visibility.

and different queries will cause the resource to be judged differently. However, when judging whether a resource is educational, the judgment criteria is stable, and it is unclear what effect query visibility would have on the judging process. Here we report the results of varying query visibility.

Each resource has an *a priori* probability of being judged likely to support learning. In this case, we are interested in the conditional probability, that is, the probability a resource will be judged likely to support learning given query visibility.

Figure 4 shows the frequency with which a number of raters judged a resources as educational, separated by query visibility. Each resource was judged by four judges under each condition. The leftmost bar shows that, with the query visible five resources received no positive ratings, while without the query visible three resources received no positive ratings. The rightmost bar indicates that all raters judged a resource educational on three occasions when the query was visible and on eight occasions when the query was not visible.

As with Figure 3, bimodal distributions indicate

a high level of agreement. The distributions of frequency with and without the query being visible do appear to be generally bimodal, however, inspection suggests that displaying the query makes it less likely that a resource will be judged educational.

The agreement measures when split by query visibility are presented in Table 7. We can see that on all measures except negative agreement, agreement is noticeably higher when the query is not visible, though  $\kappa$  is significant in both cases. There is a very high level of positive agreement when the query is not visible, meaning that when the query is not visible raters very often agree that a resource is educational. It appears that judges use different criteria to rate a resource when the query is visible.

Fisher's exact test indicates that query visibility has a weakly significant effect on judgments ( $p = 0.053$ ).

Table 7: Agreement by query visibility

	query	no query
Overlap	0.516	0.685
Negative	0.667	0.615
Positive	0.683	0.815
Overall	0.675	0.750
$\kappa$	0.350 ( $p < 0.001$ )	0.430 ( $p < 0.001$ )

Table 8: Agreement by resource group

	$R_A$	$R_B$
Overlap	0.622	0.583
Negative	0.805	0.272
Positive	0.762	0.741
Overall	0.786	0.618
$\kappa$	0.567 ( $p < 0.001$ )	0.013 ( $p = 0.827$ )

### 5.3 Resource rank

Figure 5 shows the positive ratings made on each resource, that is, ratings where raters judged the resource educational, separated by query visibility. As described in Subsection 4.1, resources 1 through 10 were included in the judgment pool because they were returned at rank position one in response to a search for the first 10 queries ( $R_A$ ), and the resources 11 through 20 were included in the judgment pool because they were the highest ranked resource judged educational from the results returned in response to the second 10 queries ( $R_B$ ).

The agreement measures when split by resource group are presented in Table 8. On all measures, agreement is lower for resources in  $R_B$ , and though we see similar values for overlap, positive agreement and overall agreement,  $\kappa$  does not show significant agreement for  $R_B$ . Negative agreement is particularly low for  $R_B$  when compared with  $R_A$ .

Fisher's exact test indicates that how a resource was added to the judgment pool has a significant effect on judgments ( $p < 0.001$ ). This means that the proportions of negative and positive judgments are different depending on whether the resource was a first ranked resource or was the highest ranked resource judged to be educational.

### 5.4 Rater comments

In the judgment interface raters were invited to make comments about their judgments. In total, raters made 91 comments from the 160 judgments. Seven of the eight raters made at least one comment after judging a resource. Approximately a third of the comments make reference to the query used to retrieve the resource. For example, after judging a resource with the query visible one rater said, "Query asking general question; resource for much more specific request which is likely irrelevant. Therefore, easy to judge," and another said, "query not specific enough," and "if it was autism and computing skills this would be a useful resource."

In some cases, the rater stated that they found the resource difficult to judge because the query was not known, "Specific resource and without search terms, difficult to determine whether relevant to query; therefore, difficult to judge."

These comments appear to suggest that raters find it more difficult to judge whether a resource is educational in the absence of the context given by a query. It might be expected that a more difficult judgment decision would take longer to make; however timing measurements reveal that there is no significant interaction between judging times and query visibility.

## 6 Discussion and future work

The methodology presented in this paper produces a ground truth that can be used in the evaluation of systems that filter web search results for educational resources. This methodology is based upon the Cranfield method, which is commonly used in IR experiments. Further, we establish how many judges should rate each resource, and whether the queries used in the retrieval of resources should be presented to assessors as part of the judgment interface.

We present a user experiment in which participants judged whether web resources were educational, in a manner similar to the way relevance assessments are collected when building test collections for use in experiments using the Cranfield method.

Our results show a high level of general agreement. Indeed, our results show a level of agreement higher than that used in the IR literature to justify the use of a single assessor. We conclude that, given this high level of agreement, an appropriate methodology for building a test collection for the evaluation of systems that filter for educational resources would involve having resources categorised by a single judge rather than have multiple judges categorise each resource. In particular, for fixed time and number of judges, it would be more useful to judge a larger number of resources than have multiple judgments on fewer resources.

In relation to query visibility, our results show a high level of agreement both with and without the query visible. The level of agreement is higher when the query is not visible, though this is only weakly significant overall. While we found nothing to suggest that the query needs to be displayed, judges did report that they found the task difficult. The task may be made simpler by asking judges to rate resources in an artificial context and then to make a judgment as to whether the resource is educational in other contexts.

This effect is significant when the resource is not the first ranked result. This result is intuitively reasonable, as the search engine used for retrieving the resource has rated the resource as less relevant than other resources, as reflected in its ranking. It may be that the query distracts raters from the task of judging whether a resource is likely to support learning, and causes them to judge relevance instead.

When people use search engines generally, they issue a query and judge how well the documents returned meet their information need. That is, they judge the relevance of returned resources to their query. Therefore, when presented with a resource to judge, and the query that was used to retrieve it, it is unsurprising that their judgments reflect relevance. As we are interested in filtering educational resources, relevance is handled by the search engine, and therefore should be factored out for our purposes.

Our experiment used a fixed description when asking judges to rate resources. Future work could examine what instructions should be given to judges and what effect this may have on agreement.

The selection of resources appropriate for inclusion in a test collection must also be addressed. We contend that it is appropriate to submit queries to a search engine and select returned resources for the collection. In this work, initial queries came from a log of queries submitted to a repository for e-learning material. This is appropriate in that the users were searching for the type of material in which we are interested. However, a user's search behaviour may be different when searching a specific repository as opposed to the wider web, and thus the queries may not be representative of the sorts of queries that would be submitted to a filtering system. Equally, queries

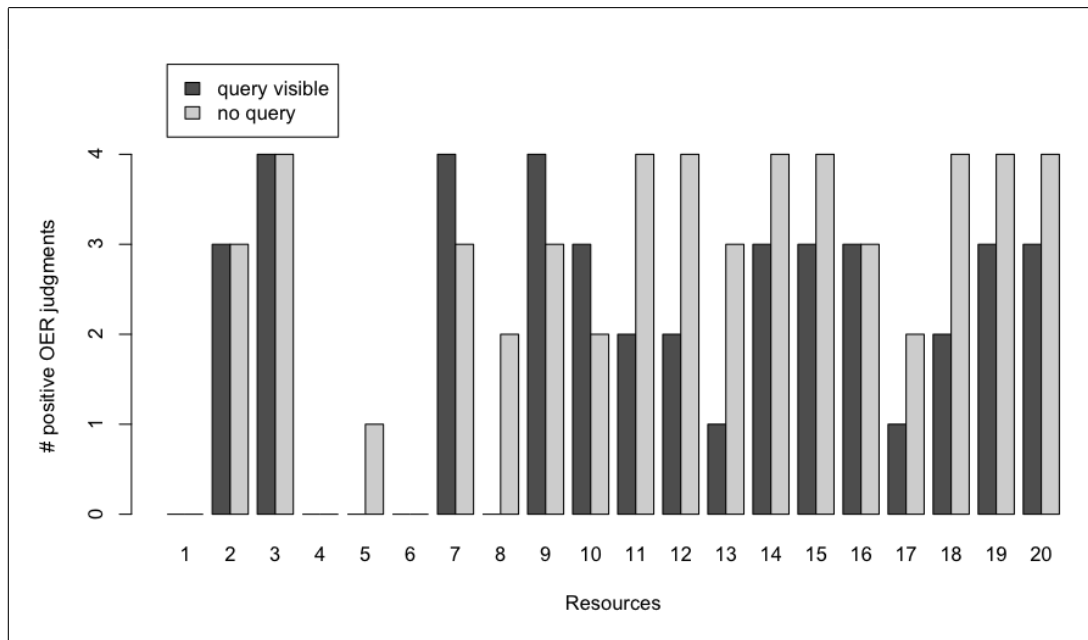


Figure 5: Positive OER judgments by resource and query visibility.

selected from a general query log, without knowledge of user intent, are likely to be inappropriate.

Resources must then be selected from the ranked resources returned by a search engine in response to a query. Table 5 shows that most of the resources  $R_B$  (those included in the pool because they were judged likely to support learning) were ranked in the top 10 results. The mean rank was 7.7 and the median was 3. Also, half the resources in  $R_A$  (those included as the first ranked result) were judged educational by a majority of the judges. This suggests that a reasonable percentage of highly ranked resources will be judged likely to support learning, and that we need not have taken the precaution of pre-judging some resources. Therefore, it is appropriate to include the first  $N$  results from the returned ranked results in the collection. The results of this preliminary study suggest that  $N = 10$  is sufficient.

Our future work will involve the construction of a test collection appropriate for the evaluation of educational resource filters. We will construct such filters, using some of the features from the resources judged in this work as starting points, and evaluate them using the methodology outlined here. For example, resources not judged educational often include a large proportion of links when compared to content, whereas resources judged educational often have large amounts of text separated by headings and a higher proportion of internal links.

One shortcoming of this work, and an opportunity for future work, is that we have only considered single page resources. However, it is likely that multi-page resources are more interesting from a learning point of view. This is related to the concept of *granularity* as used in reference to reusable learning objects.

## References

- Adomavicius, G. & Tuzhilin, A. (2005), 'Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions', *IEEE Transactions on Knowledge and Data Engineering* **17**(6), 734–749.
- Agresti, A. (1992), 'A survey of exact inference for contingency tables', *Statistical Science* **7**(1), 131–153.
- Belkin, N. J. & Croft, W. B. (1992), 'Information filtering and information retrieval: two sides of the same coin?', *Communications of the ACM* **35**(12), 29–38.
- Braschler, M. & Peters, C. (2002), CLEF methodology and metrics, in C. Peters, M. Braschler, J. Gonzalo & M. Kluck, eds, 'Cross-Language Information Retrieval and Evaluation', Springer-Verlag, Lecture Notes in Computer Science, Vol. 2406, pp. 394–404.
- Buckley, C. & Voorhees, E. (2005), *Retrieval System Evaluation*, MIT Press, Cambridge, MA, USA, pp. 53–75.
- Burgin, R. (1992), 'Variations in relevance judgments and the evaluation of retrieval performance', *Information Processing and Management* **28**(5), 619–627.
- Cleverdon, C. (1967), 'The Cranfield tests on index languages devices', *Aslib Proceedings* **19**(6), 173–194.
- Cohen, J. (1960), 'A coefficient of agreement for nominal scales', *Educational and Psychological Measurement* **20**(1), 37–46.
- Fleiss, J. L. (1971), 'Measuring nominal scale agreement among many raters', *Psychological Bulletin* **76**(5), 378–382.
- Griffiths, J. & Brophy, P. (2005), 'Student searching behaviour and the web: use of academic resources and Google', *Library Trends* **53**(4), 539–554.
- Gwet, K. (2001), *Statistical tables for inter-rater agreement*, STATAXIS Publishing Company.
- Harris, M. C. & Beiers, H. (2005), Barriers to the reuse of learning objects, in 'EdMedia 2005 - World Conference on Educational Multimedia, Hypermedia & Telecommunications', pp. 482–489.
- Harter, S. P. (1996), 'Variations in relevance assessments and the measurement of retrieval effectiveness', *Journal of the American Society for Information Science* **47**, 37–49.

- Hildreth, C. R. (2001), 'Accounting for users' inflated assessments of on-line catalogue search performance and usefulness: an experimental study', *Information Research* **6**(2).
- Kazai, G., Gövert, N., Lalmas, M. & Fuhr, N. (2003), The INEX evaluation initiative, in H. Blanken, T. Grabs, H.-J. Schek, R. Schenkel & G. Weikum, eds, 'Intelligent search on XML data', Springer-Verlag, Lecture Notes in Computer Science, Vol. 2818, pp. 279–293.
- Kundel, H. L. & Polansky, M. (2003), 'Measurement of observer agreement', *Radiology* **228**(2), 303–308.
- Landis, J. R. & Koch, G. G. (1977), 'The measurement of observer agreement for categorical data', *Biometrics* **33**, 159–174.
- MacLeod, D. (2005), Learning object repositories: Deployment and diffusion, E-learning report, CANARIE Inc., Ottawa, Ontario.  
URL: [http://http://www.canarie.ca/funding/elearning/2005\\_LOR\\_final\\_report.pdf](http://http://www.canarie.ca/funding/elearning/2005_LOR_final_report.pdf)  
Accessed: 12 September 2008.
- Neven, F. & Duval, E. (2002), Reusable learning objects: a survey of LOM-based repositories, in 'Proceedings of the tenth ACM international conference on Multimedia', pp. 291–294.
- Saracevic, T. (2007), 'Relevance: A review of the literature and a framework for thinking on the notion in information science. part ii: nature and manifestations of relevance', *Journal of the American Society for Information Science and Technology* **58**(13), 1915–1933.
- Sim, J. & Wright, C. C. (2003), 'The kappa statistic in reliability studies: use, interpretation, and sample size requirements', *Physical Therapy* **85**(3), 257–268.
- Uebersax, J. (2008), 'Raw agreement indices'.  
URL: <http://ourworld.compuserve.com/homepages/jsuebersax/raw.htm>  
Accessed: 10 February 2009.
- UNESCO (2002), 'UNESCO promotes new initiative for free educational resources on the Internet'.  
URL: [http://www.unesco.org/education/news\\_en/080702\\_free\\_edu\\_ress.shtml](http://www.unesco.org/education/news_en/080702_free_edu_ress.shtml)  
Accessed: 18 February 2009.
- Voorhees, E. M. (1998), Variations in relevance judgments and the measurement of retrieval effectiveness, in 'SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', ACM, pp. 315–323.
- Wiley, D. A. (2007), On the sustainability of open educational resource initiatives in higher education, Technical report, Organisation for economic co-operation and development (OECD).  
URL: <http://www.oecd.org/dataoecd/33/9/38645447.pdf>  
Accessed: 17 February 2009.

# Dynamic Framed-Slot ALOHA Anti-Collision using Precise Tag Estimation Scheme

Prapassara Pupunwiwat

Bela Stantic

Institute for Integrated and Intelligent Systems  
Griffith University, Queensland, 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. Despite an emergence of RFID technology, multiple tag identification, where a reader identifies a multiple number of tags in a very short time, is still a major problem. This is known as *Collision* problem and can be solved by using anti-collision scheme. The current tree-based anti-collision approach suffers from long identification delay, while the ALOHA-Based approach suffers from tag starvation problem due to inaccurate Frame-size. In this paper, we propose a “Precise Tag Estimation Scheme” for a *Dynamic Framed-Slot ALOHA* (DFSA), which estimates precise number of tags around the reader. In this empirical study, we compare our approaches with the original tag estimation in DFSA. The results indicate that the various parameters used by “Precise Tag Estimation Scheme”, including *empty slots* variables and/or *collision slots* variables, have an impact on system efficiency. Thus, the number of frames and slots used by DFSA can be minimised by adjusting correct variables.

**Keywords:** Radio Frequency Identification - RFID, Anti-Collision, Frame-Size Estimation, Backlog Estimation

## 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 a RFID system, when numerous tags are present in the interrogation zone at the same time, the reader requires an ability to read data from the individual tag. A technical scheme that handles tag collision without any interference is called an anti-collision protocol. A RFID reader is a powerful device that has sufficient power and memory, while a passive tag

requires energy from the radio signal sent by a reader. The reader issues a request command to the tags and each tag will send its ID to the reader. If only one tag responds, the reader can receive the tag's ID. However, if more than two tags send their IDs simultaneously, a collision occurs and the IDs need to be re-transmitted according to an anti-collision scheme. The main focus of an anti-collision scheme is to read multiple tags as fast and reliably as possible.

The two types of tag anti-collision algorithms widely used in RFID systems are the *Tree-based Deterministic Anti-collision* and the *ALOHA-based Probabilistic Anti-collision*. Tree-based algorithms make trees while performing the tag identification procedure using a unique ID of each tag, which lead to lengthy queries issued by reader that causes long identification delay. On the other hand, ALOHA-based algorithms decreases the probability of collision by scheduling the responses of tags at random time, which lead to tag starvation problems where not all tags can be identified.

In this study, we propose a new “Precise Tag Estimation Scheme” (PTES) for Backlog estimation and Frame-size estimation compatible with Dynamic Framed-Slot ALOHA. The motivation of this work is to achieve a more accurate estimation of number of tags within an interrogation zone, which leads to a more accurate frame-size and system efficiency. The methodology for PTES is first derived, and experiment is then conducted in order to prove the efficiency of the proposed technique. The results and analysis of the experiments have indicated that PTES, using various parameters, has an impact on DFSA system efficiency and the number of frames and slots used can be minimised by adjusting the correct variables.

The remainder of this paper is organised as follows: In section 2, some general background on RFID and information related to collision and different anti-collision schemes are provided. In section 3, we discuss different probabilistic anti-collision schemes compatible with EPC Gen2 and their limitations. In section 4, we present a new technique, the “Precise Tag Estimation Scheme” and methodology. In section 5, we present experimental evaluation, 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).

## 2.1 Tag and Reader Collision

Simultaneous transmissions in RFID systems lead to collisions as the readers and tags typically operate on the same channel. Three types of collisions are possible: Reader-Reader collision, Reader-Tag collision, and Tag-Tag collision.

- **Reader-to-Reader:** Interference occurs when one reader transmits a signal that interferes with the operation of another reader and prevents the second reader from communicating with tags in its interrogation zone (Jain & Das 2006). Reader-to-Reader collision can be easily avoided by determining the appropriate reader's deployment that prevents direct signal interference between two or more readers.
- **Reader-to-Tag:** Interference occurs when one tag is simultaneously located in the interrogation zone of two or more readers, where more than one reader attempts to communicate with that tag at the same time (Jain & Das 2006).
- **Tag-to-Tag:** Tag collision in RFID systems, also known as Multi-access, happens when multiple tags are energised by the RFID tag reader simultaneously, and reflect their respective signals back to the reader at the same time. This problem is often seen whenever a large volume of tags must be read together in the same reader zone. The reader is unable to differentiate these signals.

## 2.2 Tag Anti-Collision Approaches

Tag collision or Multi-access problem is more complex than those within reader collision categories. Therefore, in this paper we only focus on tag anti-collision approaches. The various types of tag anti-collision approaches for Multi-access/Tag collision can be reduced to two basic types: Tree-based approach and ALOHA-based approach.

### 2.2.1 Tree-Based Approaches

The Tree-based approach 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 approach is slow and introduces a long identification delay but leads to fewer collisions, and has 100 percent successful identification rate (Choi et al. 2008).

Such Tree-based approaches can be classified into a Memory-based algorithm and a Memoryless-based algorithm (Myung & Lee 2006a), (Myung & Lee 2006b), (Ryu et al. 2007). In the Memory-based 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.

### 2.2.2 ALOHA-Based Approaches

In an ALOHA-based approach, tags respond at randomly generated times. If a collision occurs, colliding tags will have to identify themselves again after waiting a random period of time. This technique is faster than Tree-based but suffers from tag starvation problem where not all tags can be identified due to the random nature of chosen time.

The ALOHA-based approaches usually refer to "Slot ALOHA" (Quan et al. 2006), which introduces discrete time-slots for tags to be identified by reader at the specific time. The principle of Slotted-ALOHA techniques is based on the "Pure ALOHA" introduced in early 1970s (Abramson 1970), where each tag is identified randomly. To improve the performance and throughput rate, a "Frame Slotted ALOHA" (Shin et al. 2007) 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 Frame Slotted ALOHA uses a fixed frame-size and does not change the frame-size during the process of tag identification. This is simple, but not efficient for tag identification. The "Dynamic Framed Slotted ALOHA" algorithm can change the frame-size to increase the efficiency of tag identification; and there have been several researches to improve the accuracy of frame-size by implementing a Frame Estimation Tool (Wang et al. 2007), (Lee et al. 2005), (Lee et al. 2008), (Cho et al. 2007).

In this study, we focus on improving the Frame-size estimation, since EPC Class 1 Gen2 protocol (EPCGlobal 2006) adopted Dynamic Framed-Slot ALOHA probabilistic algorithm to solve the collision problem.

## 3 Dynamic Framed-Slot ALOHA Algorithm

Probabilistic algorithms are based on ALOHA protocol. Each tag in an interrogation zone selects one of the given  $N$  slots to transmit its identifier. All tags will be recognised after a few frames. Figure 1 shows an example of Dynamic Framed-Slot ALOHA anti-collision protocols. Each frame is formed of specific number of slots that is used for the communication between the readers and the tags. The frame-size is dynamically changed according to estimated number of 'Backlog', which is a number of tags that have not been read. Any slot that has more than one tags responding to it is classified as a collision slot, while any slot that has exactly one tag responding to it is a successful slot. In addition, empty slot is where there is no tag reply and should be minimised in order to increase system efficiency. Figure 1 shows that Slot 1 and 2 of Frame one and Slot 5 of Frame two are collision slots; and Slot 3 of Frame one, Slot 4 of Frame two, and Slot 6 and 7 of Frame three are successful slots.

Frame	1			2		3	
Slot	1	2	3	4	5	6	7
State	Collision	Collision	Success	Success	Collision	Success	Success
Tag A							
Tag B							
Tag C							
Tag D							
Tag E							

Figure 1: This figure shows a sample procedure of Dynamic Framed-Slot ALOHA.

### 3.1 DFSA for EPC Class 1 Gen2

EPC Class 1 Generation 2 is widely used in Ultra High Frequency (UHF) range for communications at 860-960MHz; where the standards have been created by EPCGlobal (EPCGlobal 2006). EPC Class 1 Gen2



protocol adopted the Dynamic Framed-Slot ALOHA-based probabilistic algorithm to solve the collision problem.

According to the protocol, the reader picks tag within an interrogation zone by the command “SELECT”; then issues “QUERY”, which contains a ‘Q’ parameter to specify the frame-size (frame-size  $F = 2^Q - 1$ ). Each selected tag will pick a random number between 0 to  $2^Q - 1$  and put it into its slot counter. The tag, which picks zero as its slot number, will respond to the reader with a ‘RN16’. After receiving the RN16 from tag, reader will transmit ‘ACK’ containing the received RN16, after which the tag with slot number zero will backscatter its EPC to reader. Then, reader issues “QUERYREP” or “QUERYADJUST” command to initiate another slot (Wang et al. 2007).

In order to query the remaining tags, reader may issue “QUERYREP” and each tag will subtract 1 from its own slot number, or “QUERYADJUST” (Adjust Q value); and each tag will pick a new random number within 0 to  $2^Q - 1$  as its new slot number. The tag whose new slot number is zero, will response to reader and then backscatter its EPC (Li et al. 2009). There are three kinds of slot: 1) Successful slot where there is only one tag reply, 2) Empty slot where there is no tag reply, and 3) Collision slot where there is more than one tag reply; as shown in figure 2.

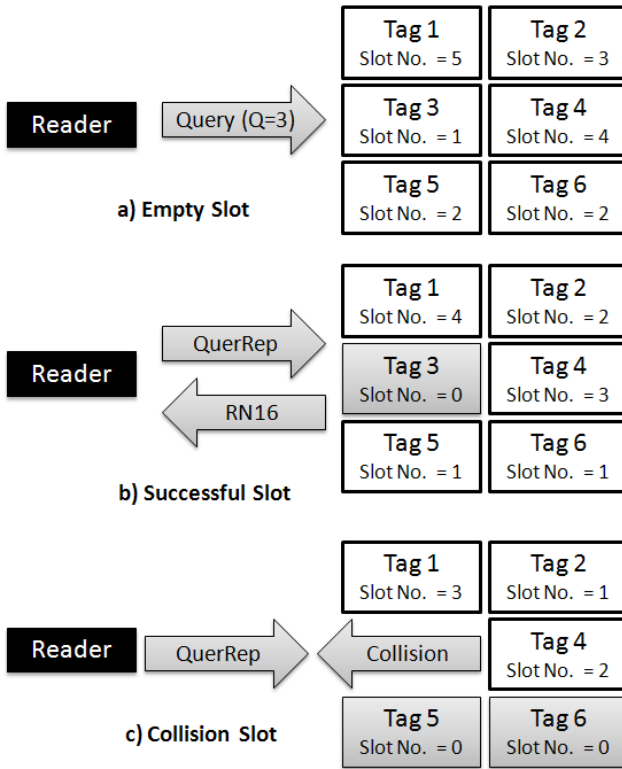


Figure 2: Empty Slot, Successful Slot, and Collision Slot in EPC Class 1 Gen2 Protocol.

### 3.2 Mathematic Fundamental for ALOHA-based Tag Estimation

In the ALOHA-based probabilistic scheme, to estimate the number of present tags, Binomial distribution is a good fundamental method (Wang et al. 2007), (Li et al. 2009). For a given initial  $Q$  in a frame with  $F$  slots and  $n$  tags, the expected value of the number of slots with occupancy number  $x$  is as follows:

$$a_x = n \times C_n^x \left(\frac{1}{F}\right)^x \left(1 - \frac{1}{F}\right)^{n-x}$$

Therefore, the expected number of Empty slot  $e$ , Successful slot  $s$ , and Collision slot  $c$  are given by the following equations:

$$\begin{cases} e = a_0 = F\left(1 - \frac{1}{F}\right)^n \\ s = a_1 = n\left(1 - \frac{1}{F}\right)^{n-1} \\ c = a_k = F - a_0 - a_1 \end{cases}$$

Thus, the system efficiency is defined as the ratio between Successful slot number and Frame-size given by the following equations:

$$E = \frac{s}{F} = \frac{n\left(1 - \frac{1}{F}\right)^{n-1}}{F} = n \frac{1}{F} \left(1 - \frac{1}{F}\right)^{n-1}$$

It is proven that the highest efficiency can be obtained if the Frame-size is equal to the number of tags, provided that all slots have the same fixed length:

$$F(\text{optimal}) = n$$

#### 3.2.1 Schoute Backlog Estimation Technique

Schoute (1983) developed a Backlog estimation technique for Dynamic Framed-Slot ALOHA using Poisson distribution. The Backlog, after the current frame Bt, is given by equation:

$$Bt = 2.39c$$

where  $c$  represents the number of collided slot in the current frame. This technique has the best performance, where fewest frames were used compared to other algorithms.

#### 3.2.2 Lowerbound Backlog Estimation Technique

The estimation function is obtained under the assumption that a collision involves at least two different tags. Therefore, Backlog after the current frame Bt is given by equation:

$$Bt = 2c$$

where  $c$  is the number of collided slot in the current frame.

#### 3.2.3 Other Backlog Estimation Techniques

There have been several researches on Backlog estimation including C-Ratio method (Cha & Kim 2005), Chen1 and Chen2 methods (Chen 2006), Vogt method (Vogt 2002), and Bayesian method (Floerke-meier 2007). These methods are either having worse performances than simple Schoute's method or too complicated to implement for RFID system. Therefore, we only compare our method to Schoute and Lowerbound methods because the two methods are simple and have excellent performances.

## 4 Methodology

In order to overcome shortcomings of inaccurate Frame-size estimation methods, we propose a “Precise Tag Estimation Scheme” (PTES), which introduces three tag estimation methods compatible with *Dynamic Framed-Slot ALOHA* used in EPC Class

1 Generation 2. Each method of the PTES estimates specific number of tags according to a specific equation. After precise number of tags is estimated, frame-size for the next round of identification process can be predicted. This section will describe the specific requirements for tag estimation, initial  $Q$  value, suggest frame-size, the newly proposed PTES, and sample tag estimation and allocation.

#### 4.1 Slots Observation and Initial Q-Value

According to DFSA algorithm, the reader picks tag within an interrogation zone by the command "SELECT"; then issues "QUERY", which contains a 'Q' parameter to specify the frame-size,  $F = 2^Q - 1$ . For our methodology, initial  $Q$  value can be any number between 1 and 10 since we assume that a reader can at most pick up no more than 800 tags per round. After first round of identification, collision slots and empty slots will be observed and used to estimate number of tags. Three methods are proposed for Precise Tag Estimation Scheme: 1) Method One - Various Empty Parameters, 2) Method Two - Various Collision Parameters, and 3) Method Three - Various Collision and Empty Parameters. The three methods consist of different equations for number of tags estimation. After the number of tags has been estimated, frame-size for the next identification round can be configured. The suggested frame-size is explained in the following sub-section.

#### 4.2 Suggested Frame-Size

The suggested frame-size for our methodology is set according to the estimated number of tags. For example, if an estimated number of tags are to be around 100 tags, the suggested frame-size would have a  $Q$  value of 7. Since the frame-size is calculated by  $2^Q - 1$ , the frame-size where  $Q = 7$  will allow at most 128 tags ( $0$  to  $2^7 - 1$ ) to be identified. Therefore, if the estimated number of tags is between 65 and 128 tags, the suggested  $Q$  would equal to 7. Table 1 shows minimum and maximum number of tags allowed per suggested frame-size. The table only demonstrates up to  $Q = 10$  since we assume that no more than 800 tags will be captured each round by the reader.

$2^Q$	$2^1$	$2^2$	$2^3$	$2^4$	$2^5$
Min	0	3	5	9	17
Max	2	4	8	16	32
$2^Q$	$2^6$	$2^7$	$2^8$	$2^9$	$2^{10}$
Min	33	65	129	257	513
Max	64	128	256	512	1024

Table 1: Suggested Frame-size for specific estimated number of tags. Maximum tags of each Frame-size is calculated by  $2^Q - 1$

#### 4.3 Precise Tag Estimation Scheme

In this paper, we propose three tag estimation methods for a Precise Tags Estimation Scheme. In method One - *Various Empty Parameters* (VEP), we use a fix parameter to calculate collision slot and use a variable to predict empty slot for the next round of identification. In method two - *Various Collision Parameters* (VCP), we use a variable to predict collision slot for the next round. Finally, in method three - *Various Collision and Empty Parameters* (VCEP), we use two variables to predict collision slot and empty slot for the new identification round. The aims of all methods

are to clarify that both collision slots and empty slots have an impact on backlog prediction; and that more than one variable can be used to predict Frame-size for an upcoming round effectively. The three methods are explained within this sub-section.

##### 4.3.1 Method One - VEP

*Various Empty Parameters* method intends to obtain the optimal parameter in order to calculate and predict the closest number of remaining tags for the next identification round. We assume that for the current identification round, each collision slot has at least 2 tags collided. However, we cannot know for sure how many tags actually caused the collision. There is exactly 1 tag per successful slot, thus, we do not take successful slot into consideration. On the other hand, empty slot will continuously occur during the next rounds of identification. Therefore, VEP method is created to find the optimal parameter to predict the number of empty slot; and to find the impact of variable used on empty slots prediction.

The VEP method uses a variable between 0.1 and 0.9 to predict the number of empty slot. Since empty slot does not engage any tag, we assume that the parameter for empty slot calculation falls between 0.1 and 0.9 (more than 0 but less than 1). Equation (1) shows *Backlog* (Number of unidentified tags) estimation using fix parameter  $V_1 = 2.0$  for collision slot prediction and variable  $0 < V_2 < 1$  for empty slot prediction.

$$Backlog = V_1(c) + V_2(e).....(1)$$

where  $c$  = Collision Slot;  $e$  = Empty slot;  $V_1 = 2.0$ ; and  $V_2$  = Variable between 0.1 and 0.9 incrementing by 0.1.

$$Backlog = 2.0(c) + 0.1(e)$$

$$Backlog = 2.0(c) + 0.2(e)$$

.....

$$Backlog = 2.0(c) + 0.9(e)$$

Therefore, there are 9 possible optimal  $V_2$  for this method.

##### 4.3.2 Method Two - VCP

*Various Collision Parameters* method intends to obtain the optimal parameter in order to calculate and predict the closest number of remaining tags for the next identification round. Similarly to VEP method, we assume that for the current identification round, each collision slot has at least 2 tags collided. However, we cannot know for sure how many tags actually caused the collision. There is exactly 1 tag per successful slot, therefore, we do not take successful slot into consideration. In addition, empty slot does not engage any tag. Accordingly, we also do not take empty slot into consideration. VCP method focuses on finding optimal variable to calculate and predict the number of collision slot for the next identification round; and to find the impact of variable used on collision slots prediction.

The VCP method uses different parameter between 2.0 and 3.0 to predict the number of collision slot. Since collision slot engage at least 2 tags, we assume that the parameter for collision slot calculation falls between 2.0 and 3.0 (more than 2 but possibly less than 3). However, number of tags per collision slot can be more than 3 tags. According to Schoute's method (Schoute 1983), which has the best Backlog estimation, the parameter used is 2.39. Therefore, we

assumed that the optimal parameter falls between 2 and 3. Equation (2) shows Backlog estimation using variable  $2.0 \leq V_1 \leq 3.0$  for collision slot prediction.

$$\text{Backlog} = V_1(c) \dots \dots \dots (2)$$

where  $c$  = Collision Slot; and  $V_1$  = Variable between 2.0 and 3.0 incrementing by 0.1.

$$\text{Backlog} = 2.0(c)$$

$$\text{Backlog} = 2.1(c)$$

.....

$$\text{Backlog} = 3.0(c)$$

Therefore, there are 11 possible optimal  $V_1$  for this method.

#### 4.3.3 Method Three - VCEP

*Various Collision and Empty Parameters* method intends to obtain the optimal parameter in order to calculate and predict the closest number of remaining tags for the next identification round. In this method, we also assume that for the current identification round, each collision slot has at least 2 tags collided, but we do not know how many tags actually caused the collision. There is exactly 1 tag per successful slot, thus, we do not take successful slot into consideration. On the other hand, empty slot will continuously occur during the next rounds of identification. Therefore, VCEP method is created to find the optimal parameters to predict the number of both collision slot and empty slot for the upcoming round.

The VCEP method uses variable between 2.0 and 3.0 to predict the number of collision slot. Since collision slot engage at least 2 tags, we assume that the parameter for collision slot calculation falls between 2.0 and 3.0. Variable between 0.1 and 0.9 is also used to predict the number of empty slot. Since empty slot does not engage any tag, we assume that the parameter for empty slot calculation falls between 0.1 and 0.9. Equation (3) shows Backlog estimation using variable  $2.0 \leq V_1 \leq 3.0$  for collision slot prediction and variable  $0 < V_2 < 1$  for empty slot prediction.

$$\text{Backlog} = V_1(c) + V_2(e) \dots \dots \dots (3)$$

where  $c$  = Collision Slot;  $e$  = Empty slot;  $V_1$  = Variable between 2.0 and 3.0; and  $V_2$  = Variable between 0.1 and 0.9 incrementing by 0.1.

$$\text{Backlog} = 2.0(c) + 0.1(e)$$

$$\text{Backlog} = 2.0(c) + 0.2(e)$$

.....

$$\text{Backlog} = 2.0(c) + 0.9(e)$$

$$\text{Backlog} = 2.1(c) + 0.1(e)$$

.....

.....

$$\text{Backlog} = 3.0(c) + 0.8(e)$$

$$\text{Backlog} = 3.0(c) + 0.9(e)$$

Therefore, there are 99 possible optimal  $V_1 V_2$  for this method. Note that some parameters from VEP method are also included in this method.

Figure 3 shows 99 possible optimal  $V_1 V_2$  variables for  $c$  and  $e$ . The optimal  $V_1 V_2$  for this method will be different from optimal variable of VEP and VCP since  $V_1$  and  $V_2$  of VCEP have impact on each other.

#### 4.4 Sample Tag Allocation and Estimation

We are now initiating a sample tag allocation and estimation. Figure 4 shows a sample of first round tag allocation, where six collision slots, four empty slots, and six successful slots occurred. For each collision slot, two or more tags collided; while empty slot engaged no tag. Each successful slot holds exactly one tag per slot. After the first round of tag allocation, VEP, VCP, and VCEP equations are applied in order to find an estimated frame-size for the next identification round.

Slot type	s	e	s	c	c	e	s	e	c	s	c	e	c	s	c	s
Tag per slot	1	0	1	2	2	0	1	0	2	1	2	0	2	1	2	1

Figure 4: A sample First Round of Tag Allocation with Initial  $Q$  of 4. Collision slot  $c = 6$ , Empty slot  $e = 4$ , and Successful slot  $s = 6$ .

##### 4.4.1 Sample Estimation - VEP

After the first round of identification shown in figure 4, VEP method uses variable  $V_2$  between 0.1 to 0.9 to estimate empty slot and fixed  $V_1$  of 2.0 to estimate collision slot for the next round. After applying Equation (1), number of estimated tags for the next round are as shown in table 2 under round one ( $c = 6$ ,  $e = 4$ ). We can see that by using various parameters of  $V_2$ , different numbers of tags were predicted. For example, where  $V_2 = 0.2$ , the number of estimated tag can be calculated as follows:

$$\text{Backlog} = 2.0(6) + 0.2(4) = 12.8 \approx 13$$

Therefore, the estimated number of tags for the next round is equal to 13 tags. Hence, the new  $Q$  adjust is equal to 4 (see table 1).

Round one ( $c = 6$ , $e = 4$ )		
Variable ( $V_2$ )	Tag Estimation	Q Adjust
0.1	12	4
0.2	13	4
0.3	13	4
0.4	14	4
0.5	14	4
0.6	14	4
0.7	15	4
0.8	15	4
0.9	16	4
Round two ( $c = 3$ , $e = 6$ )		
Variable ( $V_2$ )	Tag Estimation	Q Adjust
0.1	7	3
0.2	7	3
0.3	8	3
0.4	8	4
0.5	9	4
0.6	10	4
0.7	10	4
0.8	11	4
0.9	11	4

Table 2: Sample tag estimation and Frame-size ( $Q$ ) adjustment using Method One - VEP.

Accordingly, after the first round, the adjustment of  $Q$  values using different parameters for tag estimation is equal to 4. In order to further demonstrate the estimation of frame-size, a sample of second round tag allocation is shown in figure 5.

$$\begin{pmatrix} 2.0c, 0.1e & 2.0c, 0.2e & 2.0c, 0.3e & 2.0c, 0.4e & 2.0c, 0.5e & 2.0c, 0.6e & 2.0c, 0.7e & 2.0c, 0.8e & 2.0c, 0.9e \\ 2.1c, 0.1e & 2.1c, 0.2e & 2.1c, 0.3e & 2.1c, 0.4e & 2.1c, 0.5e & 2.1c, 0.6e & 2.1c, 0.7e & 2.1c, 0.8e & 2.1c, 0.9e \\ 2.2c, 0.1e & 2.2c, 0.2e & 2.2c, 0.3e & 2.2c, 0.4e & 2.2c, 0.5e & 2.2c, 0.6e & 2.2c, 0.7e & 2.2c, 0.8e & 2.2c, 0.9e \\ 2.3c, 0.1e & 2.3c, 0.2e & 2.3c, 0.3e & 2.3c, 0.4e & 2.3c, 0.5e & 2.3c, 0.6e & 2.3c, 0.7e & 2.3c, 0.8e & 2.3c, 0.9e \\ 2.4c, 0.1e & 2.4c, 0.2e & 2.4c, 0.3e & 2.4c, 0.4e & 2.4c, 0.5e & 2.4c, 0.6e & 2.4c, 0.7e & 2.4c, 0.8e & 2.4c, 0.9e \\ 2.5c, 0.1e & 2.5c, 0.2e & 2.5c, 0.3e & 2.5c, 0.4e & 2.5c, 0.5e & 2.5c, 0.6e & 2.5c, 0.7e & 2.5c, 0.8e & 2.5c, 0.9e \\ 2.6c, 0.1e & 2.6c, 0.2e & 2.6c, 0.3e & 2.6c, 0.4e & 2.6c, 0.5e & 2.6c, 0.6e & 2.6c, 0.7e & 2.6c, 0.8e & 2.6c, 0.9e \\ 2.7c, 0.1e & 2.7c, 0.2e & 2.7c, 0.3e & 2.7c, 0.4e & 2.7c, 0.5e & 2.7c, 0.6e & 2.7c, 0.7e & 2.7c, 0.8e & 2.7c, 0.9e \\ 2.8c, 0.1e & 2.8c, 0.2e & 2.8c, 0.3e & 2.8c, 0.4e & 2.8c, 0.5e & 2.8c, 0.6e & 2.8c, 0.7e & 2.8c, 0.8e & 2.8c, 0.9e \\ 2.0c, 0.1e & 2.9c, 0.2e & 2.9c, 0.3e & 2.9c, 0.4e & 2.9c, 0.5e & 2.9c, 0.6e & 2.9c, 0.7e & 2.9c, 0.8e & 2.9c, 0.9e \\ 3.0c, 0.1e & 3.0c, 0.2e & 3.0c, 0.3e & 3.0c, 0.4e & 3.0c, 0.5e & 3.0c, 0.6e & 3.0c, 0.7e & 3.0c, 0.8e & 3.0c, 0.9e \end{pmatrix}$$

Figure 3: Variable  $V_1$  and  $V_2$  for Collision slot and Empty slot calculation in method three - VCEP. There are 99 possible combinations of  $V_1$  and  $V_2$  in order to find optimal parameters for  $c$  and  $e$  prediction.

Slot type	s	s	e	e	e	s	e	e	e	s	c	c	s	s	s	c
Tag per slot	1	1	0	0	0	1	0	0	0	1	2>	2>	1	1	1	2>

Figure 5: A sample second round of tag allocation where  $Q = 4$ . Collision slot  $c = 3$ , Empty slot  $e = 6$ , and Successful slot  $s = 7$ .

After the second round of identification shown in figure 5, VEP method again uses variable  $V_2$  between 0.1 to 0.9 to estimate empty slot and fixed  $V_1$  of 2.0, to estimate collision slot for the next round. After applying Equation (1), numbers of estimated tags for the next round are as shown in table 2 under round two ( $c = 3$ ,  $e = 6$ ). We can see that different numbers of tags were predicted and parameter  $Q$  is adjusted according to specific number of tags. For example, where  $V_2 = 0.7$ , the number of estimated tag can be calculated as follows:

$$\text{Backlog} = 2.0(3) + 0.7(6) = 10.2 \approx 10$$

Therefore, the estimated number of tags for the next round is equal to 10 tags. Hence, the new  $Q$  adjust is equal to 4 (see table 1).

Following the second round, the adjustment of  $Q$  value for parameters  $V_2 = 0.1 - 0.3$  is equal to 3, while the  $Q$  value for  $V_2 = 0.4 - 0.9$  is equal to 4. In order to identify all tags within the interrogation zone, VEP with fixed parameter  $V_1$  and variable  $V_2$  is applied for each identification round until no more collision occur and all tags are identified.

#### 4.4.2 Sample Estimation - VCP

After the first round of identification shown in figure 4, VCP method uses variable  $V_1$  between 2.0 to 3.0 to estimate collision slot. By applying Equation (2), numbers of estimated tags for the next round are shown in table 3. By using various parameters of  $V_1$ , different numbers of tags were predicted. For example, where  $V_1 = 2.8$ , the number of estimated tags can be calculated as follows:

$$\text{Backlog} = 2.8(6) = 16.8 \approx 17$$

Therefore, the estimated number of tags for the next round is equal to 17 tags. Hence, the new  $Q$  adjust is equal to 5 (see table 1).

Subsequently to the first round of identification, the adjustment of  $Q$  value for parameters  $V_1 = 2.0 - 2.7$  is equal to 4, while the  $Q$  value for  $V_1 = 2.8 - 3.0$  is equal to 5. In order to identify all tags within the interrogation zone, VCP with variable  $V_1$  is applied

for each identification round until no more collision occurs and all tags are identified.

Round one ( $c = 6$ )		
Variable ( $V_1$ )	Tag Estimation	Q Adjust
2.0	12	4
2.1	13	4
2.2	13	4
2.3	14	4
2.4	14	4
2.5	15	4
2.6	16	4
2.7	16	4
2.8	17	5
2.9	17	5
3.0	18	5

Table 3: Sample tag estimation and Frame-size ( $Q$ ) adjustment using method two - VCP.

#### 4.4.3 Sample Estimation - VCEP

Following the first round of identification shown in figure 4, VCEP method uses variable  $V_1$  between 2.0 to 3.0 to estimate collision slot and variable  $V_2$  between 0.1 to 0.9, to estimate empty slot for the next round. Also, after applying Equation (3), numbers of estimated tags for the next round are shown in table 4. Different numbers of tags are predicted depending on various parameters of  $V_1$  and  $V_2$ . For example, where  $V_1 = 2.5$  and  $V_2 = 0.5$ , the number of estimated tag can be calculated as follows:

$$\text{Backlog} = 2.5(6) + 0.5(4) = 17$$

Therefore, the estimated number of tags for the next round is equal to 17 tags. Hence, the new  $Q$  adjust is equal to 5 (see table 1).

After the first round, the adjustment of  $Q$  value where estimated tag is between 12 and 16 is equal to 4, while the  $Q$  value where estimated tag is between 17 and 22 is equal to 5. Likewise with VEP and VCP methods, VCEP with variable  $V_1$  and  $V_2$  is applied for each identification round until no more collision occurs, in order to identify all tags within the interrogation zone.

## 5 Experimental Evaluation

To measure a performance of "Precise Tag Estimation Scheme", we compare our results with performances of Schoute and Lowerbound methods. Different sets

Round one (c = 6, e = 4)																		
V <sub>2</sub>	0.1		0.2		0.3		0.4		0.5		0.6		0.7		0.8		0.9	
V <sub>1</sub>	Tag	Q	Tag	Q	Tag	Q	Tag	Q	Tag	Q	Tag	Q	Tag	Q	Tag	Q	Tag	Q
2.0	12	4	13	4	13	4	14	4	14	4	14	4	15	4	15	4	16	4
2.1	13	4	13	4	14	4	14	4	15	4	15	4	15	4	16	4	16	4
2.2	14	4	14	4	14	4	15	4	15	4	16	4	16	4	16	4	17	5
2.3	14	4	15	4	15	4	15	4	16	4	16	4	17	5	17	5	17	5
2.4	15	4	15	4	16	4	16	4	16	4	17	5	17	5	18	5	18	5
2.5	15	4	16	4	16	4	17	5	17	5	17	5	18	5	18	5	19	5
2.6	16	4	16	4	17	5	17	5	18	5	18	5	18	5	19	5	19	5
2.7	17	5	17	5	17	5	18	5	18	5	19	5	19	5	19	5	20	5
2.8	17	5	18	5	18	5	18	5	19	5	19	5	20	5	20	5	20	5
2.9	18	5	18	5	19	5	19	5	19	5	20	5	20	5	21	5	21	5
3.0	18	5	19	5	19	5	20	5	20	5	20	5	21	5	21	5	22	5

Table 4: A sample tag estimation and Frame-size (Q) adjustment using Method Three - VCEP.

of tags are considered and all three proposed algorithms (VEP, VCP, and VCEP) are implemented and applied for tag estimation.

Firstly, the reader picks tags within an interrogation zone by the command “SELECT”; then issues “QUERY” with ‘Q’ parameter to specify the frame-size. We have chosen the initial  $Q$  to be equal to 8. In the first round of identification, each selected tag will pick a random number between 0 to  $2^8 - 1$  and place it into its slot counter. The tag which picks zero as its slot number will respond to the reader. Secondly, the reader issues “QUERYREP” or “QUERYADJUST” command to initiate another slot. If the “QUERYADJUST” command is issued, one of the Precise Tag Estimation Scheme will be applied and the new  $Q$  (Frame-size) will be predicted for the next round. Finally, in order to identify all tags within the interrogation zone, either VCP, VCEP, or VCEP will be applied after each identification round until no more collision occurs and all tags are identified.

**Specification:** An Intel Core 2 CPU with 2.40GHz processor and 3GB RAM is used for testing. A Microsoft Window XP professional with Service Pack 3 is installed on the computer. Algorithms for tags simulation are implemented using Java JCreator.

### 5.1 Data Set

We conducted three experiments using different tag sets: 1) the performance comparison of VEP method versus Schoute method and Lowerbound method; 2) the performance comparison of VCP method versus Schoute method and Lowerbound method; and 3) the performance comparison of VCEP method versus Schoute method and Lowerbound method. While performing DFSA anti-collision algorithm using various Tag estimation methods, number of tags is supposedly unknown.

- **Experiment One:** The aim of experiment one is to find the impact of Empty slot on Backlog estimation.
- **Experiment Two:** The aim of experiment two is to find the impact of different parameters used to predict number of Collision slot.
- **Experiment Three:** The aim of experiment three is to find the optimal parameters for Backlog estimation and next Frame-size prediction.

Table 5 shows that in experiment one, two, and three, there are three methods applied on a tag set of 300 tags. We performed ten runs on the data set and present the average results. Numbers in bracket show number of variable used for that method.

300 tags		
Ex 1	Ex 2	Ex 3
VEP (9)	VCP (11)	VCEP (99)
Sch (1)	Sch (1)	Sch (1)
LB (1)	LB (1)	LB (1)

Table 5: Data set used in experiment one, two, and three. Sch = Schoute method and LB = Lowerbound method. There are total of 11 variables used in experiment one, 13 variables in experiment two, and 101 variables in experiment three.

## 5.2 Results

Results are compared using both ‘Frame’ and ‘Slot’. Since frame-size is dynamic in DFSA, each frame involves different number of slot allocation. Thus, the method that used fewer frames may have higher slot counters than the other methods. In this sub-section, we present and clarify all experimental results and provide analysis and discussions.

### 5.2.1 Experiment One Result

Based on the experiment, figure 6 shows an average of ten runs of: a) number of slots, and b) number of frames, compared between VEP method ( $V_1 = 2.0$ , variable  $0 < V_2 < 1$ ) versus Schoute (Sch) and Lowerbound (LB) methods. We can see that when  $V_2$  is between 0.1 and 0.2, VEP uses less total slots (collision, empty, and successful slot) than both Sch and LB. When  $V_2$  is between 0.3, 0.4, and 0.5, total number of slots of VEP method is slightly higher than Sch and LB. However, the number of frames used by VEP is lower. When  $V_2$  is higher than 0.5, number of slots increased rapidly.

From the result, we conclude that empty slots have impacted on the Backlog estimation and the next frame-size prediction. We can also summarise that by using fix parameter  $V_1 = 2$  and lower number for variable  $V_2$ , especially where  $V_2$  is between 0.1 and 0.5, the VEP can predict the number of Backlog more accurately.

### 5.2.2 Experiment Two Result

Figure 7 shows an average of ten runs of: a) number of slots, and b) number of frames, compared between VCP method (variable  $2.0 \leq V_1 \leq 3.0$ ) versus Sch and LB methods. VCP uses only one variable  $V_1$  to predict collision slot, where no empty slot was taken into consideration. By using VCP method, the results using different  $V_1$  are similar to each other, where only variable  $V_1 = 2.9$  and 3.0 have high number of slots used.



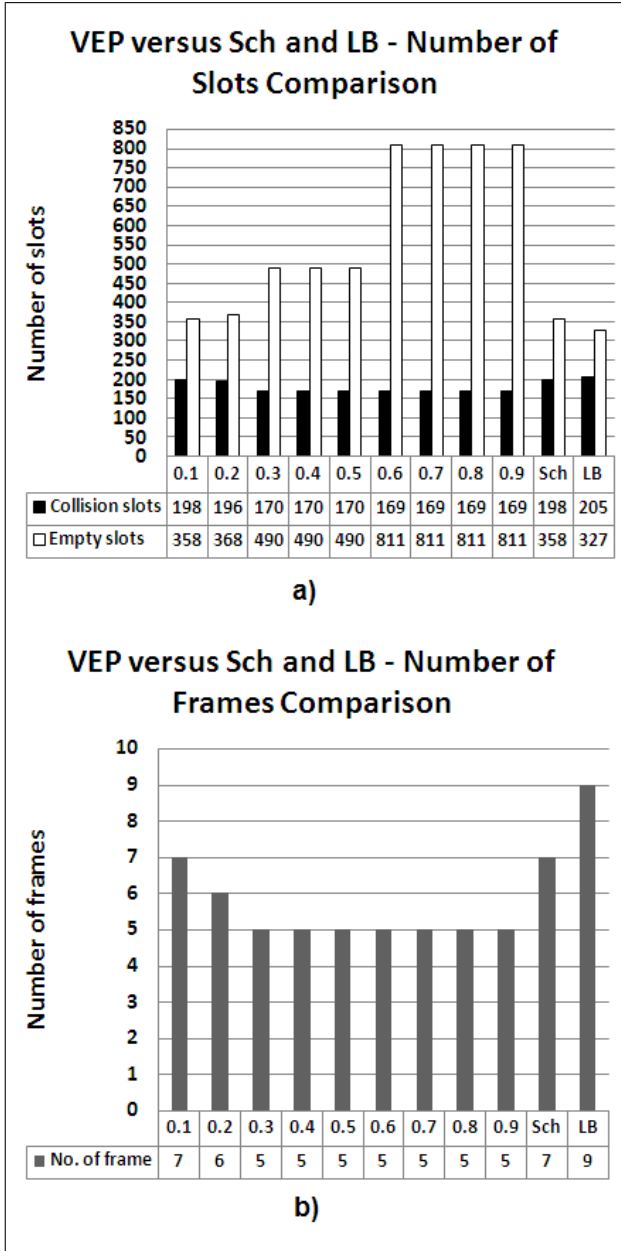


Figure 6: Comparison between VEP method with different variables versus Sch and LB methods: a) Number of slots, b) Number of frames.

From the result, we can conclude that variable  $V_1$  between 2 and 2.8 can be used to predict the number of collision slot efficiently. The result are in line with Schoute method where it indicated that 2.39 is an optimal value for  $V_1$ .

### 5.2.3 Experiment Three Result

Figure 8 shows an average of ten runs of: a) number of slots, and b) number of frames, compared between VCEP method (variable  $2.0 \leq V_1 \leq 3.0$ , variable  $0 < V_2 < 1$ ) versus Sch and LB methods. VCEP uses both variable  $V_1$  and  $V_2$  to predict collision slot and empty slot. VCEP method also includes variables used in VEP method, where  $V_1 = 2$  and  $0 < V_2 < 1$ . Due to the space availability, figure 8 only shows those results for variables that have better outcome compared to Sch and LB methods. From the figure, we can see there are thirty variables out of ninety-nine variables that used either less number of slots or number of frames compared to Sch and LB methods.

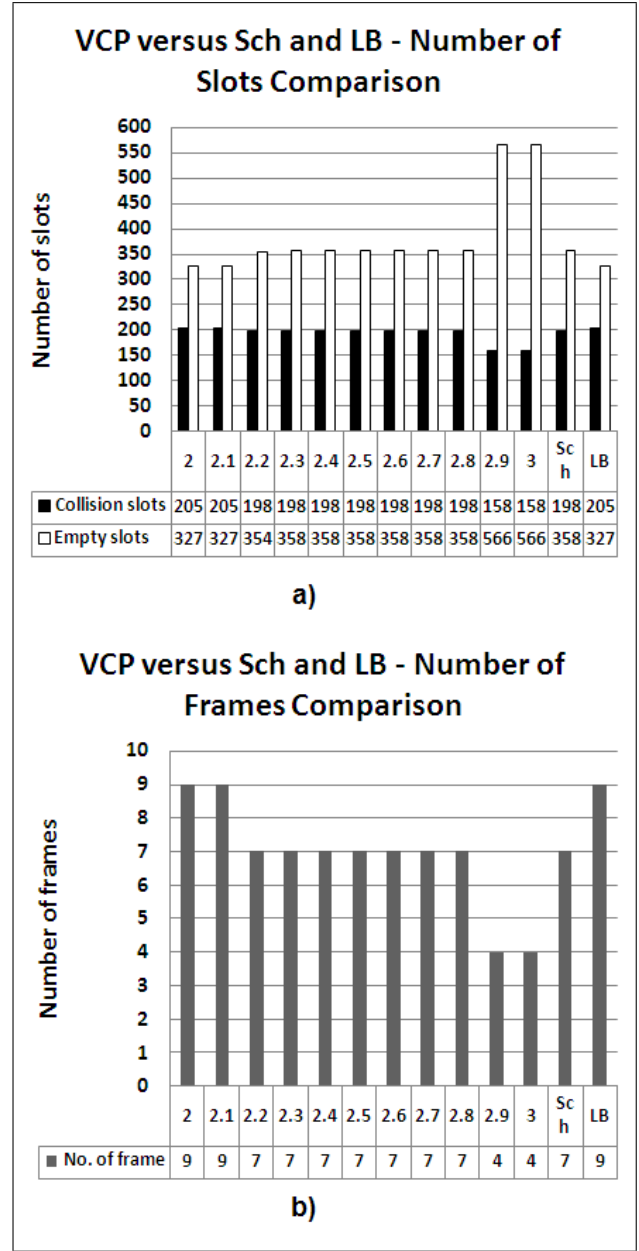


Figure 7: Comparison between VCP method with different variables versus Sch and LB methods: a) Number of slots, b) Number of frames.

There are four major variables with visible improvement in total slots and frame counters, these variables are: 1)  $V_1 = 2$ ,  $V_2 = 0.1$ , 2)  $V_1 = 2$ ,  $V_2 = 0.2$ , 3)  $V_1 = 2.1$ ,  $V_2 = 0.1$ , and 4)  $V_1 = 2.2$ ,  $V_2 = 0.1$ . From the result we conclude that, any variable  $V_1$  between 2 and 2.2 and variable  $V_2$  between 0.1 and 0.2, can be used as an efficient Backlog estimation.

### 5.3 Analysis and Discussion

This sub-section compares and analyses all results from the three experiments. Figure 9 shows that different variables have impacted on total number of slots and frames. When only one variable  $V_1$  is used to predict the number of collision slot (VCP), any variable between 2 and 2.8 is suitable (Shown as 'White' area in figure 9). On the other hand, when variable  $V_2$  is taken into consideration, only few variables give the best results, these variables are 1)  $V_1 = 2$ ,  $V_2 = 0.1$ , 2)  $V_1 = 2$ ,  $V_2 = 0.2$ , 3)  $V_1 = 2.1$ ,  $V_2 = 0.1$ , and 4)  $V_1 = 2.2$ ,  $V_2 = 0.1$ . Any other lower

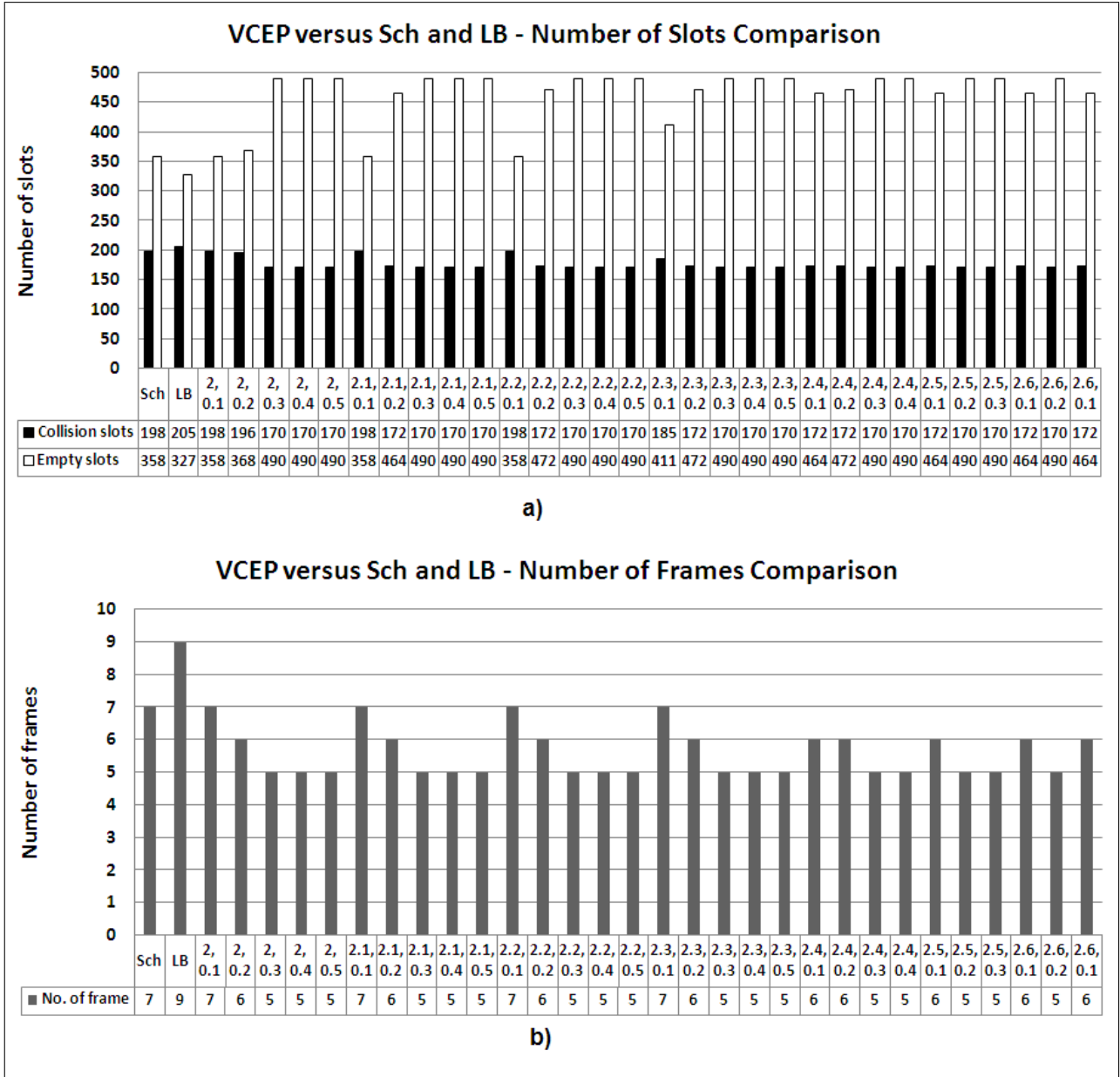


Figure 8: Comparison between VCEP method with different variables versus Sch and LB methods: a) Number of slots, b) Number of frames.

variables give various results; some with less number of slots and some with less number of frames compared to Sch and LB methods (Shown as 'Grey' area in figure 9). The 'Black' area in figure 9 represents those variables with poor results where total number of slots and frames are higher than Sch and LB.

The analysis clarifies that there are more than one suitable variables to predict Backlog; and that these parameters involve both collision slots and empty slots. From the analysis and discussion, we conclude that the DFSA performed more efficiently compared to Sch and LB when the parameter  $V_1$  and  $V_2$  used are low numbers. In order to minimise number of frames and slots, the variable  $V_1$  should be between 2 to 2.8 for VCP method; the variable  $V_2$  should be between 0.1 to 0.5 for VEP method; and the variable  $V_1$  should be between 2 to 2.7; and the variable  $V_2$  should be between 0.1 to 0.5 for VCEP method. However, total number of slots and frames also depends on the combination of the two parameters for VCEP (including VEP) method. Both parameters should be one low and one high or both low, to optimise the scheme.

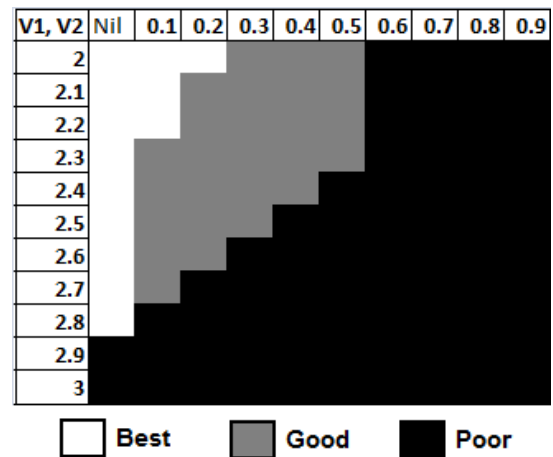


Figure 9: Result matrix of VEP, VCP, and VCEP methods using different variables compared to Schoute and Lowerbound methods.

## 6 Conclusion

In this work, we investigated the significance of RFID tags anti-collisions and developed efficient tag estimation method to improve the system efficiency of DFSA. We proposed a “Precise Tag Estimation Scheme”, which estimate precise number of tags around the reader.

The results and analysis have indicated that various parameter used by “Precise Tag Estimation Scheme”, including *empty slots* variables and/or *collision slots* variables, have an impact on system efficiency. Hence, variables used in VEP, VCP, and VCEP methods can change number of frames and slots, and the smaller values of both parameters  $V_1$  and  $V_2$  have a better impact on system efficiency.

In terms of future work, we intend to further test “Precise Tag Estimation Scheme” on different data sets with different number of tags. Different initial  $Q$  parameter will also be investigated to determine the impact on number of total frames and slots.

## Acknowledgements

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

## References

- Abramson, N. (1970), The ALOHA System - Another Alternative for Computer Communications, in ‘Proceedings of Fall Joint Computer Conference, AFIPS Conference’, Houston, Texas, pp. 281–285.
- Brown, M., Patadia, S. & Dua, S. (2007), *Mike Meyers’ Certification Passport: CompTIA RFID+ Certification*, McGraw-Hill.
- Cha, J. R. & Kim, J. H. (2005), Novel Anti-collision Algorithms for Fast Object Identification in RFID System, in ‘ICPADS ’05: Proceedings of the 11th International Conference on Parallel and Distributed Systems - Workshops’, IEEE Computer Society, Washington, DC, USA, pp. 63–67.
- Chen, W. T. (2006), ‘An Efficient Anti-Collision Method for Tag Identification in a RFID System’, *IEICE Transactions* **89-B**(12), 3386–3392.
- Cho, H., Lee, W. & Baek, Y. (2007), LDFSA: A Learning-Based Dynamic Framed Slotted ALOHA for Collision Arbitration in Active RFID Systems, in ‘Advances in Grid and Pervasive Computing Second International Conference’, Vol. 4459, Springer Berlin/Heidelberg, Paris, France, pp. 655–665.
- 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](http://www.freshpatents.com/Query-tree-based-tag-identification-method-in-rfid-systems-dt20080508ptan20080106383.php).
- EPCGlobal (2006), ‘EPCGlobal Tag Data Standards Version 1.3: Ratified Specification’. <http://www.epcglobalinc.org/standards/tds/>.
- Floerkemeier, C. (2007), Bayesian Transmission Strategy for Framed ALOHA Based RFID Protocols, in ‘RFID, 2007. IEEE International Conference on RFID Gaylord Texan Resort’, Grapevine, TX, USA, pp. 228–235.
- Jain, S. & Das, S. R. (2006), Collision avoidance in a dense RFID network, in ‘WiNTECH ’06: Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization’, ACM, New York, NY, USA, pp. 49–56.
- Landt, J. (2001), *Shrouds of Time The history of RFID*, An Aim Publication, Pittsburg, PA.
- Lee, C. W., Cho, H. & Kim, S. W. (2008), ‘An Adaptive RFID Anti-Collision Algorithm Based on Dynamic Framed ALOHA’, *IEICE Transactions* **91-B**(2), 641–645.
- Lee, S. R., Joo, S. D. & Lee, C. W. (2005), An enhanced dynamic framed slotted aloha algorithm for rfid tag identification, in ‘MOBIQUITOUS ’05: Proceedings of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services’, IEEE Computer Society, Washington, DC, USA, pp. 166–174.
- Li, B., Yang, Y. & Wang, J. (2009), ‘Anti-collision Issue Analysis in Gen2 Protocol - Anti-collision issue analysis considering capture effect’, *Auto-ID Labs White Paper*. <http://www.autoidlabs.org/single-view/dir/article/6/320/page.html>.
- Myung, J. & Lee, W. (2006a), ‘Adaptive binary splitting: a RFID tag collision arbitration protocol for tag identification’, *Mob. Netw. Appl.* **11**(5), 711–722.
- Myung, J. & Lee, W. (2006b), 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, Seoul, Korea, 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.
- Schoute, F. C. (1983), ‘Dynamic Frame Length ALOHA’, *IEEE Transactions on Communications* **31**(4), 565–568.
- 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, Beijing, China, pp. 693–700.
- Vogt, H. (2002), Efficient Object Identification with Passive RFID Tags, in ‘Pervasive ’02: Proceedings of the First International Conference on Pervasive Computing’, Springer-Verlag, London, UK, pp. 98–113.
- Wang, Z., Liu, D., Zhou, X., Tan, X., Wang, J. & Min, H. (2007), ‘Anti-collision Scheme Analysis of RFID System’, *Auto-ID Labs White Paper*. <http://www.autoidlabs.org/single-view/dir/article/6/281/page.html>.



# Scalable Online Index Construction with Multi-core CPUs

Hiroyuki Yamada

Motomichi Toyama

Department of Information and Computer Science  
Keio University, Yokohama, Japan  
Email: hiroyuki@db.ics.keio.ac.jp,  
toyama@ics.keio.ac.jp

## Abstract

Inverted index is a core element of current text retrieval systems. They can be dynamically constructed using online indexing approaches in the environment which even a small delay in timeliness cannot be tolerated, and the index must always be queryable and up to date. Recently, efficient online index construction schemes have been proposed, however, previous works have not focused on scalability with the modern commodity hardware resources such as multi-core CPUs. In this paper, we propose a scalable online index construction method that better utilizes multi-core CPUs. Using experiments on 30 GB of web data, we demonstrate the efficiency of our method in practice, showing that it dramatically reduces online index construction time without sacrificing query performance.

**Keywords:** Information Retrieval, Text Databases, Inverted Index, Online Index Construction, Index Maintenance

## 1 Introduction

Inverted index is a core element of current text retrieval systems. It is a data structure that maps a word, or atomic search item, to the set of documents, or set of indexed units, that contain that word - its postings. An individual posting may be a binary indication of the presence of that word in a document, or may contain additional information, such as its frequency in that document and an offset for each occurrence, required for various non-boolean search algorithms. Inverted index can be constructed using either off-line approaches or online approaches. In off-line approaches, one or more passes are made over a static set of input data and, at the completion of the process, an index is available for querying. On the other hand, online approaches are used when even a small delay in timeliness cannot be tolerated and the index must always be queryable and up to date. In the current search environment, online approaches are more general and there have been various methods proposed recently.

Basic techniques to support document insertions into an existing index usually follow a standard scheme: two indexes are maintained, one in memory, the other on disk. Postings for new documents are accumulated in main memory until it is exhausted, at which point they are transferred to disk and combined with existing on-disk data. This operation can be performed by an in-place update scheme (17) or by merging the old index with the new data, resulting in a new index that supersedes the old one (9). More efficient merging techniques are proposed (3; 10), which allows a controlled number of on-disk indexes to increase indexing performance without decreasing query processing performance excessively.

In recent years, commodity hardware have much more resources than few years ago. Regarding CPUs, major vendors currently have dual-core and quad-core CPUs, and some vendor has an eight-core CPU. Also, a lot of memory is available in 64 bit operating systems. On the current desktop/server machine, dual-core CPU and 4 GB memory is a very common setting and multiple dual/quad-cores with more main memory is often used in servers. However in this situation, previous online index construction methods are not designed to scale for such modern hardware resources, that is, index construction time does not reduce as the number of available cores increases even though scalability is one of the most important factor in current database systems.

In this paper, we propose a scalable online index construction method which better utilizes multi-core CPUs. The principle of our new method is that multiple in-memory indexes are parallelly constructed by multiple cores, and aims to reduce the total construction time. To the best of our knowledge, this is the first study focused on exploiting multi-core CPUs in online index construction. Our experiments show that online index construction using this approach significantly reduces index construction time without sacrificing query performance.

The rest of this paper is organized as follows: we begin by describing prior work on online index construction in Section 2. In Section 3, we provide new inverted index data structure which in-memory indexes can be parallelly constructed with, and introduce some optimization techniques to get better performance. Section 4 provides experimental results and their evaluation and we give a discussion about the proposed method and related work in Section 5. Finally, Section 6 gives the conclusion and future work.

## 2 Background

### 2.1 Inverted Index Structures

Inverted index contains two main parts: a vocabulary, listing all the terms that appear in the document

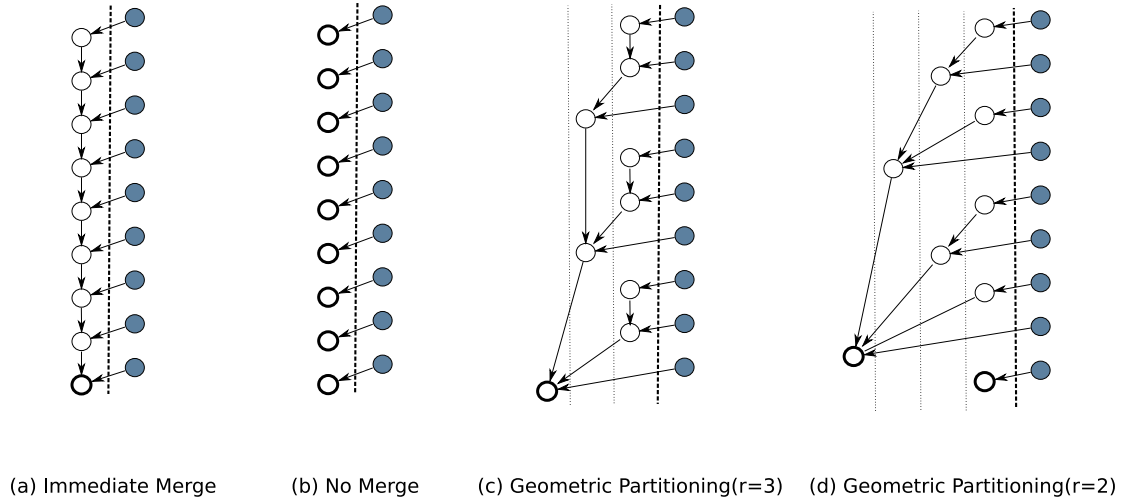


Figure 1: Merging methods in online index construction. The colored node is the sub-index generated by in-memory index. The thick-line node is the final sub-index.

collection; and a set of inverted lists, one per term. Each inverted list contains a sequence of postings (also sometimes known as pointers), together with a range of ancillary information, which can include within-document frequencies and a subsidiary list of positions within each document at which that term appears (12). A range of compression techniques have been developed for inverted lists (1; 13; 18; 12; 20), and, even if an index contains word positional information, it can typically be stored in around 25% of the space occupied by the original text. Index compression also reduces the time required for query evaluation.

The standard form of inverted index stores the postings in each inverted list in document ID order, and is referred to as being document sorted. It is the index organization that we consider in this paper. Other index orderings are frequency sorted and impact sorted. None of the online mechanisms we describe in this paper apply to these other forms of index organization.

A retrieval system processes queries by examining the postings for the query terms, and using them to calculate a similarity score that estimates the likelihood that the document matches the query. This process requires that all terms in the collection be indexed, with the exception of a small number of common terms that carry little information (stop-words). Phrase queries can also be resolved via the index if it contains word positions. In this case the retrieval system treats each phrase in the query as a term, and infers an inverted list for it by combining the lists for the component terms. Stop-words also need to be indexed for phrase queries to be efficiently resolved.

Inverted lists are typically stored on disk in a single contiguous extent or partition, meaning that once the vocabulary has been consulted, one disk seek and one disk read is required per query term.

The addition of a further document to an existing inverted index adds a new document pointer to a large number - potentially thousands - of inverted lists. Seeking on disk for each list update would be catastrophic, and in practical systems the disk costs are amortized across a series of updates. To this end, the inverted index in a dynamic retrieval system is stored in two parts: an in-memory component that provides an index for recently inserted documents; and an on-disk component, which is periodically combined with the in-memory part of the index in a merging event, and then written back to disk. This approach is effective

because a typical series of documents has many common terms. The disk-based merging process can be sequential rather than random-access and all of the random-access operations can take place in main memory. Also, documents are searchable as soon as they are inserted. However, querying is now more complex, since the in-memory part of each term's inverted list must be logically combined with the on-disk part.

## 2.2 Index Merging

As described in the previous section, various merging techniques between in-memory indexes and on-disk indexes have been proposed. Figure 1 shows the major merging methods, which are described in the following subsections.

### 2.2.1 Immediate Merge

The first merge strategy has been proposed by Lester et al. (9). The indexing system maintains one on-disk and one in-memory index. As soon as main memory is full, the in-memory postings are merged with the existing on-disk index, creating a new index. The old index is deleted. This strategy minimizes the number of disk seeks necessary to fetch a posting list. Its disadvantage is that for every merge operation the entire index has to be scanned. Thus, the number of disk operations necessary to index the whole collection is quadratic in the size of the text collection.

### 2.2.2 No Merge

The second strategy does not perform any merge operations. When memory is full, postings are sorted and written to disk, creating a new on-disk sub-index. On-disk indexes are never merged. When the posting list for a given term has to be retrieved from the index, sub-lists are fetched from all sub-indexes. The advantage of No Merge is its high indexing performance (linear number of disk operations). Its disadvantage is that fetching a posting list requires  $O(n)$  disk seeks, where  $n$  is the size of the text collection.

### 2.2.3 Geometric Partitioning

The two strategies described so far represent the two extremes. The third strategy is a compromise: a

newly created on-disk sub-index is sometimes merged with an existing one, but not always.

Lester et al. (10) propose a scheme that breaks the index into tightly controlled number of partitions. They introduce a key parameter  $r$  (usually,  $r = 2$  or  $r = 3$  is chosen). If main memory can hold  $b$  postings, then the  $k$ th partition contains not more than  $(r - 1)r^{k-1}b$  postings. In addition, at level  $k$  the partition is either empty, or contain at least  $r^{k-1}b$  postings. Whenever the creation of a new on-disk index leads to a situation where there are more than  $(r-1)r^{k-1}b$  postings in  $k$ th partition, they are merged into a new index and dispatched to the appropriate partition. This strategy is referred to as Geometric Partitioning. Lester et al. (10) gives a cost analysis of this strategy. Büttcher and Clarke (3) also propose a similar strategy referred to as Logarithmic Merge and it is regarded as 'r=2 Geometric Partitioning' (11).

Geometric Partitioning and Logarithmic Merge strategies are designed for growing text collections and offer better index maintenance performance than Immediate Merge. The disadvantage is that query processing performance is worse, as posting list of each term may spread across a number of on-disk sub-indexes.

### 2.2.4 Other Strategies

In-place update strategy involves minimizing the change to the index at each stage by, whenever possible, writing new postings at the end of the existing lists, and related works have been broadly researched in (5; 2; 8; 14; 15). Büttcher et al. (4) proposed hybrid strategy based on a distinction between short and long posting lists, which maintains short posting lists following a merge-based approach, while long lists are updated in-place. Also, Guo et al. (6) proposed another merge strategy that can dynamically adjust the sequence of sub-index merge operations during index construction, and offers better query processing performance than previous methods with support for instantaneous document deletions.

## 3 Parallel Construction of Multiple In-memory Indexes

We have discussed the index update strategies for continuously growing text collections in the previous section. Indexing with Geometric Partitioning is very efficient in terms of disk I/O and practically one of the best option in a current dynamic environment. However, in-memory part is still time-consuming with in-memory index construction, which includes parsing, index compression and in-memory inversion. Also, it has no focus on scalability with modern hardware resources, that is, its performance does not increase as hardware resource increases.

To deal with the issue, we focus on Geometric Partitioning with multi-core CPUs and propose a new method to achieve scalable online index construction. We first propose a novel data structure to achieve parallel construction of multiple in-memory indexes, and then, introduce some optimization techniques enhancing its performance. In summary, our main contributions are as follows:

1. We propose a novel data structure of inverted index which enables parallel construction of multiple in-memory indexes. Also, we present an intersection algorithm for the proposed data structure briefly.
2. We propose some optimization techniques called Out-of-Order Merging and Lazy Merging, which enhance the parallel construction performance.

3. We present the experimental results on a crawled 30 GB web-based document collection and show that the proposed method dramatically reduces online index construction time without sacrificing query performance.

### 3.1 Multiple In-memory Indexes

In order to achieve scalable index construction with multi-core CPUs, constructing multiple in-memory indexes parallelly by cores seems to be a good approach. This approach divides an time-consuming in-memory index construction process into the number of cores available. Figure 2 shows parallel construction of multiple in-memory indexes and on-disk merging in a timeline view. As shown in Figure 2, each in-memory process constructs each index and merging occurs when in-memory process completes its construction. However, if using a naive approach, parallel construction of multiple in-memory indexes has difficulty in managing document sorted inverted lists. In this paper, we propose a solution for this problem.

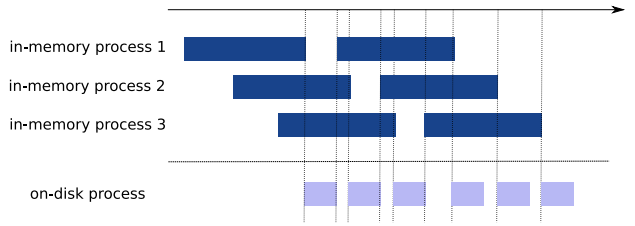


Figure 2: Parallel construction of multiple in-memory indexes and on-disk merging in a timeline view.

### 3.2 Inverted Index with Two-level Document ID

To deal with the problem described above, we propose a novel inverted index data structure with two-level document ID, which manages document IDs in inverted lists in two level with Global Sequence Number (GSN) and local document ID. With this scheme, in-memory processes can independently proceed with local document ID as usual way as long as an unique GSN is assigned to those in-memory indexes. Also, by adding the GSN to the beginning of the local document IDs, finally created inverted lists are ordered globally by GSN and locally by local document ID as shown in Figure 3. One thing to keep in mind when dealing with parallel in-memory processes is that the merge operation must be coordinated between the processes not to proceed concurrently, so that inverted lists are properly created in ascending GSN order.

To meet the memory requirement, the specified memory is split into the number of in-memory processes. For example, to make 8 in-memory processes parallelly runnable with 1 GB memory as a buffer, 128 MB memory is assigned to each in-memory process.

### 3.3 Out-of-Order Merging

By assigning a GSN to an in-memory index just before merging, merging can be done in an out-of-order manner and this gives better performance. That is, there is no order between in-memory indexes. As soon as any in-memory index exceeds a specified threshold, merging for the index can be executed immediately without any stall when no other processes are

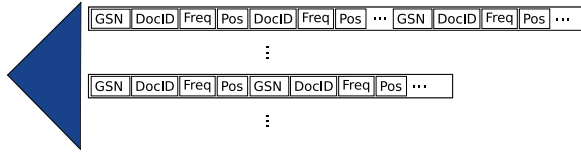


Figure 3: Inverted lists with two-level document ID. Each inverted list is globally in an GSN order and locally in an local document ID (specified as DocID) order. There could be multiple DocIDs between GSNs, which depends on term frequency and document frequency.

in merging. Here we describe the basic steps for each process in constructing inverted index with Out-of-Order Merging.

1. For each in-memory process,
  - (a) Postings for new documents are accumulated in the in-memory index.
  - (b) When the in-memory index size reaches to the threshold,
    - i. Assign a GSN to the in-memory index.
    - ii. Merge the in-memory index with the existing on-disk indexes. (wait while another index is in merging.)
  - (c) After the merging finishes, go back to step 1a.

### 3.4 Lazy Merging

As we described, each process has its own split buffer for in-memory index. Its size is relatively small compared with the buffer size in a single in-memory process, so frequent merging is avoidable and incurring a lot of disk I/O as a result. We propose an optimization technique called Lazy Merging for this issue. It prepares a queue first and puts the in-memory index to the queue as soon as it exceeds the buffer size. Then the process can proceed another construction of in-memory index with a new buffer. When the queue becomes full, then flushing proceeds with multi-way merging the queued in-memory buffers with existing on-disk indexes. This method avoids the frequent small sized merging and saves a lot of disk I/O. Figure 4 shows conceptual view of Lazy Merging. It is actually implemented more efficiently without memory copy and consumption of unused buffers. It is also a little similar to the well-known 'double buffering' technique, but our method defers I/O operation and merging occurs in batch afterwards, and buffers and in-memory processes are independent so in-memory process proceeds index construction as long as there is an available buffer in the pool.

To meet the memory requirement, we split the memory into the number of buffers being used, so that this works efficiently without extra memory consumption. From our experiments, when we have  $n$  in-memory processes, index buffer should be split into  $2n$  and size of the queue should be set in  $n$  to achieve better performance. For example, When we have 1 GB buffer available for 8 in-memory processes, the buffer had better to be split into 16 and buffer queue is set in size 8. The following describes details about the index construction steps with Lazy Merging.

1. Prepare an empty queue for in-memory index

2. For each in-memory process,

- (a) Postings for new documents are accumulated in in-memory index.
- (b) When the in-memory index size reaches to the threshold, assign a GSN to the in-memory index and push it into the queue.
  - i. If the queue is not full, go back to step 2a.
  - ii. If the queue is full, go to next step.
- (c) Multi-way merge the queued indexes with the existing on-disk indexes.
- (d) After merging finishes, the queued buffers are cleared and accept new buffers to be pushed, and go back to step 2a.

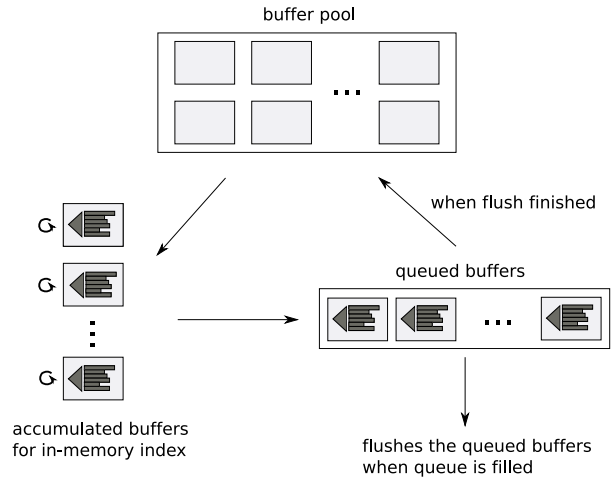


Figure 4: Conceptual view of Lazy Merging. The queued buffers are being multi-way merged to create a new index when the queue is filled.

### 3.5 Intersection Algorithm

Intersection algorithm in querying, such as in boolean queries and phrase queries, must be modified to adapt the inverted index with two-level document ID created by the proposed method. Algorithm 1 shows pseudo intersection algorithm between two posting lists. The main difference is that GSN checking operation is added before document ID checking operation. (to make it easy to understand, checking positions is not included in this pseudo code.)

## 4 Experiments

We have experimented with a crawled 30 GB web-based document collection, which consists of 22.7 million documents, and measured the index construction time and the query time. All experiments are performed on linux based on a dual-processor Quad-Core Xeon 5345 machine with 7,200-rpm SATA hard drive, and it has 8 GB of RAM with no significant other processes running at the time of the experiments. We adjusted the number of enabled cores by setting `/sys/devices/system/cpu/cpu[CPU ID]/online` parameter through the linux file system. In this experiments, we focused on Geometric Partitioning ( $r=2$ ) as an online index construction method. Inverted lists include term frequency and positional indexes for each document ID and document IDs and



**Algorithm 1** INTERSECT( $p1, p2$ )

---

```

1:  $answer \leftarrow \langle \rangle$ 
2: while  $p1 \neq \text{NIL}$  and  $p2 \neq \text{NIL}$  do
3:   if  $GSN(p1) = GSN(p2)$  then
4:      $ADD(answer, GSN(p1))$ 
5:     while  $hasDoc(p1)$  and  $hasDoc(p2)$  do
6:       if  $docID(p1) = docID(p2)$  then
7:          $ADD(answer, docID(p1))$ 
8:          $p1 \leftarrow nextDoc(p1)$ 
9:          $p2 \leftarrow nextDoc(p2)$ 
10:      else
11:        if  $docID(p1) < docID(p2)$  then
12:           $p1 \leftarrow nextDoc(p1)$ 
13:        else
14:           $p2 \leftarrow nextDoc(p2)$ 
15:        end if
16:      end if
17:    end while
18:     $p1 \leftarrow nextGSN(p1)$ 
19:     $p2 \leftarrow nextGSN(p2)$ 
20:  else
21:    if  $GSN(p1) < GSN(p2)$  then
22:       $p1 \leftarrow nextGSN(p1)$ 
23:    else
24:       $p2 \leftarrow nextGSN(p2)$ 
25:    end if
26:  end if
27: end while

```

---

positional indexes are stored as the differences from the previous number. Also, GSN gaps are taken in inverted index with two-level document ID. They are also compressed by variable-byte codes (12; 20), which is a well-known and very effective compression method adopted in many search engines. The elapsed times are presented for index construction including all parsing, index compression, indexing, and list merging phases. The query time includes searching vocabularies, fetching lists from the disk and lists intersection, but doesn't include other processes such as scoring results, fetching documents and snippet generation, because those processes are not affected by the proposed data structure.

We have implemented the system from scratch in C++. We used `std::map` in STL for the in-memory index and Berkeley DB-like original B+-tree database called Lux IO<sup>1</sup> for the on-disk index. Parallel in-memory processes are implemented with POSIX threads and the number of in-memory processes are the same as the number of enabled cores in these experiments. In this system, the documents to be indexed are queued and each in-memory process fetches one document at a time when processable, and constructed in-memory indexes are merged to create on-disk indexes as described previous section. The system overview is shown in Figure 5.

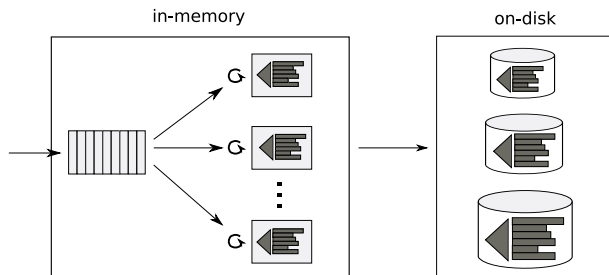


Figure 5: System overview with multiple in-memory indexes.

The specified buffer for indexing counts the

<sup>1</sup><http://luxio.sourceforge.net/> - Yet Another Fast Database Manager.

amount consumed for actual data of vocabularies and postings. That is, it doesn't include library dependent data such as internal structures managed by the library. So indexing with 1 GB buffer actually uses more memory than specified. Also, we applied Geometric Partitioning to both vocabularies and postings, which is described in (11) as a more scalable indexing method.

Figure 6 shows the index construction time by our proposed method, using 1 GB buffer with Out-of-Order Merging for the 30 GB document collection, for the range of the number of enabled cores. There are three lines, one at the top shows the construction time with the existing method and the other two at the bottom show the construction time with the proposed method with or without Lazy Merging. These results indicate that the proposed method reduces the construction time dramatically as the number of enabled cores increases, even though the existing method has no effect on the construction time with 8 cores enabled. The reduction rate for the proposed method degrades gradually, but Lazy Merging technique relax this degradation and additional performance gain is obtained as expected. This degradation is considered to be due to stall of threads, which finish in-memory part and wait for merging event to finish. Thus Lazy Merging, which reduces disk I/O incurred with merging, becomes a more effective technique as the number of enabled cores increases.

Figure 7 shows the query time with two-term phrase queries by a single thread for the inverted index with two-level document ID, which is created by 8 threads, and conventional inverted index. It is the sum of the query time for all existing on-disk indexes. It depends on the existing number of indexes at the time experiments performed, so we measured the query time for the range of the size of documents indexed. We did not measure the query time for in-memory indexes in these experiments because it is relatively small compared with the query time for on-disk indexes and thus ignorable. And also, we measured three types of queries, highly frequent, moderately frequent and less frequent, for the index of 30 GB data. This is because that the posting list of a less frequent query is likely to be more occupied by GSNs, and this might affect the query performance badly. Figure 7 indicates that inverted index with two-level document ID does not slow down the query time, and the size of the inverted lists created by this scheme is almost the same as the size of the conventional inverted lists as shown in Table 1. We believe the reason for this is the followings. The document IDs between GSNs are relatively smaller than the document IDs assigned with the usual way and the gaps between document IDs become smaller. Also, the gaps between GSNs usually do not become too large. So, the compression methods such as variable-byte codes work effectively for those small numbers, and thus the size of the proposed inverted lists does not become much larger.

## 5 Discussion

In this paper, we focused on an online index construction method with multi-core CPUs and single disk. Applying this method to a machine with multiple disks and getting more scalable index construction is going to be a next challenge.

Utilizing multiple cores/CPU and multiple disks can also be considered in parallel information retrieval (7; 16; 19). It can also be distributed information retrieval with virtual hosts by virtualization technology. So, a machine with multiple cores/CPU and multiple disks is regarded as multiple processors or

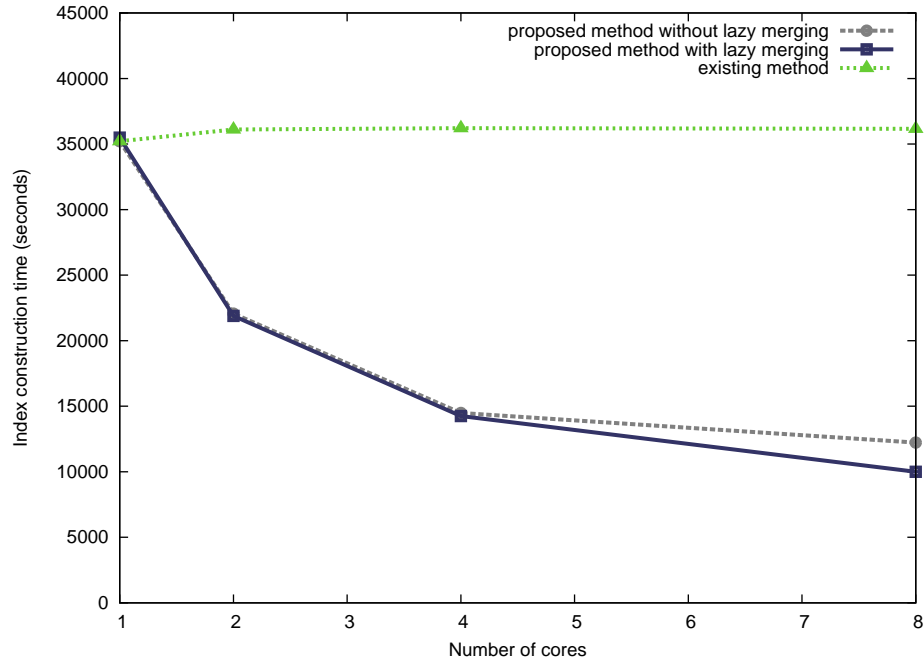


Figure 6: Index construction time for the 30 GB collection on dual processor Quad-Core Xeon 5345 with 1 GB memory buffer, for the range of the number of enabled cores (1,2,4,8).

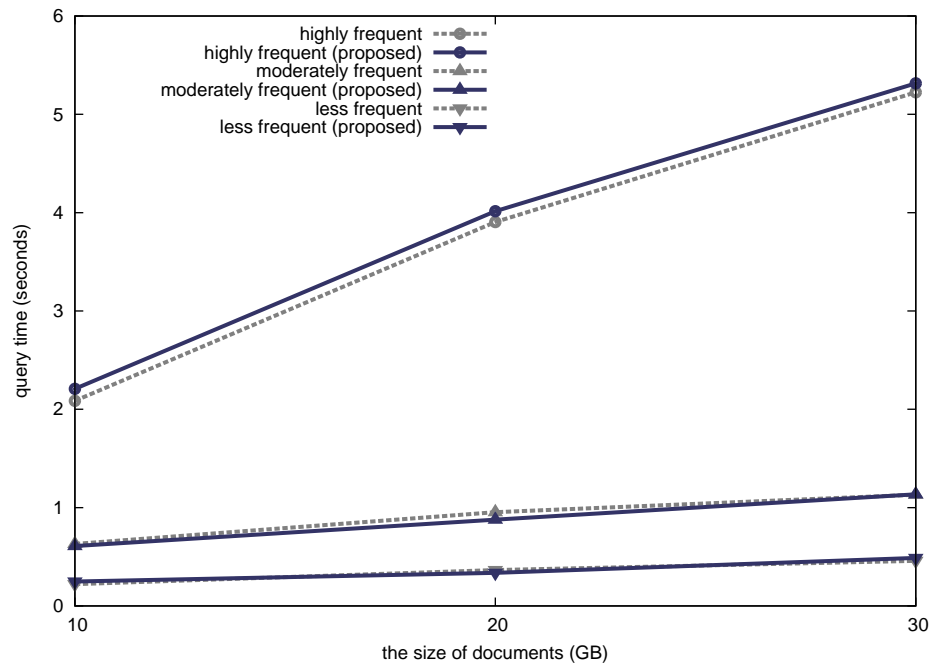


Figure 7: Query time with two-term phrase queries by a single thread for inverted index with two-level document ID (with GSN), which is created by 8 threads, and conventional inverted index, for the range of the size of documents indexed.

	size of inverted lists with two-level document ID	size of conventional inverted lists	increase rate
highly frequent	184307990 (bytes)	184306856 (bytes)	0.00062 (%)
moderately frequent	34019764 (bytes)	34018979 (bytes)	0.0023 (%)
less frequent	13444111 (bytes)	13441946 (bytes)	0.016 (%)

Table 1: Size of inverted lists of selected phrase queries which created by two-level document ID scheme and by a conventional scheme for 30 GB document collection.

multiple hosts, and document partitioning technique can be applied to get better scalability. The method described in this paper still can be applied in each processor which likely has multiple cores (2-4 cores) to handle a lot of operations such as querying and accepting TCP connections when indexes are owned by server processes for instance.

Document partitioned inverted indexes can be constructed independently using the split resources of modern hardware as described above, however, independent online index construction increases the number of indexes per CPU. For example, let's consider that we have a machine with Quad-Core CPU and 4 disks and 100 GB data to be indexed with 1 GB memory available for a buffer. We can have 4 logically independent processors with 1 core and 1 disk each. In this setting, indexing with Geometric Partitioning is parallelly proceeded in each processor for 25 GB data with 256 MB memory buffer, and the index construction time can be reduced by factor of 4 compared with the time by a single core. However, the number of indexes created is also 4 times larger ( $4 * \log(25G/256MB) = 28$ ), and this degrades query performance excessively in exchange for reduction of index construction time.

We believe that applying the proposed method to multiple disks and creating the same number of indexes as a single processor does by collaborating on-disk merging could be another interesting research area in online index construction with modern hardware. Of course, that kind of machine can be clustered, and document partitioned technique is applied to get more scalability that cannot be obtained by a single machine.

Another issue about multi-core CPUs is the cache consciousness. Even though we focused on multi-core CPUs in this paper, we didn't really take into consideration cache consciousness in the in-memory process. It could be another very important issue in terms of processing with multi-core CPUs, and utilizing cores with cache consciousness to shorten the time in in-memory process, then the total index construction time could be reduced.

## 6 Conclusion and Future Work

In this paper, we have proposed a novel method for online index construction using multi-core CPUs, which is based on the principle of parallel construction of multiple in-memory indexes for scalable online index construction. Our experimental evaluation has shown that the proposed method with some optimizations reduces index construction time close to by a factor of 4 with 8 cores without loss of query time.

We believe that optimization of the method is a promising area for future work. Also, applying multiple disks to the method to get more scalable index construction is another work in the future.

## References

- [1] V. N. Anh and A. Mat. Inverted index compression using wordaligned binary codes. *Information Retrieval*, 8(1):151-166, January 2005. Source code available from [www.cs.mu.oz.au/al-istair/carry/](http://www.cs.mu.oz.au/al-istair/carry/).
- [2] A. Biliris. The performance of three database storage structures for managing large objects. In M. Stonebraker, editor, *Proc. ACM SIGMOD Int. Conf. on the Management of Data*, pages 276-285, San Diego, California, June 1992.
- [3] S. Büttcher and C. L. A. Clarke. Indexing time vs. query time trade-offs in dynamic information retrieval systems. In *Proceedings of the 14th ACM Conference on Information and Knowledge Management (CIKM 2005)*. Bremen, Germany, November 2005.
- [4] S. Büttcher, C. L. A. Clarke and Brad Lushman. Hybrid Index Maintenance for Growing Text Collections. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR 2006)*. Seattle, Washington, USA. pages 356-363, 2006.
- [5] D. R. Cutting and J. O. Pedersen. Optimizations for dynamic inverted index maintenance. In J.-Luc Vidick, editor, *Proc. ACM SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 405-411, Brussels, Belgium, September 1990. ACM. ISBN 0-89791-408-2.
- [6] R. Guo, X. Cheng, H. Xu and B. Wang. Efficient On-line Index Maintenance for Dynamic Text Collections by Using Dynamic Balancing Tree. In *Proceedings of the 16th ACM CIKM Conference on Information and Knowledge Management*, pages 751-759, November 2007.
- [7] B. Jeong and E. Omiecinski. Inverted File Partitioning Schemes in Multiple Disk Systems. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 1994.
- [8] T. J. Lehman and B. G. Lindsay. The Starburst long field manager. In P.M. G. Apers and G. Wiederhold, editors, *Proc. Int. Conf. on Very Large Databases*, pages 375-383, Amsterdam, The Netherlands, August 1989. ISBN 1-55860-101-5.
- [9] N. Lester, J. Zobel, and H. E. Williams. In-Place versus Re-Build versus Re-Merge: Index Maintenance Strategies for Text Retrieval Systems. In *Proceedings of the 27th Conference on Australasian Computer Science*, pages 15-23, Darlinghurst, Australia, 2004.
- [10] N. Lester, A. Moffat, and J. Zobel. Fast online index construction by geometric partitioning. In *Proceedings of the 14th ACM CIKM Conference on Information and Knowledge Management*, pages 776-783, November 2005.
- [11] N. Lester, A. Moffat and J. Zobel. Efficient online index construction for text databases *ACM Transactions on Database Systems (TODS)*, 2008.
- [12] C. D. Manning, P. Raghavan and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press. 2008.
- [13] F. Scholer, H. E. Williams, J. Yiannis, and J. Zobel. Compression of inverted indexes for fast query evaluation. In K. Järvelin, M. Beaulieu, R. Baeza-Yates, and S. H. Myaeng, editors, *Proc. ACM SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 222-229, Tampere, Finland, August 2002.
- [14] W. Y. Shieh and C. P. Chung. A statistics-based approach to incrementally update inverted files. In H. R. Arabnia, editor, *Proc. Int. Conf. on Information and Knowledge Engineering*, pages 38-43, Las Vegas, Nevada, June 2003. CSREA Press.

- [15] K. Shoens, A. Tomasic, and H. Garcia-Molina. Synthetic workload performance analysis of incremental updates. In W. B. Croft and C. J. van Rijsbergen, editors, Proc. International ACM SIGIR Conf. on Research and Development in Information Retrieval, pages 329-338, Dublin, Ireland, July 1994.
- [16] C. Stanfill. Partitioned posting files: a parallel inverted file structure for information retrieval. Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval, pages 413-428, Brussels, Belgium, 1989.
- [17] A. Tomasic, H. Garcia-Molina, and K. Shoens. Incremental updates of inverted lists for text document retrieval. In Proc. ACM SIGMOD Int. Conf. on the Management of Data, pages 289-300, Minneapolis, Minnesota, May 1994. ACM.
- [18] I. H. Witten, A. Moffat, and T. C. Bell. Managing Gigabytes: Compressing and Indexing Documents and Images. Morgan Kaufmann, San Francisco, California, second edition, 1999.
- [19] B. Yates and R. Neto. Modern Information Retrieval. Addison Wesley, 1999.
- [20] J. Zobel and A. Moffat. Inverted files for text search engines. ACM Computing Surveys, 38(2), 2006.



# Recursive Partitioning Method for Trajectory Indexing

Elizabeth Antoine, Kotagiri Ramamohanarao, Jie Shao and Rui Zhang

Department of Computer Science and Software Engineering  
The University of Melbourne, Victoria  
Australia

Email: {eantoine, rao, jsh, rui}@csse.unimelb.edu.au

## Abstract

A trajectory is defined as the record of time-varying spatial phenomenon. The trajectory database is an important research area that has received a lot of interest in the last decade, with the objective of trajectory databases being to extend existing database technology to support the representation and querying of moving objects and their trajectories. Querying in trajectory databases can be very expensive due to the nature of the data and the complexity of the query processing algorithms. Given also that location-aware devices, like the GPS, are present everywhere these days, trajectory databases will soon face an enormous amount of data. Consequently the performance in the presence of a vast amount of data will be a significant problem and efficient indexing schemes are required to support both updates and searches efficiently.

This paper provides the methodology for using the recursive partitioning technique for indexing trajectories in the unrestricted space, which is called the Recursively Partitioned Trajectory Index (RPTI). RPTI uses the two-level indexing structure, as does the state of art indexing scheme, SETI, and maintains separate indices for the space and time dimensions. We present the algorithms for constructing the RPTI and the algorithms for updates that include insertion and deletion. We also provide the results of the experimental study on the RPTI and have demonstrated that RPTI is better than SETI in handling trajectory-based queries and is competitive with SETI in handling coordinate-based queries. The structure of RPTI can be easily implemented by using any of the existing spatial indexing structures. The only design parameters required are the standard disk page size and maximum level of recursive partitioning.

**Keywords:** Indexing Trajectories, Recursive Partitioning Method, Trajectory and Coordinate-based Queries

## 1 Introduction

Location-aware devices such as the GPS, which is the sequel to satellite and atomic clock technologies, have led to a number of new applications, such as fleet management and location-based solutions. Mobile phones that are E911-enabled are used to locate the mobile phone user with a variation of a few hundred meters, which is helpful in providing location infor-

mation during emergencies. In addition to outdoor location devices like the GPS, there are devices for determining the location indoors. The 3D ultrasonic location system [Ward and Jones, 1997], which is low-power and wireless, is one such system, and has led to new applications such as context-aware applications that respond to the user as he/she moves around.

Since the location-aware devices are used extensively these days, trajectory (or moving object) databases will soon be faced with the task of managing an enormous amount of data. The performance of the trajectory databases will significantly decline in the presence of a vast amount of data as the sequential access is time-consuming. Indexing structures are efficient in handling large data sets in various application domains, as they allow random look-ups; hence, indexing structures would be efficient in handling trajectory data as well. Traditional indexing structures like B-trees [Comer, 1979] are not efficient in ordering the multidimensional data and therefore cannot be used in spatial and trajectory databases.

Extensive work has been done on spatial indexing and R-tree variations have been found to be efficient in query processing [Guttman, 1984, Bentley and Friedman, 1979, Sellis et al., 1987, Beckmann et al., 1990, Kamel and Faloutsos, 1994]. In the domain of trajectory indexing, R-tree variations and extensions, such as, the three dimensional R-tree [Theodoridis et al., 1996], the TB tree [Pfoser et al., 2000], the STR tree [Pfoser et al., 2000] and the SETI tree [Chakka et al., 2003] have been introduced for indexing past locations of moving objects. In this paper we examine one such indexing structure that decouples the indexing of the space dimension from the time dimension and uses the recursive partitioning method for indexing in the space dimension. Similar to the SETI structure, the RPTI uses the  $R^*$ -tree to index the time intervals.

The sequel of the paper is organized as follows. Section 2 describes the related work, data model and queries. Section 3 explains the RPTI structure. Section 4 provides the algorithms for the RPTI structure. Section 5 details the experimental study of the RPTI structure. Section 6 gives the conclusion.

## 2 Related Work, Data Model and Queries

### 2.1 Related Work

In this section, we will be discussing some of the indexing structures that record past locations of the moving objects and those that organize motion in an unrestricted space. R-tree and its variations are found to be efficient in handling multidimensional data and have found their way into the commercial database systems. Hence, we will be discussing some of the R-tree variations and extensions that are used in indexing trajectories. We will be discussing the three

Copyright ©2010, Australian Computer Society, Inc. This paper appeared at the Twenty-First Australasian Database Conference (ADC2010), Brisbane, Australia, January 2010. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 104, Heng Tao Shen and Athman Bouguettaya, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

dimensional R-tree [Theodoridis et al., 1996] and the TB trees [Pfoser et al., 2000] which are efficient in handling coordinate-based queries. We will also discuss the SETI [Chakka et al., 2003] structure, which partitions the space into hexagons and then indexes the time dimension for every hexagon. This is useful in analysing the recursive partitioning of space in indexing trajectories.

The three dimensional R-tree [Theodoridis et al., 1996] are based on the R-tree index, which is widely used for indexing of spatial data. It is based on the previous efforts of the authors with regard to multimedia application modelling. The proposed indexing schemes was the first step towards a new direction of spatio-temporal indexing, while research has mainly focused on content-based image indexing - i.e., fast retrieval of objects using their content characteristics (colour, texture, shape). A three dimensional R-tree is a straightforward method of indexing trajectories. This method extends the R-tree to be used in a three dimensional space, with time as an extra spatial dimension (2D space + 1D time). Three dimensional R-trees are designed to handle coordinate-based queries, like the time-slice and time-interval queries, and is inefficient in answering trajectory-based queries. Based on the experimental study in [Chakka et al., 2003], SETI is shown to outperform the three dimensional R-tree. SETI partitions the space into static non-overlapping partitions and, for each partition, it uses the  $R^*$ -tree to build a sparse index over the time dimension [Chakka et al., 2003].

The TB-tree, introduced in [Pfoser et al., 2000], is fundamentally different from the access method they previously discussed (STR-tree [Pfoser et al., 2000]). The STR-tree introduces a new insertion/split strategy to achieve trajectory orientation, while not compromising the space discrimination capabilities of the index too greatly. Apart from this, the STR-tree is an R-tree-based access method. An underlying assumption when using the R-tree is that all inserted geometries are considered as parts of trajectories and the segments are stored independently in the R-tree and the STR-tree structures.

The TB-tree, provides an access method that strictly preserves trajectories, such that a leaf node only contains segments belonging to the same trajectory, thus the index is best understood as a trajectory bundle. As a drawback, line segments that are not part of the same trajectory but lie spatially close will be stored in different nodes. As the overlap increases, the space discrimination decreases and, thus, the classical range query cost increases. However, by giving up on space discrimination, there is a gain in trajectory preservation [Pfoser et al., 2000]. The structure of the TB-tree is actually a set of leaf nodes, each containing a partial trajectory, connected by a doubly linked list that preserves trajectory evolution. By visiting an arbitrary leaf node, these links allow us to retrieve the whole or part of the trajectory at minimal cost.

In spite of its clear advantages on trajectory-based query processing, the TB tree has a crucial drawback in its insertion strategy, as new trajectory data are always inserted to the right end of the tree, leading its performance to depend on the order of the data insertion. When an object enters an area where the position transmission system does not work, its information is locally stored and later transmitted to the central server, while other moving objects would have transmitted their positions. This violates the assumption that the trajectory data are inserted in the index in a purely chronological order [Manolopoulos et al., 2005]. Deletions are often ignored in the trajectory indexing structures and this is true with the TB trees as well. However, trajectories of objects that are no

longer useful need to be deleted and this leaves holes in the TB-tree structure.

The Scalable and Efficient Trajectory Index (SETI) [Chakka et al., 2003] partitions the spatial dimension into static, non-overlapping partitions and, for each partition, it builds a sparse index over the time dimension. Any spatial index can be used for this sparse index, including the R-tree and its variants, like the  $R^*$ -tree. The boundaries of the spatial dimensions remain constant or change very slowly over the lifetime of the trajectory data set growth, whereas the time dimension is continually increasing. As the extent of the spatial dimensions does not change, an indexing structure could partition the spatial dimensions statically however, the number of partitions should be calculated prior to the processing. Within each spatial partition, the indexing structure only needs to index lines in a 1D (time) dimension. Consequently, such an approach will not exhibit the rapid degradation in index performance that is generally observed for 3D indexing techniques.

Each trajectory segment is stored as a tuple in a data file, with the restriction that any single data page only contains trajectory segments that belong to the same spatial cell. The lifetime of a data page is defined as the minimum time interval that completely covers the time-spans of all the segments stored in that page. The lifetime values of all pages that are logically mapped to a spatial cell are indexed using an  $R^*$ -tree [Chakka et al., 2003]. These temporal indices are sparse indices, as only one entry for each data page is maintained instead of one entry for each segment. Using sparse indices has two distinct advantages: there are smaller index overheads and an improved insertion performance [Chakka et al., 2003]. The temporal indices also provide the temporal discrimination in searches.

There are several advantages for this technique and some are given below. As the objects that are actually indexed are one-dimensional time lines, the indexing structure does not suffer from the curse of dimensionality [Berchtold et al., 1998], which causes the performance of indexing structures to degrade rapidly as the number of dimensions increases. Updates to trajectories tend to add new segments to the ends of existing trajectories. To support high append rates, SETI keeps the last location of each object in an in-memory front-line structure. When a new location update for an object is available, the last position of that object in the front-line structure is looked up and adds the trajectory segment to the SETI index. As only a sparse  $R^*$ -tree index is used on the time dimension, such updates are very fast.

The index scales well to handle large trajectory data sets because of the use of multiple sparse indices. SETI can be viewed as a logical indexing structure that can be built on top of existing spatial indexing techniques, such as an R-tree. Consequently, implementing SETI is much easier than implementing a new physical indexing structure. In SETI, spatial discrimination is maintained by logically partitioning the spatial extent into a number of non-overlapping spatial cells. Each cell contains only those trajectory segments that are completely within the cell. If a trajectory segment crosses a spatial partitioning boundary, then that segment is split at the boundary and inserted into both cells. We provide the data model and discuss on the query processing in the following sections.

## 2.2 Data Model

When an object, in this example - a car, moves around space; the successive locations of the car are recorded

at specific time instances using the location-aware devices. The locations of the object are connected to form a sequence of line segments using the interpolation methods. The set of line segments (or polyline) represents the trajectory of the moving point object and is illustrated in Figure 1.

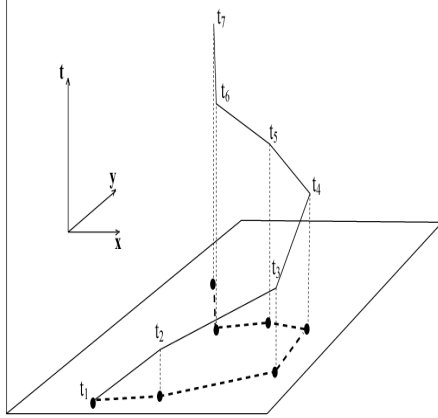


FIGURE 1: The Locations of the Spatial Object and the Corresponding Trajectory [Pfoser et al., 2000]

The motion of an object is approximated as the series of line segments where each line segment connects two consecutive update positions of the object [Güting et al., 2000, Kollios et al., 1999, Papadias et al., 2000, Pfoser et al., 2000, Pitoura and Samaras, 2001, Saltenis et al., 2000]. Each line segment ‘s’ is represented by its segment id ‘sid’ and the connected line segments of an object are called its trajectory, and are identified by its unique trajectory id ‘tid’. The representation of a trajectory (*trj*) is stated formally in [Chakka et al., 2003].

A trajectory is represented as:

$$trj (tid, (u_0, u_1, u_2, \dots, u_n, \dots))$$

$(u_0, u_1, u_2, \dots, u_n, \dots)$  is a sequence of points representing the positions of the moving object. Each point  $u_i$  is given by  $(x_i, y_i, t_i)$  where  $(x_i, y_i)$  represents the position of the moving object in space recorded at the time  $t_i$ .  $(u_0, u_1, u_2, \dots, u_n, \dots)$  are sequenced in the increasing order of time ( $u_i < u_{i+1}$ ).

The trajectory segment of an object is represented by:

$$s_i (tid, sid, u_{i-1}, u_i)$$

$(u_{i-1}, u_i)$  are the two update positions of the object and  $s_i$  represents the line segment that connects the two update positions. The insertion procedure of the RPTI structure has  $u_i$  as its input parameter and the procedure constructs the line segment based on the previous location  $u_{i-1}$  and inserts the new line segment  $s_i$ , as represented above with some additional variables. These will be discussed in the following sections. Given the above formal definition of trajectories, we provide a brief note on query types in the next section.

### 2.3 Queries

Querying in trajectory databases can be very expensive due to the nature of the data and the complexity of the query processing algorithms. Given also that location-aware devices, like the GPS, are ubiquitous these days, trajectory databases will soon face an enormous amount of data. Consequently the performance in the presence of a vast amount of data will be a significant problem. Queries on moving objects can

be broadly classified into two categories: queries that ask questions about the future positions of moving objects; and queries that ask questions about the historical positions of moving objects [Pfoser et al., 2000, Theodoridis et al., 1996, Frentzos, 2003, De Almeida and Güting, 2005, Chakka et al., 2003]. We focus on the queries relating to the historical positions of moving objects.

The historical queries are further classified into coordinate-based queries and trajectory-based queries. Coordinate-based queries include the time-slice queries, which select all the objects that lie within a given area at a given time instant; time-interval queries, which include the objects that lie within a given area and given time period; and nearest neighbour queries. The trajectory-based queries are further classified into topological queries and navigational queries.

Topological queries examine the whole or part of the trajectory of an object. For example, the predicate ‘collision’ examines two trajectories if they intersect in a given area at a specific time instant. The navigational query involves the information derived from the trajectories, such as the speed and the heading of an object. The average or top speed of an object is obtained by the fraction of travelled distance over time. The heading or the direction of travel of the object is computed by determining the vector between two specified positions [Pfoser et al., 2000]. The following are examples of the coordinate and trajectory-based queries; and their respective SQL statements [Erwig and Schneider, 2002].

#### Coordinate-based queries:

**Time-interval Query:** “find all objects that are present within a given area during a given time interval” (range query in space and time dimension). Example for time-interval queries is given below. ‘Id’ represents the unique identifier of the cars. ‘Line’ is the additional information that is stored about the travel; ‘trajectory’ as already mentioned, represents, the moving path of the cars. The above query represents the query window at the time dimension. It extracts the information about all the cars that were present at the time period ‘P’, without any clause in the space dimension.

*Q1: Where exactly were the cars during the time period P?*

```
SELECT Id, Line, trajectory AS Stretch
FROM Cars WHERE Trip present P;
```

**Time-slice Query:** “find all objects that are present within a given area during a given time instant” (range query in space with a zero extent in time dimension). Example for time-slice queries is given below.

‘Loc’ represents a point  $(x, y)$  in the 2D space and time instant ‘I’ represents the point  $(t)$  at the time dimension. It extracts the ‘Id’ of all the cars present at a particular location (Loc) in the space dimension and at a particular time instant (I) in the time dimension.

*Q2: Give the IDs of all the cars present in the location X at time instant I?*

```
SELECT Id from Cars C WHERE Trip
passes Loc AND Trip present I;
```

**Trajectory-based queries:**

“find the average speed of a moving object given its trajectory id”. Example for trajectory query is given below.

‘Car’ is a relation and ‘h’ represents the car that has the Id, ‘C’. The following query retrieves the whole trajectory information for the car ‘C’ and calculates the average speed of the car given by ‘*average\_speed*’ and ‘h.route’ represents the trajectory Id.

Q1: What is the car’s (C) average speed?

```
SELECT h.route, average_speed FROM
Car h WHERE h.id = C;
```

In order to answer the queries efficiently, the data needs to be represented appropriately. Due to the tremendous increase in the amount of data that are stored in the database system, the sequential access becomes time-consuming and the need for an efficient storage and access mechanism arises. Indexes provide the basis for rapid random look-ups and, for the indexing to be worthwhile, the process of creating the index must not be time-consuming, otherwise the queries could be answered more efficiently without an index. While dealing with the proximity queries in spatial data, the database management system needs to sort the spatial data efficiently, based on the space occupied by the data, to enable random loop-ups. Such techniques are known as spatial indexing methods [Samet, 1995]. The same is true for trajectory queries.

### 3 Recursively Partitioned Trajectory Index (RPTI)

In trajectory data sets that represent the past locations of moving objects, the extent of the spatial dimensions change very slowly over the lifetime of the trajectory data set growth, whereas the time dimension is constantly increasing. Similar to the SETI structure [Chakka et al., 2003], the RPTI indexing structure partitions the space by exploiting the static space dimension. Instead of partitioning the space into non-overlapping spatial cells, space is partitioned hierarchically. The number of levels is denoted by  $L$  and at every level  $l$  ( $l = 0, \dots, L - 1$ ), the number of partitions is  $4^l$  and each level is composed of  $2^l - 1$  equally spaced lines in each dimension. A line segment intersected by any partitioning line of level  $l$  belongs to level  $l + 1$ .

At each level, for every partition an indexing structure like the  $R^*$ -tree is used to index lines in the time dimension. The number of  $R^*$ -trees at every level is given by  $4^l$ . Similar to the SETI structure, the RPTI indexing mechanism achieves good spatial and temporal discrimination. Discrimination represents the ability to identify the candidate set of index entries with few false hits. The Recursively Partitioned Trajectory Index (RPTI) structure is illustrated in Figure 2. Each segment of the trajectory is stored as a tuple in the level files. The insertion and deletion strategies are explained in detail in the following section. For each level file, a space-filling curve, such as the z-ordering curve [Orenstein and Merrett, 1984], is used to calculate the z-order value for each line segment. The line segment is stored as a tuple in its corresponding level file, with the restriction that any data page will only contain line segments that belong to the same partition.

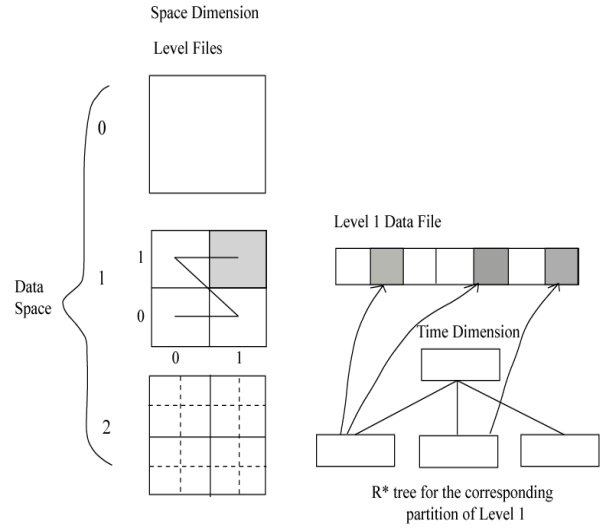


FIGURE 2: Recursively Partitioned Trajectory Index (RPTI) Structure

For every partition, an  $R^*$ -tree is used to index the *life-time* of every data page. The *life-time* of every page covers the time-spans of all the line segments stored in that page. Any spatial indexing structure other than  $R^*$ -tree can be used for this purpose. As mentioned in the SETI structure [Chakka et al., 2003], the  $R^*$ -tree index is sparse as there is only one entry for each data page [Chakka et al., 2003]. This makes the searching efficient but, during insertions and deletions, the life time of the data page might change. If the life time of a data page changes, then the corresponding  $R^*$ -tree should be updated.

### 4 Algorithms for RPTI Structure

This section explains in detail the insertion, deletion and search process for the RPTI structure and presents the relevant algorithms.

#### 4.1 Insertion

RPTI maintains a *front-line* structure similar to the one presented in SETI, which has the last update location of every moving object. The last update position of all the trajectories is maintained in a hash structure that is indexed by the trajectory id ‘*tid*’, which is unique for every moving object.

When there is an update in the position of the moving object, say ‘*obj*’, the hash structure is used to get the information of the last updated position based on its ‘*tid*’. A line segment is constructed connecting the last update position and the new position, and this is inserted into the RPTI structure. The level of the line segment ‘*Level*’, with its z-order value ‘*zvalue*’, is computed and the new line segment is inserted into the corresponding level file. The line segment tuple inserted into the level files is of the form:

$$s_i(tid, sid, u_{i-1}, u_i, zvalue)$$

The SETI structure results in line segments intersecting spatial cell boundaries and is handled by splitting the line segments in the space and time dimensions. However, splitting the line segments leads to additional computation during updates and searches, where the split line segments have to be merged. The number of partitions is an important factor in any partitioning strategy that affects the area covered by the partitions. If the partitions cover large areas, then the space discrimination of the index is reduced. This

can, however, be tolerated when the number of line segments that fall inside the partitions is small. If the partitioning is done very finely, then the number of segments that cross the boundary of the spatial cells increases, which increases the computational time [Chakka et al., 2003] for SETI.

RPTI structure avoids the splitting of line segments by inserting the line segments at their appropriate level, and as is illustrated in Figure 3. The line segments of a given trajectory may be found in more than one level. An in-memory structure called *track – trajectory* is maintained to hold the information about every trajectory. The tuple inserted to the *track – trajectory* file is of the form  $(tid, Level, zvalue)$ . When a new line segment is inserted to the RPTI structure, the *track – trajectory* is consulted; no updates occur if the level file and z-order value for the trajectory matches with the level file and z-order value of the new line segment, otherwise a new tuple with a trajectory id, level file and z-order value is inserted into the *track – trajectory*. The updates also occur if the *'tid'* is not found in the *track – trajectory* file. More than one entry for a trajectory denotes the distribution of the line segments of the corresponding trajectory over the level files.

By doing this, the whole trajectory can be retrieved by consulting the *track – trajectory* file and the problem of a line segment spanning over multiple partitions is eliminated; a problem which is prevalent in the SETI structure. Such a *track – trajectory* structure can also be maintained for SETI but the splitting of line segments cannot be avoided, which increases the computational time for handling trajectory-based queries that require the whole or part of the trajectory. As mentioned in [Chakka et al., 2003], about 8% of the line segments cross the partition boundaries and this increases not only the index size but also leads to additional computation, which is eliminated in RPTI.

The updates to the corresponding  $R^*$ -tree do not happen frequently. It is only updated when the inserted line segment tuple falls in a new data page. As already mentioned,  $R^*$ -tree is used to index only the *life – time* of data pages and, hence, there are infrequent updates to the  $R^*$ -trees. However, if the insertion changes the *life – time* of the data page, the corresponding index entry in the  $R^*$ -tree is updated. Algorithm 1 illustrates the insertion process.

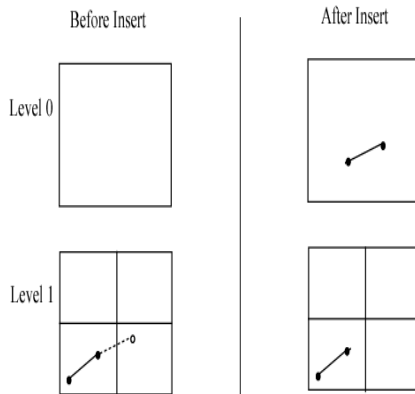


FIGURE 3: Insertion in RPTI

#### Algorithm 1: InsertSegment

```

input: New position ' $u_i$ ' of object ' $obj$ ',
        Trajectory id ' $tid$ ' of object ' $obj$ '
begin
    Consult front – line and retrieve last
    position ' $u_{i-1}$ ' using ' $tid$ ';
    if no entries found for ' $tid$ ' then
        Insert the tuple  $(tid, u_i)$  in front – line;
        exit;
    end
    Insert the tuple  $(tid, u_i)$  in front – line;
    Construct new segment ' $s_i$ ' connecting last
    and new positions;
    Compute ' $Level$ ' and ' $zvalue$ ' for  $s_i$ ;
    /* (see Section 3) */
     $old - lifetime \leftarrow$  life time of data page
    where  $s_i$  is to be inserted;          /* (see
    Section 3) */
    Insert tuple ' $s_i$ '  $(tid, sid, u_{i-1}, u_i, zvalue)$ 
    to level file ' $Level$ ';
     $new - lifetime \leftarrow$  life time of data page
    after insertion;          /* (see Section 3) */
    Consult track – trajectory and retrieve the
    tuple for trajectory id, ' $tid$ ';
    if no entries found for ' $tid$ ' then
        Insert tuple  $(tid, level, zvalue)$ ;
    else
        for each entry that represent ' $tid$ ' do
            if level in each entry  $\neq Level$  AND
            z-order in each entry  $\neq zvalue$  then
                Insert tuple  $(tid, level, zvalue)$ ;
            end
        end
    end
    if  $old - lifetime \neq new - lifetime$  then
        Consult the corresponding  $R^*$ -tree
        based on ' $Level$ ' and ' $zvalue$ ';
        if no  $R^*$ -tree found then
            Construct  $R^*$ -tree with index entry
             $(new - lifetime, ptr\_to\_the\_datapage)$ 
        else
            Retrieve index entry  $old - lifetime$ 
            and update with  $new - lifetime$ ;
        end
    end
end

```

## 4.2 Deletion

Deletion of a trajectory can be efficiently done using the RPTI structure. Deletion is often ignored while proposing a trajectory index because of the assumption that deleting the trajectory of a moving object is meaningless after the transmitted positions are recorded. However, deletions are necessary when the trajectory of a moving object is no longer useful. Deletions include deleting a particular segment of a trajectory or deleting the whole trajectory of a moving object.

A list of trajectory Ids with their associated level file ' $Level$ ' and ' $zvalue$ ' is maintained in the *track – trajectory* file. Given the trajectory id and the segment id, deleting a particular segment or deleting the entire trajectory becomes straightforward. The deletion process is illustrated in Algorithm 2. Deleting trajectory segments from a data page changes the *life – time* of the data page and hence the respective  $R^*$ -tree indices need to be updated if the deleted segment tuple changes the *life – time* of the data page. The worst case scenario for deletion would re-

quire reading one disk page for each trajectory line segment when retrieving a single trajectory. This is due to the fact that the trajectory line segments are organized based on the spatial and temporal relations, which results in the trajectory line segments belonging to a trajectory to be placed in different disk pages. This is similar to the SETI structure; however, the computational time for deletion in RPTI is reduced by avoiding the splitting of line segments across partition boundaries.

---

**Algorithm 2: DeleteSegment**


---

```

input: Trajectory id 'tid' of object 'obj'
begin
    Consult track – trajectory and retrieve the
    tuple for the trajectory id, 'tid';
    Retrieve the data page that holds the
    segment tuple of trajectory 'tid' based on
    the Level and zvalue;
    old – lifetime  $\leftarrow$  life time of retrieved data
    page;
    Delete all the segment tuple of 'tid' or a
    particular segment of 'tid' if 'sid' is given;
end
    new – lifetime  $\leftarrow$  life time of data page after
    deletion;
    if old – lifetime  $\neq$  new – lifetime then
        Consult the corresponding R*-tree based on
        'Level' and 'zvalue';
        Retrieve index entry old – lifetime and
        update with new – lifetime;
    end

```

---

### 4.3 Search

The search is executed in the following steps for the coordinate-based queries.

**Step 1:** In the space dimension, the partitions that intersect the query window are found for every level.

**Step 2:** At each level, for every partition that intersects, the temporal index is searched based on the temporal predicates and the relevant data pages are retrieved. As mentioned earlier, the life-time of every data page is maintained in the *R\**-tree.

**Step 3:** If the data pages that are retrieved belong to the partitions that are completely inside the query window, then all the trajectory id of the segments that belong to the page are returned. Data pages retrieved might have more than one segment belonging to a trajectory, in which case the trajectory id is returned only once. However, if the segment id is needed, it can also be listed. When the data pages belong to the partitions that are not completely inside the query window, the spatial predicate is applied to each segment tuple in the data page.

The search is done as illustrated in the steps above and it produces the list of segments or trajectories that overlap with the query window. The search process of RPTI is quite similar to that of the SETI structure. However, the duplicate elimination that occurs because of the splitting of line segments that cross cell boundaries is not found in RPTI.

The search process for the trajectory-based queries, where the whole or part of the trajectory is retrieved, is straightforward and is illustrated in Algorithm 3.

---

**Algorithm 3: SearchTrajectory**


---

```

input: Trajectory id 'tid' of object 'obj'
begin
    Consult track – trajectory and retrieve the
    tuples for the trajectory id, 'tid';
    Retrieve the data page that holds the
    segment tuples of trajectory 'tid' based on
    the Level and zvalue;
    List all the segment tuples of 'tid' or a
    particular segment of 'tid' if 'sid' is given;
end

```

---

Similar to the case of deletion, the worst case scenario for trajectory-based queries would require reading one disk page for each trajectory line segment when retrieving a single trajectory. This is due to the fact that the trajectory line segments are organized based on spatial and temporal relations, which results in the trajectory line segments belonging to a trajectory needing to be placed in different disk pages. This is similar to the SETI structure; however, the computational time for trajectory-based queries in RPTI is reduced by avoiding the splitting of line segments across partition boundaries. The following section provides the experimental results for the RPTI structure for both the coordinate and trajectory-based queries.

## 5 Experiments and Results for RPTI

In order to assess the performance benefits of RPTI, we compare it with our prototype implementation of the SETI structure. The SETI structure performs better than the three dimensional R-tree and the TB tree for both the time-slice and time-interval queries as described in [Chakka et al., 2003]. In the following, we present the experimental results for the time-interval queries and trajectory queries. The experiments were conducted on an Intel®Core™2 Duo Processor at 2.00GHz that has the main memory capacity of 2.0GB. The Windows experience rating index for the processor is 4.7 out of 5.

Synthetic data sets are generated using the GSTD generator of spatiotemporal datasets [Theodoridis et al., 1999] to create trajectories of moving objects. For a specified number of moving objects, the GSTD data generator produces specified number of segments per moving object. The number of trajectories for the data sets is kept as 1K and for every trajectory the number of segments is varied to form 5 GSTD data sets. The total number of segment tuples for 5 GSTD data sets are 4M, 8M, 12M, 16M and 20M respectively and are represented as GSTD-4M, GSTD-8M, GSTD-12M, GSTD-16M and GSTD-20M. The size of the level files for the 5 GSTD data sets is 144MB, 288MB, 432MB, 576MB and 720MB respectively.

The number of partitions is restricted for the SETI structure, as the splitting of line segments that cross the spatial cell boundary degrades the performance of the index. The recursive partitioning method in the RPTI structure eliminates the problem of splitting line segments by moving the line segments that cross the boundary to a higher level. The RPTI method performs better when the spatial dimension is partitioned very finely, as the partitioning forces the line segments that cross the boundary to be moved to the higher level and this leads to a uniform distribution of line segments at the lower levels. This is illustrated in Figure 4. For the purpose of understanding the distribution of line segments over level files, we have plotted the average number of line segments per partition for the data set of 1M line segments (number

of trajectories and the number of line segments per trajectory is kept as 1K).

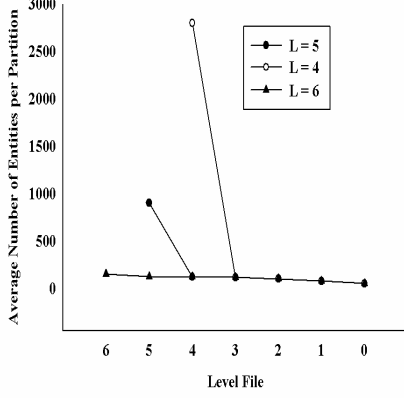


FIGURE 4: Distribution of Line Segments across Level Files for Varying L (Level until the Recursive Partitioning is done)

### 5.1 Performance of Insertion

The experimental results for the insertion performance of the GSTD data set with 4M segments (GSTD-4M) is given in Figure 5. For the SETI structure, the insertion of line segments involves the splitting of line segments if they cross spatial cells. But for the RPTI structure, the insertion is straightforward and it involves the update of the *track – trajectory* file if the trajectory id is not found in the file or if the line segment falls in a new level or a new partition.

After the level files are generated and the time indices are constructed, we present the performance of inserting 1K segments as an average time for inserting a single line segment. The result demonstrates that the RPTI structure outperforms the SETI for insertion, and that the reason for the increase in average time for the SETI is due to the splitting process, which is eliminated in our recursive partitioning method.

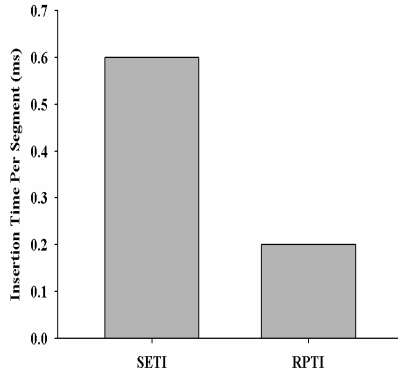


FIGURE 5: Insertion Performance

### 5.2 Time-interval Queries

The performance of RPTI for the time-interval queries tends to be competitive with the SETI structure. The slight increase in computation time is because of the recursive partitioning of space in RPTI; each level file needs to be checked for the relevant partitions. The number of partitions to be examined for false positives is higher as the partitions that are not completely enclosed by the query window tend to be

found in almost every level. The number of segments per trajectory is varied and experiments were conducted on the GSTD data set with 1K trajectories. The query window in the experiments was chosen to have 0.1% and 1.0% query selectivity. The total number of line segments in the data sets is 4M, 8M, 12M, 16M and 20M respectively. The results show that the RPTI structure is competitive with the SETI for the time-interval queries. Figures 6 and 7 illustrate the results of the experiments by varying the number of line segments per trajectory and the query window size.

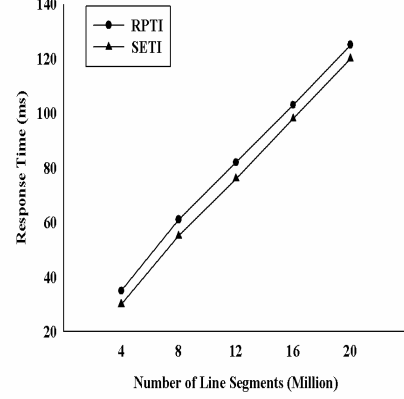


FIGURE 6: Time-interval Query Performance for 0.1% Query Selectivity

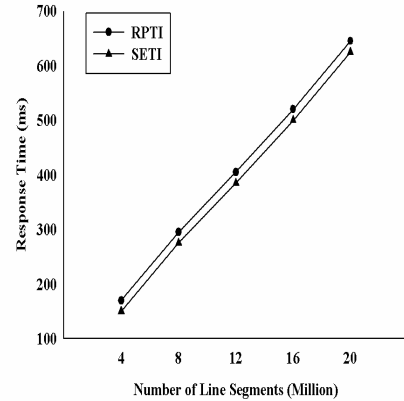


FIGURE 7: Time-interval Query Performance for 1.0% Query Selectivity

If the query window in the space dimension is large, then almost every partition at the higher levels intersects with the query window and this increases the number of false positives. The duplicate elimination in the SETI structure results in further computation and this is eliminated in RPTI. Though the number of partitions for checking false positives is higher in RPTI, the increase in computational time is compensated by the elimination of duplicates.

### 5.3 Time-slice Queries

In this section, we provide the experimental results of the time-slice query for the SETI and RPTI structures. Similar to the results of the time-interval query, the increase in response time for RPTI is due to the query window being executed at every level. However, the results show that the RPTI structure is competitive with the SETI. As already mentioned, the time-slice query executes the range query in the space dimension with a zero extent in the time dimension.



Figure 8 illustrates the experimental results for the time-slice queries. The number of segments per trajectory is varied and experiments were conducted on GSTD data set with 1K trajectories. The total number of line segments in the data sets is 4M, 8M, 12M, 16M and 20M respectively. During the search process, if the data pages belong to the partitions that are not completely inside the query window, the spatial predicate is applied to each segment tuple in the data page.

The number of partitions that are not completely inside the query window is high for the RPTI structure, as the area of the partition is large in the higher levels. Almost every partition in the higher levels tends to intersect the query window and, hence, the spatial predicate is applied to each segment tuple in the level files at the higher level. When the query window size is of 0.01% selectivity, the results of the RPTI and SETI tend to be similar. As with the time-interval queries, the time-slice query requires the look-up of partitions at every level and this results in the slight increase in response time of the RPTI over the SETI.

However, the performance of the RPTI greatly depends on the query window size and the position of the query window. The variation in the number of partitions that will be for false positives at every level is dependent on the query window size and the position of the query window. The results shown in Figure 8 are for the query window size of 0.01% selectivity, however there might be an increase in the response time of RPTI if the query window size is large.

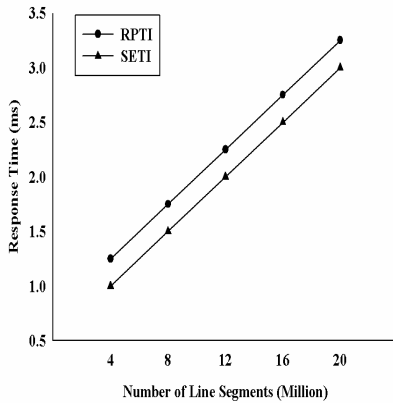


FIGURE 8: Time-slice Query Performance for 0.01% Query Selectivity

#### 5.4 Trajectory-based Queries

The RPTI structure performs better than the SETI structure for the trajectory queries, as the search process for retrieving the segments of the trajectory is quite straightforward. The SETI structure, however, requires the search process that involves duplicate elimination.

We also present the performance of the RPTI structure for trajectory queries by varying the number of moving object trajectories in the data set. The number of moving objects varies from 40K to 160K and the number of segments per moving object is kept as 100. The number of segments in the data set ranges from 4M to 16M. We present the response time of the retrieving segment id of 100 trajectories given the trajectory id. The experimental results show that the RPTI structure is better in answering trajectory-based queries than the SETI, as is illustrated in Figure 9. For a fair comparison, we maintained a sepa-

rate file for SETI, similar to the *track – trajectory* in the RPTI, that keeps track of the cells associated with a trajectory.

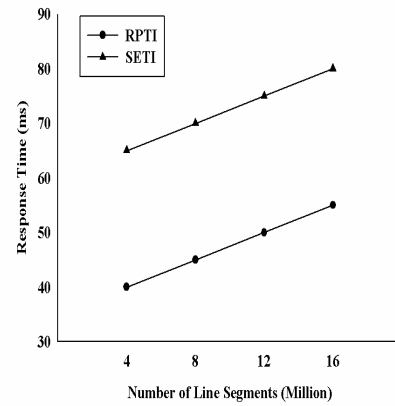


FIGURE 9: Trajectory Query Performance Scaling with the Number of Moving Objects

Deletion of trajectories is similar to trajectory queries where the whole or part of the trajectory is retrieved based on the trajectory id and, as mentioned earlier, the response time for deletion using the RPTI structure is less compared to the SETI. Deletion of trajectories in SETI involves deleting the line segments that cross the boundary twice and updating the  $R^*$ -tree twice if the deletion changes the *life – time* of the data pages.

The only design parameters required are the standard disk page size and maximum level of recursive partitioning. However, in SETI, the number of spatial partitions, which is a crucial parameter in any spatial partitioning strategy, is highly dependent on the distribution of data sets.

## 6 Conclusion

We have discussed the use of the recursive partitioning technique for trajectory indexing. Similar to the SETI structure, the RPTI decouples the space and time dimension. Contrary to the SETI structure, the RPTI recursively partitions the space and keeps track of the segments of a trajectory. Hence, the RPTI is better than SETI in handling trajectory-based queries and is competitive with the SETI in handling coordinate-based queries. The R-tree structure or its variations can be used to index the time dimension at every level. As, the RPTI is efficient in retrieving all the segments of the trajectory, deletion of trajectories is efficiently done. The time-interval queries require the look-up of partitions at every level and the structure increases the number of false positives during the search process. Hence, the response time of the RPTI for time-interval queries and time-slice queries is slightly higher than the SETI. The structure of the RPTI can be easily implemented by using any of the existing spatial indexing structures.

## Acknowledgements

This work is sponsored in part by National ICT Australia (NICTA).

## References

Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The  $r^*$ -tree: An efficient and robust access method for points and rectangles.



- In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 322–331, Atlantic City, NJ, May 1990. ACM Press.
- Jon Louis Bentley and Jerome H. Friedman. Data structures for range searching. *ACM Computing Survey*, 11(4):397–409, 1979.
- Stefan Berchtold, Christian Böhm, and Hans-Peter Kriegel. The pyramid-technique: Towards breaking the curse of dimensionality. *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 142–153, June 1998.
- V. Prasad Chakka, Adam Everspaugh, and Jignesh M. Patel. Indexing large trajectory data sets with seti. In *Proceedings of Conference on Innovative Data Systems Research (CIDR)*, 2003.
- Douglas Comer. The ubiquitous b-tree. *ACM Computing Survey*, 11(2):121–137, 1979.
- Victor Teixeira de Almeida and Ralf Hartmut Güting. Indexing the trajectories of moving objects in networks. *Geoinformatica*, 9(1):33–60, 2005.
- Martin Erwig and Markus Schneider. *STQL – A Spatio-Temporal Query Language*, chapter 6. Mining Spatio-Temporal Information Systems. Kluwer Academic Publishers, 2002.
- Elias Frentzos. Indexing objects moving on fixed networks. In *Proceedings of the 8th International Symposium on Spatial and Temporal Databases (SSTD)*, pages 289–305, Santorini Island, Greece, July 2003. Springer.
- Ralf Hartmut Güting, Michael H. Böhlen, Martin Erwig, Christian S. Jensen, Nikos A. Lorentzos, Markus Schneider, and Michalis Vazirgiannis. A foundation for representing and querying moving objects. *ACM Transactions on Database Systems*, 25(1):1–42, 2000.
- Antonin Guttman. R-trees: A dynamic index structure for spatial searching. *SIGMOD’84, Proceedings of Annual Meeting*, pages 47–57, June 1984.
- Ibrahim Kamel and Christos Faloutsos. Hilbert r-tree: An improved r-tree using fractals. In *VLDB ’94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 500–509, Santiago de Chile, Chile, September 1994. Morgan Kaufmann.
- George Kollios, Dimitrios Gunopulos, and Vassilis J. Tsotras. On indexing mobile objects. In *PODS ’99: Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 261–272, Philadelphia, Pennsylvania, June 1999. ACM Press.
- Yannis Manolopoulos, Alexandros Nanopoulos, and Apostolos N. Papadopoulos. *R-Trees: Theory and Applications (Advanced Information and Knowledge Processing)*. Springer, 1 edition, September 2005. ISBN 1852339772.
- Jack A. Orenstein and T. H. Merrett. A class of data structures for associative searching. In *PODS ’84: Proceedings of the Third ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 181–190, Waterloo, Ontario, Canada, April 1984. ACM.
- Dimitris Papadias, Yufei Tao, Panos Kalnis, and Jun Zhang. Indexing spatio-temporal data warehouses. In *Proceedings of 18th International Conference on Data Engineering (ICDE)*, pages 166–175, San Jose, CA, March 2000. IEEE Computer Society.
- Dieter Pfoser, Christian S. Jensen, and Yannis Theodoridis. Novel approaches in query processing for moving object trajectories. In *VLDB 2000: Proceedings of the 26th International Conference on Very Large Data Bases*, pages 395–406, Cairo, Egypt, September 2000. Morgan Kaufmann.
- Evaggelia Pitoura and George Samaras. Locating objects in mobile computing. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):571–592, 2001.
- Simonas Saltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario A. Lopez. Indexing the positions of continuously moving objects. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 331–342, Dallas, Texas, USA, May 2000. ACM.
- Hanan Samet. Modern database systems: The object model, interoperability, and beyond. In *Modern Database Systems*. ACM Press and Addison-Wesley, 1995. ISBN 0-201-59098-0.
- Timos K. Sellis, Nick Roussopoulos, and Christos Faloutsos. The r+-tree: A dynamic index for multi-dimensional objects. In *VLDB ’87: Proceedings of the 13th International Conference on Very Large Data Bases*, pages 507–518, Brighton, England, September 1987. Morgan Kaufmann.
- Yannis Theodoridis, Michalis Vazirgiannis, and Timos K. Sellis. Spatio-temporal indexing for large multimedia applications. In *ICMCS ’96: Proceedings of the 1996 International Conference on Multimedia Computing and Systems*, pages 441–448, 1996.
- Yannis Theodoridis, Jefferson R. O. Silva, and Mario A. Nascimento. On the generation of spatiotemporal datasets. In *Proceedings of the 6th International Symposium on Advances in Spatial Databases*, pages 147–164, Hong Kong, China, July 1999. Springer.
- Andy Ward and Alan Jones. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, 1997.



# Efficient Best Path Monitoring in Road Networks For Instant Local Traffic Information

Shuo Shang

Ke Deng

Kai Zheng

School of Information Technology and Electrical Engineering  
the University of Queensland

St. Lucia, Brisbane, QLD 4072, Australia

Email: {shangs, dengke, kevinzheng}@itee.uq.edu.au

## Abstract

The shortest path problem has been well studied previously. To improve the utility, the traffic conditions can be modeled to associate a weight to each road segment. The recent trend is to apply data mining techniques over use history which usually covers a long period of time such as months. However, this method fails to reflect the instant (i.e. temporary) traffic conditions change such as traffic accident or road work. Due to the temporary nature, the local instant traffic conditions makes more sense when an object is moving in road networks. In this work, we investigate the shortest path monitoring problem while the instant traffic conditions in local region update repeatedly around a moving object to a given destination. A simple way is to apply A\* algorithm repeatedly. However, the weakness is obvious. Because only a small fraction, i.e. the local area, of the whole networks have changed and the other parts keep intact. That means, for many vertices, that their paths (or the lower bounds of their paths) to the destination are still valid. This motivates us to maintain these information and reuse in the following computations. Our method is based on two tree structures where one records the previous computing results and the other aims to reduce the search space of subsequent processing. The experiments over real data set demonstrate an improvement of processing efficiency by one degree of magnitude at a small memory cost. In addition, the tree can be shared when monitoring the shortest paths for several moving objects to the same destination.

**Keywords:** Shortest Path Monitoring, Instant Local Traffic Information

## 1 Introduction

The shortest path problem has been well studied previously. A good survey of shortest path methods has been made by Pallottino et al.(1997) and Fu et al.(2006). While the shortest path may need to be recomputed if the object deviates from the planned path for some reasons, it also happens due to the traffic conditions. More recently, the traffic conditions of road networks have been considered in order to accurately capture the real shortest path, such as speed patterns, driving patterns and a multitude of other factors (proposed by Gonzalez et al. 2007) that provide important information in the computation of desirable routes. To do that, each road segment is assigned a weight which is derived from the use history of massive local users. However, the historic traffic conditions sometimes may be unreliable, because the traffic conditions of road networks in the real life often change tem-

porarily because of reasons such as traffic accidents and road works. Such kinds of information usually cannot be captured by the method weighting road segments based on historic traffic conditions.

One possible solution is to reflect the instant (i.e. temporary) changes in the global weighted road networks. However, it is not sensible to do so due to the following reason. The temporary changes of traffic conditions in road networks quite often, if not always, impact the proximity only. The users far away are not interested in these temporary impacts since these impacts change even when these users plan to travel these areas in the future. What these users need is the instant traffic conditions when they are close to these areas. Thus, the principle of processing instant traffic conditions is that more attention is paid on quantitative details of close region and on qualitative information of remote region. In this work, we studied the problem of the real-time shortest path monitoring due to the update of instant local traffic conditions.

The real-time shortest path planning monitoring is critical in various applications. While it is important in real life such as for route planning of rescue vehicle in emergency, we emphasize the scenarios in the virtual world such as traffic simulation system involving tens of thousands of agents (i.e. cars). Nowadays, more and more online racing games simulate the road networks and traffic conditions of real city such as *Midtown Madness*<sup>1</sup> and the car racing in *Second Life* game platform<sup>2</sup>, and they usually attract millions to play online simultaneously. In a typical scenario, several online players race from the same source to the same destination. The instant local traffic situation around each car is the *non-player character* (NPC) which is controlled by the gamemaster in the games. The real-time shortest path needs to be updated to guide the tour and the quick response is essential to the success of such online race games.

In this work, we aim to efficiently monitor the shortest path when local traffic conditions change. Since the changes happen arbitrarily during the traveling, the shortest path is the best possible path at current local traffic conditions and thus they are potentially best traveling choice. But users can choose to follow or not. In any case, the instant suggestion as the response to the traffic changes is valuable to users. To do that, the basic idea in this work is to compute the shortest path once and effectively reuse the results in the following monitoring. Initially, the classic A\* algorithms (proposed by Hart et al. 1968) are applied. This results are recorded using two tree-like data structures. While one tree is for recording the previous computing results, the purpose of the other tree is to reduce the search space of subsequent processing. At any instance when the latest local traffic conditions is known, the methods are developed to update the traverse trees and identify the new shortest path efficiently. The experiments over real data set demonstrate an improvement of processing efficiency by one degree of magnitude at a small cost

Copyright ©2010, Australian Computer Society, Inc. This paper appeared at the Twenty-First Australasian Database Conference (ADC2010), Brisbane, Australia, January 2010. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 104, Heng Tao Shen and Athman Bouguettaya, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

<sup>1</sup>[www.microsoft.com/games/midtown/](http://www.microsoft.com/games/midtown/)

<sup>2</sup>[heidballinger.wordpress.com/2008/07/19/car-racing-in-second-life/](http://heidballinger.wordpress.com/2008/07/19/car-racing-in-second-life/)

of storage. The tree can be shared when monitoring the shortest paths for several moving objects (e.g. cars in the online games) to the same destination. Note that the proposed method is also applicable to provide instant shortest path suggestion to users who deviate the planned paths at his own discretion.

The shortest path monitoring is also known as the dynamic shortest path planning (DSP) (Frigioni et al. 1996, 1998, 2000 and King 1999 and Demetrescu et al. 2004). In these works, the underlying networks with constant edge weights is allowed to be updated from time to time. The updates include insertion/deletion of edges and edge-weight updates, and these updates are known globally. Frigioni et al. (1996) studied the single-source DSP problem where the shortest path is maintained from a given source to all other vertices. Note that the single-source DSP is same to our problem only if no local instant traffic conditions is considered. King (1999) investigated the all-pair DSP problem where the shortest paths for all pairs of vertices are concerned. The all-pair DSP problem may deal with our problem but the local nature of this problem implies that it is not sensible to use the techniques for the all-pair DSP. Two steps are taken in all-pair DSP. The first step is to compute the shortest paths among all pairs using well known Floyd-Warshall algorithm in  $O(|V|^3)$  or Johnson's algorithm for sparse network in  $O(|V|^2 \lg |V| + |V||E|)$  where  $|V|, |E|$  are the numbers of vertices and edges in the network separately (Cormen et al. 2001). Then, the maintaining algorithm is applied when the weights of the edges change. The most recent algorithm (Demetrescu et al. 2004) is in  $O(|V|^2 \lg^3 |V|)$  amortized time per update. To solve our problem, all-pair DSP are much higher even comparing to direct applying Dijkstra's algorithm per update  $O(|V| \lg |V| + |E|)$ . Since heuristic algorithm A\* is no worse than Dijkstra's algorithm, we compares our method with simply applying A\* per update. Besides, our problem also differs from the time-dependent shortest path problem (Ding et al. 2008) where the job is to find the best departure time.

The contributions of this work are in three folds. First, this is the first effort to investigate the impact of instant local traffic conditions to the shortest path planning. Second, we proposed two tree structures to support the path monitoring to the single destination. The trees can be shared among several user if they have the same destination. Third, the proposed method is applicable to shortest path updating due to deviation of the planned path.

The remainder of the paper is organized as follows. In section II, problem description is presented and we introduce the related work in section III. Our approach including traverse trees, bounds and searching algorithm are introduced separately in section IV. Then, we discuss the memory cost and the traverse tree sharing among multiple moving objects in section V. The experiment results on real data set are demonstrated and explained in section VI. This paper is concluded in section VII.

## 2 Problem Description

In this paper, we use road networks to illustrate spatial networks. A road network can be modeled as a graph  $G = (E, V)$ , where  $V$  is a set of nodes corresponding to road junctions and  $E$  is a set of (non-directional) edges between two nodes in  $V$  corresponding to road segments. According to the traffic conditions, each road segment is associated with a non-negative weight. An edge can be a straight line or a poly line. If there is an edge in  $E$  linking two nodes in  $V$ , these two nodes are adjacent to each other. Let  $Adj(v)$  denote all the adjacent nodes of  $v \in V$ . Let  $d(v, v')$  be the distance along a path between two points  $v$  and  $v'$  (i.e. the total length of the edges along a network path between the two nodes). If there is no path connecting  $v$  and  $v'$ ,  $d(v, v') = \infty$ . The network distance

Table 1: A list of notations

Notation	Description
$V$	The set of all nodes
$E$	The set of all edges
$o$	The moving object, the central of range $A$
$A$	The circular region of instant local traffic conditions
$r$	The radii of range $A$
$weight(n_1, n_2)$	The weight of edge $(n_1, n_2)$
$d(v, v')$	the distance along a path between two points $v$ and $v'$
$Adj(v)$	all the adjacent nodes of $v \in V$
$(n_1, n_2)$	The edge between $n_1$ and $n_2$ , or the path from $n_1$ to $n_2$ if in a clear context
$SP(n_1, n_2),  SP(n_1, n_2) $	The shortest path between $n_1$ and $n_2$ and its length
$d_N(n_1, n_2)$	The network distance between $n_1$ and $n_2$
$d_E(n_1, n_2)$	The Euclidean distance between $n_1$ and $n_2$
$lb(v)$ and $ub.v$	the lower bound of node $v$
$ub(sp)$ and $SP.ub$	the upper bound of shortest path from $s$ to $t$
$Temp.path$	the shortest one of all known access paths from $s$ to $t$

between  $v$  and  $v'$ , denoted as  $d_N(v, v')$ , is defined using the network shortest path between the two nodes, denoted as  $SP(v, v')$ . In addition, we use  $d_E(v, v')$  to denote their Euclidean distance. The problem of this work can be described as follows:

Given a road network  $G = (E, V)$  and two points  $s, t \in V$ , an object  $o$  moves from  $s$  to  $t$ . An initial shortest path  $SP(s, t)$  is computed by A\* algorithm. At any instance,  $o$  is informed about the instant local traffic conditions in the surrounding region  $A$ , i.e. the weights of the road segments in  $A$  are updated. Without loss of generality, the local region  $A$  is represented as a circular region with radius  $r$  around the current location of  $o$ . The problem is to update  $SP(s, t)$  to  $SP'(s, t)$  on  $G$  with the updated local traffic conditions.  $SP'$  can be same as  $SP$  or not. Table 1 shows a list of notations used in this paper.

## 3 Related work

Two representative network shortest path algorithms are the Dijkstra's algorithm (proposed by Dijkstra.1959) and the A\* algorithm (proposed by Hart et al.1968). They both propagate a search "wavefront" from a source node  $v_s$  until a destination node  $v_d$  is reached. A heap  $H$  is used to keep all the nodes in the wavefront. The initial step of Dijkstra's algorithm is to put every node  $v \in Adj(v_s)$ , together with the distance  $d(v_s, v)$  (set to the length of the edge linking  $v_s$  and  $v$ ), into  $H$ . Then, the algorithm iterates an expansion process by replacing a node  $v \in H$ , where  $d(v_s, v)$  is the minimum for all nodes in  $H$ , by all the nodes in  $Adj(v)$ . For each node  $v' \in Adj(v)$ ,  $d(v_s, v')$  is set to  $d(v_s, v) + d(v, v')$  if  $v'$  is not yet in  $H$  or (in case  $v'$  in  $H$ ) the existing estimate of  $d(v_s, v')$  is larger than  $d(v_s, v) + d(v, v')$ . This process terminates when  $v_d$  is selected from  $H$  to expand (and  $d_N(v_s, v_d) = d(v_s, v_d)$ ). Dijkstra's algorithm can compute the shortest paths from a source node to multiple destination nodes. If there is only one destination, the wavefront expansion process can be optimized towards the direction of the destination node. This is the key idea of A\*. Instead of selecting  $v \in H$  with the minimum  $d(v_s, v)$ , a node  $v' \in H$  with the minimum  $d(v_s, v') + d_E(v', v_d)$  is selected to expand<sup>3</sup>. That is, when selecting a node  $v'$ , not only the computed network distance from  $v_s$  to  $v'$  is considered, the Euclidean distance from  $v'$  to  $v_d$  is also used as a directional guide. For any node  $v \in H$ ,  $d(v_s, v) + d_E(v, v_d)$  is called the *distance lower bound* of  $v$  from  $v_s$  to  $v_d$  (denoted as  $lb(v, v_s, v_d)$ , or  $v.lb$  when not causing ambiguity). Clearly, any valid

<sup>3</sup> $d(v_s, v') = d_N(v_s, v')$  when  $v'$  is selected to expand.

network path from  $v_s$  to  $v_d$  via  $v$  cannot be shorter than  $v.lb$ .

Gonzalez et al.(2007) stresses the importance of road hierarchy, speed pattern and driving pattern in the path planning. The speed pattern means that the speed may change at different time and road segments due to the traffic congestion or road condition. Therefore, to compute the fastest path exactly, the speed of each road segment should be dynamic. The speed pattern can be extracted from the use history of local person through the data mining techniques. The basic idea is that the choices of local people should always be the best, because they must have enough reasons to choose this way, such as security, road conditions and so on. The traffic pattern mining has also been studied by Agrawal et al.(1995), Han et al.(2000), Pei et al.(2001), Kanoulas et al.(2006). However, there is still a notably weakness of using historic traffic data. That is, the road conditions are changing continuously, the speed patterns based on mining the historic traffic conditions sometimes would be unreliable. They may not reflect the real conditions of road network and some temporarily changing of road conditions, such as road construction or traffic accident, will lead to serious mistake in route planning.

The dynamic shortest-path (DSP) problem (Frigioni et al.1996, 1998, 2000 and King 1999 and Demetrescu et al. 2004) is to recompute the shortest-path repeatedly while the underneath graph with constant edge weights is allowed to be updated from time to time. The updates include insertion/deletion of edges and edge-weight updates. Frigioni et al.(1996) study the single-source DSP problem and the fully dynamic algorithms are proposed with optimal space requirements and processing time. Experimental evaluations for single-source DSP algorithms can be found in Frigioni et al.(1996). Our problem can be viewed as a single-destination DSP problem which is different from the single-source problem due to the consideration of instant local traffic conditions. King (1999) investigates the all-pair DSP problem and Demetrescu et al.(2004) improves all-pair DSP algorithm by giving the optimal worst-case time. Two steps are taken in all-pair DSP. The first step is to compute the shortest paths among all pairs using well known Floyd-Warshall algorithm in  $O(|V|^3)$  or Johnson's algorithm for sparse network in  $O(|V|^2 \lg |V| + |V||E|)$  (where  $|V|, |E|$  are the numbers of vertices and edges in the network separately). The second step is to update the all impacted shortest paths when the weights of the edges change. The most recent algorithm (Demetrescu et al. 2004) is in  $O(|V|^2 \lg^3 |V|)$  amortized time per update. The all-pair DSP problem may deal with our problem but the local nature of this problem implies that it is not sensible to use the techniques for the all-pair DSP.

The time-dependent shortest path problem, proposed by Ding et al.(2008), investigates to find the best departure time such that the total travel time can be minimized over a road network, where the traffic conditions change from time to time.

#### 4 Our Approach

Once user receives the instant local traffic conditions, the system will recompute the shortest path from the current position to the destination. To do that, a simple way is to apply A\* algorithm repeatedly. However, the weakness is obvious. Because only a small fraction, i.e. the local area, of the whole graph have changed and the other parts keep intact. That means, for many vertices, that their shortest paths (or the lower bounds of their shortest paths) to the destination are still valid. This motivates us to maintain these information and reuse them in the following computations.

We proposed an algorithm called Instant Shortest Path

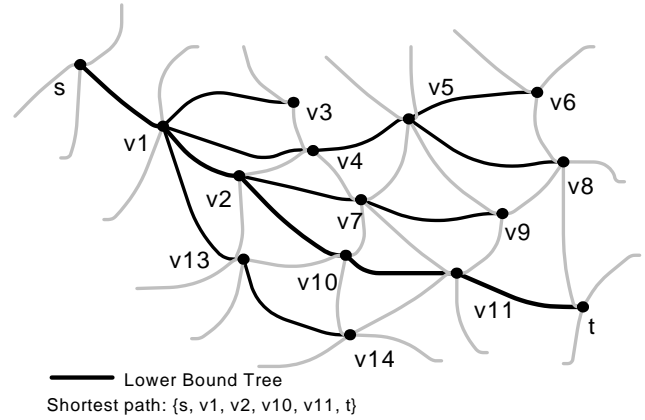


Figure 1: Once Searching Result

Monitoring (IBPM). Initially, the shortest path from the source to the destination is computed using A\* algorithm. This results are recorded using two tree structures, called Lower Bound Tree  $F$  and Upper Bound Tree  $T$ . When the local instant traffic conditions is updated, two steps are taken to update the shortest path. First, the nodes in the traverse trees are updated for the corresponding road segments. Second, the traverse trees are searched and the search space is constrained by the lower bound and upper bound provided by the traverse trees. In the path update, the principle of A\* algorithm is applied to guarantee the correctness. The details of tree structures and the techniques for update and search are introduced in details next.

#### 4.1 Two Tree Structures

Two traverse trees are established to help the shortest path monitoring. They are upper bound tree and lower bound tree. These trees are constructed when the initial shortest path is computed using A\* algorithm and kept updating until the moving object arrives the destination. Each node in traverse trees corresponds a real vertex in road networks. At any instance, one vertex only appears once in upper/lower bound tree. The upper bound and lower bound are key points of optimizations in our method.

For the lower bound tree, the initial tree roots at the source  $s$ , see an example in figure.1. Each non-leaf node  $v$  is attached with a value which is the lower bound from current location (i.e.  $s$  initially) to  $t$ , denoted as  $lb(v)$  or  $v.lb$ . In order to keep the size of the lower bound tree small, each vertex only corresponds to one node even though one vertex may be accessed through different paths (i.e. through different neighbor vertices). As a result, the path from each node to the root corresponds to the shortest path in the network. In the following process, since the weight of the edge changes, the attached lower bounds of nodes in the tree may change and consequently the tree structure may need to be updated. Some nodes will be removed from  $F$  and the lower bound tree may turn into a forest. One may encounter a problem during the tree updating. For a vertex  $v$ , its shortest path to source (i.e. the root in the tree) may go through another neighbor vertex due to the change of the weights of the networks. An example is shown in figure 2. The shortest path from source to  $v$  goes through  $v_1$  initially and then changes to go through  $v_2$ . In this situation, two operations are taken. First, the predecessor of  $v$  is updated to  $v_2$  and the link between  $v_1$  and  $v$  is removed. Second,  $v_1$  is set to be a leaf node in the tree by removing all other child nodes (explained later).

For the upper bound tree, it roots at the destination  $t$ . Initially, the computed shortest path using A\* algorithm is recorded, see the shortest path in figure.1. Each time when the shortest path is updated, the new path is

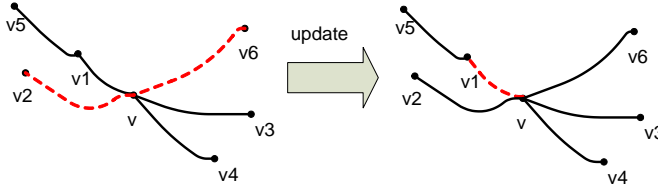


Figure 2: The Updating of Traverse Trees

recorded in the tree as well. Similar to lower bound tree, we keep one node for one vertex only in the upper bound tree. Thus, a node may appear in two shortest paths to  $t$  and thus it may have two predecessor nodes to the destination (i.e. root in the upper bound tree). The same processing applied as to lower bound tree. In the upper bound tree, the recorded shortest paths are called *access paths* since they are possible channels from the source to the destination. Among all access paths, the one with the minimum length is called *Temp\_path*. *Temp\_path* is current known shortest path. It can be used as the upper bound of the shortest path from  $s$  to  $t$  (in other words,  $|Temp\_path| \geq |SP(s, t)|$ ), denoted  $|Temp\_path|$  as  $ub(sp(s, t))$ .

We have discussed the method to update lower and upper bound trees. Now, we focus on lower and upper bound calculation. The lower bound is discussed in this section and the upper bound will be introduced in next section. The lower bound of a node  $v$  in the lower bound tree  $lb(v)$  is an estimated (or *heuristic*) distance from  $v$  to the destination  $t$ . A\* algorithm usually uses the Euclidean distance between  $v$  and  $t$ ,  $d_E(v, t)$  as estimated distance of  $|SP(v, t)|$ . According to A\* algorithm, the greater estimated distance leads to a smaller search space and better performance efficiency, and it must not be greater than  $|SP(v, t)|$  (otherwise it may lead to a wrong result). The lower bound attached with  $v$  calculated from the lower bound tree is no less than  $d_E(v, t)$  and no greater than  $|SP(v, t)|$ . That is,

$$|SP(v, t)| \geq lb(v) \geq d_E(v, t) \quad (1)$$

In our method, the lower bound calculation is from leaf to root. Firstly, we set the lower bound of each leaf node  $v$  to be the Euclidean distance between  $v$  and  $t$ ,  $d_E(v, t)$ . Then, we calculate the lower bound of other nodes based on their child nodes. In case one node  $v_1$  has more than one child node and the lower bound calculated from different child nodes are different, we choose the smallest one as the lower bound of  $v_1$ . The details of how to calculate the lower bound of nodes in lower bound tree is introduced in Algorithm.1.

---

**Algorithm 1: Lower Bound Calculation**


---

**Data:** Lower Bound Tree  $F$   
**Result:**  $lb(v)$  of each node

```

1  $\forall v \in F, lb(v) \leftarrow \infty$ ; for all leaf node  $v$  in  $F$  do
2   if  $NeighbourNumber(v) = 1$  then
3      $lb(v) \leftarrow \infty$ ;
4   else
5      $lb(v) \leftarrow d_E(v, t)$ ;
6   if
7      $lb(v.predecessor) > lb(v) + weight(v, v.predecessor)$ 
8     then
9        $lb(v.predecessor) \leftarrow$ 
10         $lb(v) + weight(v, v.predecessor)$ ;
11   $v \leftarrow v.predecessor$ ;

```

---

In Algorithm 1, all lower bounds are calculated from bottom to top. The leaf nodes have two different styles. If  $v$  has only one neighbor, its predecessor, which means  $v$  is at the end of a broken path and we set  $lb(v) \leftarrow \infty$  (line

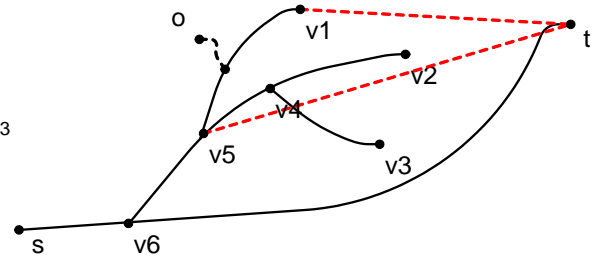


Figure 3: Lower Bound

2-3). Else, all other leaves' lower bounds are assigned as  $d_E(v, t)$  (line 4-5). For example, if  $v$  is equal to  $t$ ,  $lb(v)$  is assigned 0. Having fixed leaves' lower bound, the lower bound of all other nodes in  $F$  can be calculated from bottom to top. If one node has more than one child and the lower bounds from different child are different, we select the least one (line 6-7). For instance, as shown in figure.1,  $\{s, v_1, v_2, v_{10}, v_{11}, t\}$  is the shortest path from  $s$  to  $t$ , signed as  $SP(s, t)$ . Based on Algorithm 1,  $lb(v_{11}) = lb(t) + weight(v_{11}, t)$ ,  $lb(v_{10}) = lb(v_{11}) + weight(v_{10}, v_{11})$ , by this way,  $lb(v_7) = lb(v_9) + weight(v_7, v_9)$ . If one node has more than one child, such as  $v_2$  has two children,  $v_7, v_{10}$ . For one child  $v_7$ ,  $lb(v_2) = lb(v_7) + weight(v_2, v_7)$  and for the other child  $v_{10}$ ,  $lb(v_2) = lb(v_{10}) + weight(v_2, v_{10})$ . We select the least one of them and assign it as  $lb(v_2)$ . The calculated lower bound with each node  $v$ , the relation  $|SP(v, t)| \geq lb(v) \geq d_E(v, t)$  is held and we prove it using lemma 1,2,3 to guarantee the correctness.

**Lemma 1**  $\forall v, t, d_E(v, t)$  is the shortest distance between  $v$  and  $t$ .

**Lemma 2**  $\forall v, t$ , there is only one shortest path between  $v$  and  $t$ <sup>4</sup>.

**Lemma 3** If  $lb(v)$  is valid,  $|SP(v, t)| \geq lb(v) \geq d_E(v, t)$ .

**Proof:**

Step 1, prove  $lb(v) \geq d_E(v, t)$ .

In classic A\* algorithm,  $f(v_m) = d_N(s, v_m) + d_E(v_m, t)$ . For any non-leaf node  $v$ , according to algorithm 1,  $lb(v) = d_N(v, v_m) + d_E(v_m, t)$ .  $v_m$  is the leaf node with the least  $f(v_m)$  related to  $v$ . Because  $d_N(v, v_m) + d_E(v_m, t) \geq d_E(v, v_m) + d_E(v_m, t)$ , based on triangular inequality, we can get  $d_E(v, v_m) + d_E(v_m, t) \geq d_E(v, t)$ . So  $d_N(v, v_m) + d_E(v_m, t) \geq d_E(v, t)$ . That is  $lb(v) \geq d_E(v, t)$ . For example, in figure.3, assume that  $f(v_1) < f(v_2) < f(v_3)$ , then  $lb(v_5) = d_N(v_5, v_1) + d_E(v_1, t)$ . Because  $d_N(v_5, v_1) + d_E(v_1, t) \geq d_E(v_5, v_1) + d_E(v_1, t)$  and  $d_E(v_5, v_1) + d_E(v_1, t) \geq d_E(v_5, t)$ . We can get  $d_N(v_5, v_1) + d_E(v_1, t) \geq d_E(v_5, t)$ . That is  $lb(v_5) \geq d_E(v_5, t)$ . For any leaf node  $v$ , if  $v$  has more than one neighbor,  $lb(v) = d_E(v, t)$ . Else,  $lb(v) = \infty > d_E(v, t)$ . For all stated above,  $lb(v) \geq d_E(v, t)$ .

Step 2, prove  $|SP(v, t)| \geq lb(v)$ .

For any leaf node  $v$  in  $F$ , if  $v$  has more than one neighbor,  $lb(v) = d_E(v, t) \leq |SP(v, t)|$ . Else,  $lb(v) = \infty$ , which means  $v$  at the end of a broken path, so  $|SP(v, t)|$  is also equal to  $\infty$ . We can get  $lb(v) = |SP(v, t)| = \infty$ . For any non-leaf node  $v$  in  $F$ ,  $lb(v) = d_N(v, v_m) + d_E(v_m, t)$ . If  $SP(v, t)$  via one leaf node  $v_a$ ,  $|SP(s, v)| + |SP(v, t)| \geq f(v_a)$ . For  $v_m$  is the leaf node with the least  $f(v_m)$  related

<sup>4</sup>The tie is simple to process and not considered here.

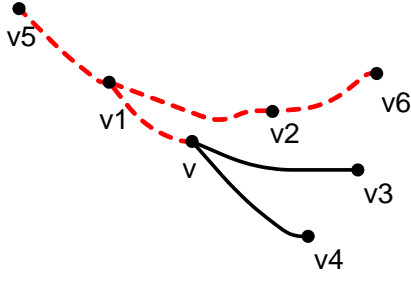


Figure 4: Lower Bound Tree Cleaning

to  $v$ , so  $f(v_a) > f(v_m)$ ,  $|SP(v, t)| + |SP(s, v)| > f(v_m)$ ,  $|SP(v, t)| > f(v_m) - |SP(s, v)|$ . Based on the theory of A\* algorithm,  $(s, v)$  is just the shortest path from  $s$  to  $v$ , so  $|SP(s, v)| = d_N(v, v_m)$ . Then,  $|SP(v, t)| > lb(v)$ . If  $SP(v, t)$  is not via leaf node,  $SP(v, t)$  must be via a candidate node  $o$ , as shown in figure.3,  $|SP(s, v_5)| + |SP(v_5, t)| \geq f(o)$ . Because  $v_m$  is a leaf node and  $o$  is still in the candidate heap ( $v$  did not have the chance to be popped in pre-searching phase), we can get  $f(o) > f(v_m)$  and  $|SP(s, v)| + |SP(v, t)| > f(v_m)$ . Therefore,  $|SP(v, t)| > lb(v)$ . For all of above,  $|SP(v, t)| \geq lb(v)$ . Integrating step 1 and step 2, when  $lb(v)$  is valid,  $|SP(v, t)| \geq lb(v) \geq d_E(v, t)$ .  $\square$

According to Lemma.3, the range of  $lb(v)$  has been controlled between  $|SP(v, t)|$  and  $d_E(v, t)$ .

In our application scenario, the object is moving continuously. When the user received the instant local traffic information, some weights of road segments will be updated and some corresponding nodes' lower bounds will also be invalid. Then, we have to "clean" the lower bound tree and remove all the invalid nodes. Once the weight of one edge  $e$  has changed, the lower bounds of end vertices of  $e$ ,  $e.end1$  and  $e.end2$ , will be invalid. In Algorithm.1, the lower bound is calculated from leaf to root, so the predecessors of  $e.end1$  and  $e.end2$  will also have invalid lower bounds. To ensure the correctness of lower bound and avoid additional computation, we propose our method to clean the lower bound tree. For each updated edge  $e$ , we find both  $e.end1$  and  $e.end2$ , then remove all the related nodes until root. For example, figure.4 shows a branch of lower bound tree originated from  $v_5$ . Assuming that  $v_6$  is a vertex of an updated edge, we remove all the related nodes above  $v_6$  (shown in dashed line). Because  $v_1$ , the predecessor of  $v$ , has been removed from  $F$ ,  $v$  becomes the root of a new branch.

#### Algorithm 2: Lower bound tree cleaning

**Data:** Lower Bound Tree  $F$   
**Result:** updated Lower Bound Tree  $F$

```

1 for each updated edge  $e$  do
2   RemoveNodeUntilRoot( $e.end1$ );
3   RemoveNodeUntilRoot( $e.end2$ );
```

Algorithm 2 describes the details how to clean invalid nodes of  $F$ . Firstly, find all the updated edges.(line 1) For each updated edge  $e$ , remove all the related node above  $e.end1$  and  $e.end2$  until root. (line 2-3)

## 4.2 Instant Best Path Monitoring

This section introduces the Instant Best Path Monitoring (IBPM) algorithm. It covers the details of upper bound maintenance and how to utilize lower and upper bound to implement our algorithm.

Utilizing upper bound tree  $T$ , we may find more than one access path from current position  $s$  to destination  $t$ , and set the shortest access path as  $Temp\_path$ . We assign the length of  $Temp\_path$  as the upper bound of

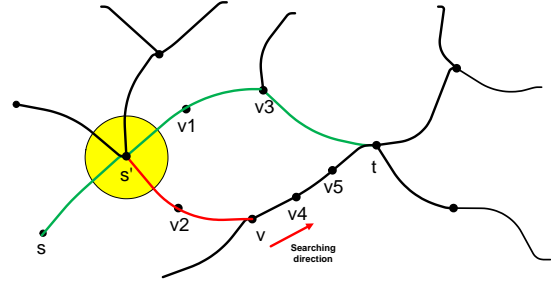


Figure 5: One possible access path

shortest path from  $s$  to  $t$ ,  $|Temp\_path| = ub(sp(s, t))$ . Different from lower bound,  $ub(sp(s, t))$  will be updated in the searching phase continually. When the current accessing node  $v$  is in the upper bound tree  $T$ , we get a new access path from  $s$  to  $t$ . Utilizing the upper bound tree  $T$ , we can calculate the length of the new access path, (denote as  $currentpath$ ) efficiently. Then, we compare  $|currentpath|$  with  $|Temp\_path|$ , if  $|currentpath| < |Temp\_path|$ , update  $Temp\_path$  to be  $currentpath$ . For instance, as shown in figure.5,  $Temp\_path$  is  $s', v_1, v_3, t$ . Node  $v$  is the current accessing node and it is at the upper bound tree  $T$ . Then, we get a new access path  $P = s', v_2, v, v_4, v_5, t$ . If  $|P| < |Temp\_path|$ , we set  $P$  to be  $Temp\_path$  and update the  $ub(sp(s, t))$ . The details will be introduced in Algorithm.3.

#### Algorithm 3: the Upper Bound Calculation

**Data:** Lower Bound Tree  $F$ , Upper Bound Tree  $T$ , Current accessing node  $v$

**Result:** the upper bound of  $SP(s, t)$

```

1  $|Temp\_path| \leftarrow \infty$ ;
2  $|currentpath| \leftarrow d_N(s, v)$ ;
3 if  $v$  in  $T$  then
4   while  $v.predecessor \neq NULL$  do
5     if  $v$  in  $F$  then
6        $|currentpath| + \leftarrow lb(v)$ ;
7       break;
8     else
9        $|currentpath| + \leftarrow$ 
10         $weight(v, v.predecessor)$ ;
11         $v \leftarrow v.predecessor$ ;
12 if  $|currentpath| < |Temp\_path|$  then
13    $|Temp\_path| \leftarrow |currentpath|$ ;
```

In Algorithm 3, the function monitors the upper bound of shortest path  $|Temp\_path|$  in searching phase. Firstly, we set the length of  $Temp\_path$  to be  $\infty$  and  $|currentpath|$  be the network distance from  $s$  to  $v$ ,  $d_N(s, t)$ . (line 2) When the current accessing node  $v$  is in the upper bound tree  $T$ , we can get the  $|currentpath|$  by calculate the sum of  $d_N(s, v) + d_N(v, t)$ . If the lower bound of  $v$  is valid and  $v \in T$ , which means  $lb(v) = d_N(v, t)$  and  $lb(v)$  is just the shortest path from  $v$  to  $t$ . The  $|currentpath|$  can be calculated directly.(line 5-6) Else, we have to calculate  $d_N(v, t)$  step by step according to  $T$ , until  $v = t$ .(line 9-10). Having got the  $|currentpath|$ , we compare it with  $|Temp\_path|$  and if  $|currentpath|$  is less than  $|Temp\_path|$ , both  $Temp\_path$  and  $UP(sp(s, t))$  will be updated.(line 11-12)

Then, we utilize the bounds (lower bound and upper bound) to implement our IBPM algorithm based on classic A\* algorithm. IBPM algorithm includes two steps, the first step (IBPM-I)uses bounds to pretreatment data, decreasing unnecessary data accessing to improve the efficiency. And the second step is based on some characters



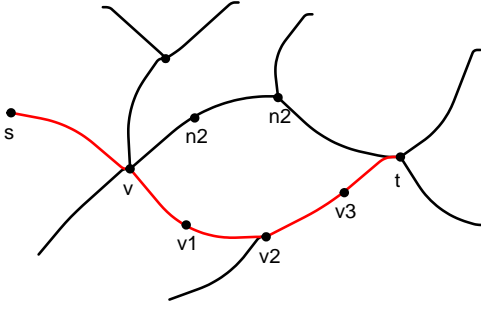


Figure 6: One Shortest path

of A\* algorithm and traverse trees, to achieve further optimization.

As mentioned in subsection 4.1, whether node  $v$ 's lower bound is valid depends on two different conditions,  $v \in F$  and  $v.pre = F.getpre(v)$ .  $v \in F$  means that the road conditions after  $v$  have not been updated and  $v.pre = F.getpre(v)$  means the current search is from the  $v.predecessor$  in  $F$  to  $v$ . Because A\* algorithm does not concern searching backward, if  $v.pre \neq F.getpre(v)$ , means the current search is from some other nodes and  $lb(v)$  may be unreliable. For each node  $v$ , if  $lb(v)$  is valid, we can use  $lb(v)$  to check whether it can be the next candidate node in  $SP(s, t)$  under current context.

From Lemma.3, if lower bound of node  $v$  is valid,  $|SP(v, t)| \geq lb(v) \geq d_E(v, t)$ . Then, we can get the following conclusions. For any node  $v$  with valid lower bound, if  $d_N(s, v) + lb(v) \geq ub(sp(s, t))$ , because  $ub(sp(s, t)) \geq |SP(s, t)|$ , we can get  $d_N(s, v) + lb(v) \geq |SP(s, t)|$ . Therefore, under current context,  $v$  can not be the next candidate node and it will not be push into candidate heap. Else, if lower bound of  $v$  is invalid, we can only use  $f(v) = d_N(s, v) + d_E(v, t)$  as the estimated distance of  $SP(s, t)$ . Under current context, if  $f(v) \geq ub(sp(s, t))$ , means  $f(v) \geq |SP(s, t)|$ , so the shortest path must not be via  $v$  and  $v$  will not be push into candidate heap.

Until now, we introduce IBPM with optimization which is based on utilizing the lower and upper bounds to reduce the search space. We call it IBPM with the first optimization step, denoted as IBPM-I. Next, we proposed a second optimization step which is based on reusing the previous computed shortest path.

**Lemma 4** As shown by red line in figure.6, if  $SP(s, t)$  is the shortest path from  $s$  to  $t$ ,  $(v, t)$  must be the shortest path from  $v$  to  $t$ .

**Proof:**

If  $(v, t)$  is not the shortest path from  $v$  to  $t$ , there must be another path  $P = SP(v, t)$  and  $|(v, t)| > |P|$ .  $|SP(s, t)| = d_N(s, v) + |(v, t)| > d_N(s, v) + |P|$ . There exist another access path from  $s$  to  $t$  and its length is less than  $|SP(s, t)|$ , which conflicts with our assumption. Therefore,  $(v, t) = SP(v, t)$ .  $\square$

The further optimization is based on the following conditions: the lower bound of  $v$  is valid and  $v$  is at a history shortest path ( $v \in T$ ), as shown in figure.7, the green line describes the lower bound tree  $F$  and red line describes the upper bound tree  $T$ . Node  $v$  in both  $F$  and  $T$  means that the road conditions after  $v$  have not been updated and  $(v, t)$  is still the shortest path from  $v$  to  $t$ . We can use these conditions to improve our search algorithm ulteriorly. For the current searching direction is from  $v_4$  to  $v$  and  $v_4$  is the predecessor of  $v$  in  $F$ . So,  $v_4$  will not be concerned as  $v$ 's neighbors and not be put into candidate heap  $Open\_list$ , because A\* algorithm does not concern searching backward. As the classic A\* algorithm, only four neighbors of  $v$  ( $n_1, n_2, n_3$  and  $v_5$ ) will be concerned and put into

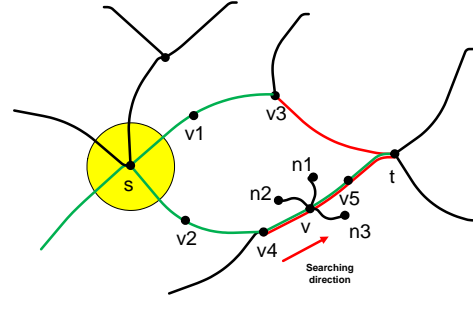


Figure 7: The Second Optimization Step

$Open\_list$ . Until now, the problem is obvious, is it necessary to put all the neighbors of  $v$  into  $Open\_list$ ?

In figure.7,  $(v, t)$  is a part of a history shortest path and the road conditions around  $(v, t)$  have not been updated. For LEMMA 4,  $(v, t)$  is still the shortest path from  $v$  to  $t$ , signed as  $SP(v, t)$ .

#### Algorithm 4: Shortest Path Determinant

---

**Data:** Lower Bound Tree  $F$ , Upper Bound Tree  $T$ ,  
Source  $s$ , destination  $t$  and  $ub(sp(s, t))$   
**Result:**  $SP(s, t)$

---

```

1 Node  $v, pre$ ;
2 Heap  $Open\_list, Close\_list$ ;
3  $Open\_list.push(s)$ ;
4 while  $Open\_list \neq NULL$  do
5   get node  $v$  from  $Open\_list$  with the least  $f(v)$ ;
6   if  $v = t$  then
7     Return  $GetShortestPath(t)$ ;
8   if  $v \in T \& F$  &  $v.pre = F.getpre(v)$  then
9      $Open\_list.push(T.getpre(v))$ ;
10  else
11    for each unscanned neighbor  $n$  of  $v$  do
12      if  $lb(n)$  is valid &  $n.pre = F.getpre(n)$  then
13        if  $d_N(s, v) + weight(v, n) + lb(n) > ub(sp(s, t))$  then
14          continue;
15      else
16        if  $f(n) > ub(sp(s, t))$  then
17          continue;
18       $Open\_list.push(n)$ ;
19   $v \leftarrow Open\_list.pop$ ;
20   $Close\_list.push(v)$ ;
21  $ShortestPath \leftarrow Temp\_path$ ;
22 Return  $ShortestPath$ ;
23 begin
24  Procedure  $Open\_list.push(n)$ 
25  if  $n \in Open\_list$  & new  $f(n)$  is not better then
26    Return;
27  if  $n \in Close\_list$  & new  $f(n)$  is not better then
28    Return;
29   $n.pre \leftarrow v$ ;
30  Remove any  $n$  in  $Open\_list$  and  $Close\_list$ ;
31   $Open\_list.add(n)$ ;
32 end
```

---

For Lemma.2, there is only one shortest path between any two nodes. So if the shortest path between  $s$  and  $t$  is via  $v$  under current context, the following nodes must be in  $SP(v, t)$ . Or there would be another shortest path from  $v$  to  $t$ , but it is conflicted with Lemma 2. Therefore, we can only concern the nodes as  $SP(v, t)$ , nor any other neighbors of  $v$ . For next searching step, only push  $v_5$ , the next node of  $v$  in  $SP(v, t)$ , into candidate heap  $Open\_list$ . Other neighbors,  $n_1, n_2$  and  $n_3$ , will not be concerned. Obviously, this optimization can improve the effect of our IBPM algorithm notably. Based on our experiment results,



comparing with A\* algorithm and naive algorithm IBPM-I, no matter time cost nor space cost, our IBPM algorithm improve the efficiency more than 10 times.

Algorithm 4 describes the whole process of the shortest path search, including all the details of our two-step optimization. Node  $v$  is the current accessing node, while node  $v.pre$  is the predecessor of  $v$  under current context (line 1). As classic A\* algorithm, we set a candidate heap *Open\_list*, and every time select the node with the least  $f(v)$  from *Open\_list* and put it into *Close\_list* (line 2,5). Search is from current source  $s$  and then we push  $s$  into candidate heap (line 3). If current accessing node  $v = t$ , means we have already find the shortest path, then current search is finished and returns the shortest path (line 6-7). If the lower bound of  $v$  is valid and  $v$  is in the upper bound tree  $T$ , we only push the next node of  $v$  in  $SP(v, t)$ , just the predecessor of  $v$  in  $T$ , into *Open\_list* and no need to concern other neighbors of  $v$  (line 8-9). That is just the second step optimization we introduce above. Else, the other neighbors of  $v$  do also need to be concerned. For each neighbor  $n$ , if the lower bound of  $n$  is valid, we compare the estimated path length,  $d_N(s, v) + weight(v, n) + lb(n)$ , with the upper bound of the shortest path, only it is less than  $ub(sp(s, t))$ ,  $n$  can be push into *Open\_list* (line 12-13,18). If the lower bound is invalid, we have to use the classic estimated length  $f(v) = d_N(s, v) + d_E(v, t)$  and compare it with  $ub(sp(s, t))$  (line 16,18). Line 12-17 is just the first step optimization. The other parts of algorithm 4 are the same as classic A\* algorithm, our optimization is based on classic A\* algorithm and according to specific application, add some restricting conditions to improve the efficiency.

## 5 Discussion

### 5.1 Memory Cost

The performance improvement to a large extents is on the cost of memory in order to reuse the previous computing results. Since the trees proposed in this work record each visited vertex only once (see section 4), the memory requirement is very small. In the worst case, the nodes need to be recorded in the memory is the size of entire network. That is the memory requirement is  $O(n)$ . For Lower Bound Tree, each node has three properties  $\{id, lowerbound, predecessor\}$  and for Upper Bound Tree, each node also has three properties  $\{id, predecessor, Euclidean Distance\}$ . Assume that each property is size 4 Bytes. In the worst case, the whole map with  $n$  nodes need to be saved in the traverse trees. So, the *Memory\_cost* =  $4Byte * 3 * n + 4Byte * 3 * n = 24nByte$ . For example, we chosen the road network of California, including 20148 nodes and under the worst conditions, every node will be stored in the traverse trees. *Memory\_cost* =  $4byte * 3 * 21048 + 4byte * 3 * 21048 = 505KB$ . This is reasonable small in most application scenarios. At the same time, the efficiency can be improved dramatically.

### 5.2 Multiple Moving Objects

As we discussed in section 1, our problem is different from single-source DSP problem due to the consideration of instant local traffic conditions. We view our problem as a single-destination DSP. When several users are moving to the same destination, they can share the tree structures proposed in the work. One scenario is in the online car racing games. Usually several (8-16) players start from the same source to the same destination. Since the shortest path instantly updated is a current best possible traveling suggestion considering the local temporary traffic conditions, the players can chose to follow or not. As a result after a period of time of the game, cars may scatter in the networks and thus their shortest paths to the destination

Table 2: Experiment setting

Name	Value
Length of query path	300 to 1500 in California Road Network, and 40 to 120 in Oldenburg Road Network.
Radius of Instant Range A	0.01 to 0.05 in California Road Network and 100 to 500 in Oldenburg Road Network. Default: 0.01 and 100
Moving Step	1 to 5 in both California Road Network and Oldenburg Road Network, Default: 1

vary. Since they share the same destination, the search spaces for them are overlapped to large extents, that is, the upper and lower bound discussed above in the overlapped region can be shared. That provides opportunity to share the tree structures among them such that the memory cost is on maintaining two tree structures only. Otherwise, if each car has its own tree structures, the memory requirement will be  $n * m$  where  $n$  is the number of cars and  $m$  is the size of trees. The efficient memory cost is critical in the application of such online games since these tree structures are kept in server side, which usually needs to support a large number of games at the same time.

## 6 Experiments

In this section, we conduct experiments on data sets of California Road Network and City of Oldenburg Road Network(stored as adjacency lists)<sup>5</sup>, which contain 21,048 nodes and 6,105 nodes respectively (see figure 9 & figure 10). All algorithms are implemented in Java and tested on a windows platform with Intel Core2 CPU (2.13GHz) and 2GB memory. The main metric we adopt is the CPU time of the monitoring process of our algorithm during the user's movement from the start to the end. Apart from this, the number of visited nodes is also a very important factor we consider as space cost. The length of query path is calculated by the number of nodes in the corresponding path. In California Road Network, the length of query path is from 300 to 1500, while in City of Oldenburg Road Network, the length of query path is from 40 to 120. The radius of Instant Range A in California Road Network is from 0.01 Longitude to 0.05 Longitude (0.01 Longitude  $\approx$  0.8 Kilometer in California road network). In City of Oldenburg Road Network, the radius is from 100 unit to 500 unit (100 unit  $\approx$  0.8 Kilometer). The moving step describes the reception of frequency of instant traffic information. In our experiments, the moving step is from 1 to 5. (*Movingstep* = 1 means when the user moves forward one node, the navigation system will receive the instant traffic information and recompute the shortest path).

The setting of our experiments is summarized in Table 2. The default setting of the radius of Instant Range A is 0.01 in the California Road Network, 100 in using the Oldenburg Road Network (both 0.01 and 100 mean approximately 800 meters, just one-minute-drive distance). The default moving step is 1, when the user arrives at each new intersection, the system will recompute the best path from current position to the fixed destination.

For the purpose of comparison, we select A\* algorithm as the basic line. In addition, a naive algorithm based on the first-step optimization of IBPM algorithm is also implemented, and we refer it as IBPM-I algorithm. In this algorithm, the way to establish, maintain and update the traverse trees is the same as that in our IBPM algorithm.

<sup>5</sup> [www.cs.fsu.edu/~lifeifei/SpatialDataset.htm](http://www.cs.fsu.edu/~lifeifei/SpatialDataset.htm)



Figure 8: California Road Network

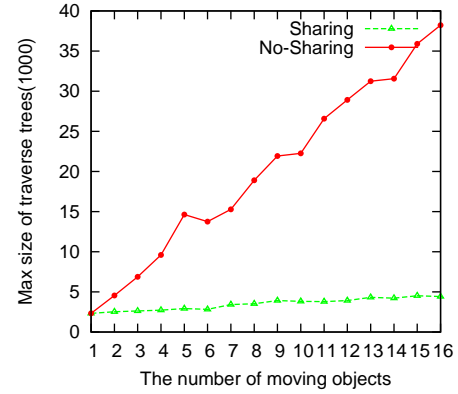


Figure 9: City of Oldenburg Road Network

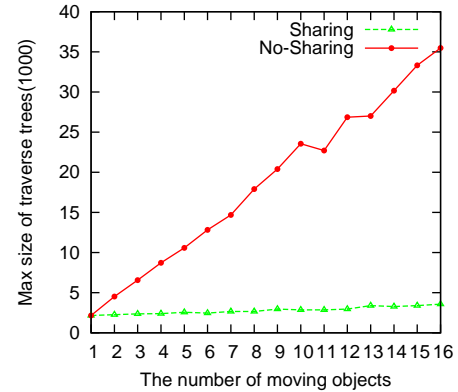
### 6.1 Effect of Path Length

First of all, we study the effect of the length of query path on CPU time and the number of visited nodes. The query path is the route on which the user travels to the destination. It is expected that the monitoring cost is in linear to the length of the path because a longer distance requires more updates and more complicated path recomputing. In this experiment, we test the performance of the query path length varying from 300 to 1200, and 40 to 120 in the California and the City of Oldenburg Road Networks respectively. The radius of Instant Range A is set to be 0.01 in California Road Network and 100 in City of Oldenburg Road Network. And the moving step is set to be 1. In the California Road Network, as shown in figure 11(a), when the path length is between 300 and 1200, the CPU time of the IBPM algorithm increases gradually from less than 1 second to nearly 3 seconds, while the time of the A\* algorithm rises dramatically to nearly 30 seconds. The time of IBPM-I algorithm is less than pure A\* algorithm, however still over 20 seconds. Figure 12(a) describes the impact of the query path length on the number of visited nodes under the default settings. Similar to the effect of the query path length on the CPU time, the number of visited nodes are much more demanding in the A\* algorithm and IBPM-I methods than in the IBPM method.

In the Oldenburg Road Network, the size of which is much smaller than California road network, the effect of IBPM algorithm is also substantially better as displayed in figure 13(a) and 14(a)). For the length of query path between 40 and 120, the CPU time of A\* algorithm starts from less than 100 ms and goes over 700ms, the IBPM-I algorithm is slightly better than A\* algorithm, though also reaching a maximum of almost 600ms. The IBPM algorithm costs only 300ms, less than half of the time of A\* algorithm. The time saving in the smaller size Oldenburg Road Network is not as apparent as in the California Road Network due to the fact that the time of maintaining and updating the traverse trees is a large proportion of total



(a) The Max Size of traverse trees in California Road Network



(b) The Max Size of traverse trees in Oldenburg Road Network

Figure 10: The Max Size of traverse trees

time.

### 6.2 Effect of Radius of Range A

The radius of Instant Range A demonstrates the range of instant traffic information being covered. Generally speaking, the larger area is covered by Instant Range A, the better the calculating result is. However, the time cost and space cost will increase similarly as the possibility of deviating from the best path from current position to the destination is higher. In this experiment, we test the performance of IBPM algorithm with the radius of Instant Range A varying from 0.01 to 0.05 in the California Road Networks and 100 to 500 in the Oldenburg Road Networks. The moving step is defaulted to be 1 again. In figure.11(b), the CPU time has been constant regardless that the radius has changed from 0.01 to 0.05. In contrast, the time cost in both A\* algorithm and IBPM-I algorithm has increased proportionally as the radius increases. Again as displayed in figure.12(b), the number of visited nodes has similar correspondence to CPU time as in figure.11(b).

For the Oldenburg Road Network in figure.13(b), the time cost in IBPM rises slightly as radius goes up from just over 200ms to 280ms, while the time cost of both A\* and IBPM-I algorithms has been more than 400ms. In figure.14(b), the number of visited nodes as required by IBPM is again much less than that by A\* and IBPM-I algorithms. It is also worth being noted that the movement of time and space cost does not change as considerably as in the example of California Road Network. The reason behind is that there are limited routes to be chosen from and the change of radius will not consequently lead to different routes in smaller size road networks.

### 6.3 Effect of Step Length K

The moving step describes the frequency of receiving the instant traffic information and recomputing the best path

from current position to the fixed destination. The bigger the moving step is, the lower frequency the information is being updated, and the less the total time cost as well as the space cost are. In California road network, the radius of instant Range A is 0.01. As can be seen from figure.11(c) and figure.12(c), the time and space cost is much less at the beginning of the travel in IBPM than in A\* and IBPM-I algorithm and this advantage has been always maintained even with the substantial decrease of time and space costs in A\* and IBPM-I algorithm with the increase of moving step from 1 to 5.

In Oldenburg Road Network (figure 13(c) and 14 (c)), we set the radius of instant Range A to be 100. The savings of time and space costs in IBPM than in A\* and IBPM-I algorithms are again clearly demonstrated.

## 6.4 Multiple Moving Objects

Multiple moving objects such as cars in the online racing games move from the same source to the same destination. The upper and lower bound trees can be shared among them. This experiment tests the impact of the number of moving objects to memory cost as shown in figure.10. The maximum number of nodes recorded in the shortest path monitoring is shown. The memory size can be inferred directly by multiplying the size of a single node (12Bytes). When the number of moving objects changes from 1 to 16, it demonstrates a constant memory requirement if they share trees. The reason is that only two trees are maintained. In contrast, if each moving object maintains its own traverse trees, the memory cost increases proportional with the number of moving objects. In this experiment, the radius is 0.01 and the moving step is 1 for California data set; the radius is 100 and the moving step is 1 for Oldenburg data set.

## 7 Conclusions

In this paper we propose two tree structures and an algorithm call Instant Shortest Path Monitoring (IBPM) in order to efficiently monitor the shortest path for an moving object when it is moving to a given destination and the local traffic condition is updated repeatedly. Our problem is different from exiting dynamic shortest path problems, i.e. single-source and all-pairs DSP. We view our problem as a single-destination with instant local traffic condition updates. The efficient processing of this problem has a wide range of applications such as online racing games and large scale traffic simulations. Comparing to simple method by running A\* algorithm over and over again, our method has dominant processing efficiency. The memory cost is linear with the nodes visited by running the A\* algorithm once.

## References

- Agrawal, R. and Srikant, R. (1995), Mining sequential patterns, in 'Proceedings of ICDE', pp. 3-14.
- Chen, Z., Shen, H. T., Xu, Q. and Zhou, X. (2009), Instant Advertising in Mobile Peer-to-Peer Networks, in 'Proceedings of ICDE', pp. 736-747.
- Chen, Z., Shen, H. T., Zhou, X. and Yu, J. X. (2009), Monitoring Path Nearest Neighbor in Road Networks, in 'Proceedings of ACM SIGMOD', pp. 591-602.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L and Stein, C. (2001), Introduction to Algorithms (second edition), The MIT Press.
- Dijkstra, E. W. (1959), A note on two problems in connection with graphs, in 'Numerische Math', 1:269-271.
- Demetrescu, C., Emiliozzi, S. and Italiano, G. F. (2004), Experimental analysis of dynamic all pairs shortest path algorithms, in 'Proceedings of SODA', pp. 369-378.
- Demetrescu, C. and Italiano, G. F. (2004), A new approach to dynamic all pairs shortest paths, in 'Journal of the ACM', vol 51(6):968-992.
- Ding, B., Yu, J. X. and Qin, L. (2008), Finding time-dependent shortest paths over large graphs, in 'Proceedings of EDBT', pp. 205-216.
- Frigioni, D., Marchetti-Spaccamela, A. and Nanni, U. (1996), Fully dynamic output bounded single source shortest path problem (extended abstract), in 'Proceedings of SODA', pp. 212-221.
- Frigioni, D., Ioffreda, M., Nanni, U. and Pasquale, G. (1998), Experimental analysis of dynamic algorithms for the single-source shortest-path problem, in 'ACM Journal of Experimental Algorithms', vol 3:5.
- Frigioni, D., Marchetti-Spaccamela, A. and Nanni, U. (2000), Fully dynamic algorithms for maintaining shortest path tree, in 'Journal of Algorithms', vol 34, pp. 351-381.
- Fu, L., Sun, D. and Rilett, L. R. (2006), Heuristic shortest path algorithms for transportation applications: state of the art, in 'Computers in Operations Research', vol 33(11), pp. 3324-3343.
- Gonzalez, H., Han, J., Li, X., Myslinska, M., and Sondag, J. P. (2007), Adaptive Fastest Path Computation on a Road Network: A Traffic Mining Approach, in 'Proceedings of VLDB', pp. 794-805.
- Hart, P. E., Nilsson, N. J. and Raphael, B. (1968), A formal basis for the heuristic determination of minimum cost paths, in 'IEEE Transactions on Systems Science and Cybernetics', pp.4(2):100-107.
- Han, J. and Pei, J. (2000), Mining frequent patterns by pattern-growth: Methodology and implications, in 'SIGKDD Explorations, Volume 2', vol.2, pp. 14-20.
- King, V. (1999), Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs, in 'Proceedings of FOCS', pp. 81-91.
- Kanoulas, E., Du, Y., Xia, T., and Zhang, D. (2006), Finding fastest paths on a road network with speed patterns, in 'Proceedings of ICDE', pp. 10-19.
- Li, F., Cheng, D., Hadjieleftheriou, M., Kollios, G. and Teng, S. (2005), On Trip Planning Queries in Spatial Databases, in 'Proceedings of SSTD', pp. 273-290.
- Mouratidis, K., Papadias, D. and Hadjieleftheriou, M. (2005), Conceptual partitioning: an efficient method for continuous nearest neighbor monitoring, in 'Proceedings of ACM SIGMOD', pp. 634-645.
- Nannicini, G., Baptiste, P., Krob, D. and Liberti, L. (2008), Fast Computation of Point-to-Point Paths on Time-Dependent Road Networks, in 'Proceeding of COCOA', pp. 225-234. Giacomo Nannicini, Philippe Baptiste, Daniel Krob, Leo Liberti
- Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U. and PrefixSpan, M.-C. Hsu. (2001), Mining sequential patterns efficiently by prefix-projected pattern growth, in 'SIGKDD Explorations, Volume 2', vol.2, pp. 14-20.
- Pallottino, S. and Scutella, M. G. (1997), Shortest path algorithms in transportation models: classical and innovative aspects, in 'Technical Report TR-97-06, 14'.

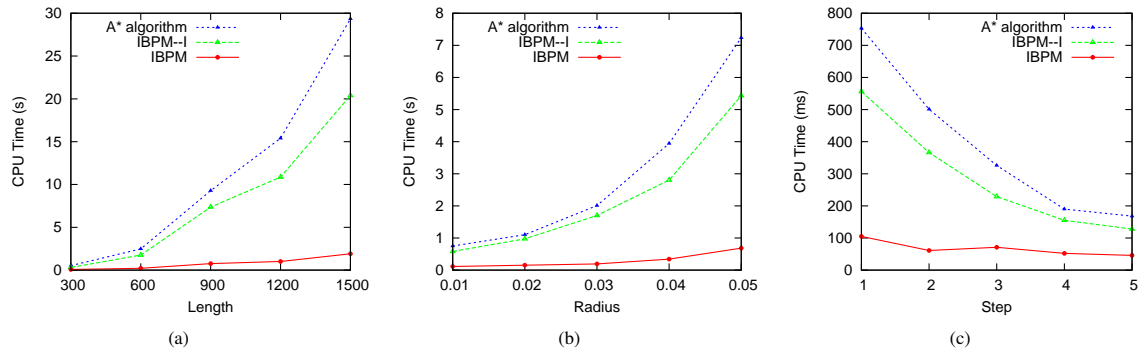


Figure 11: Performance of CPU Time in California Road Network

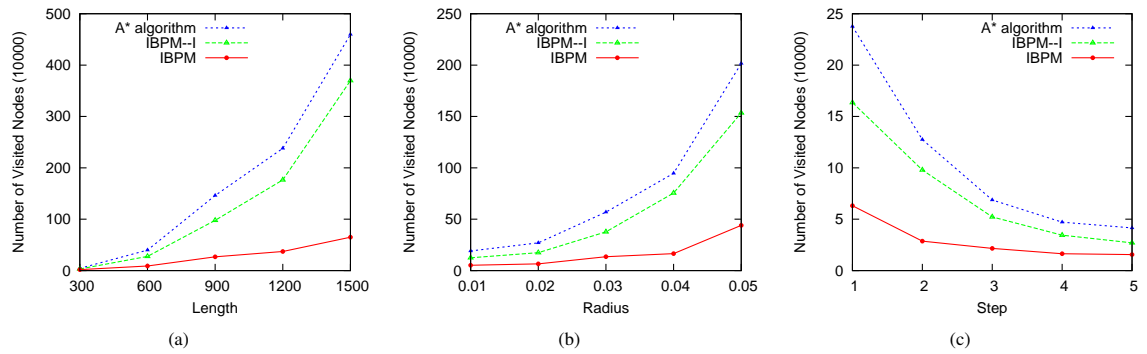


Figure 12: Performance of CPU Time in California Road Network

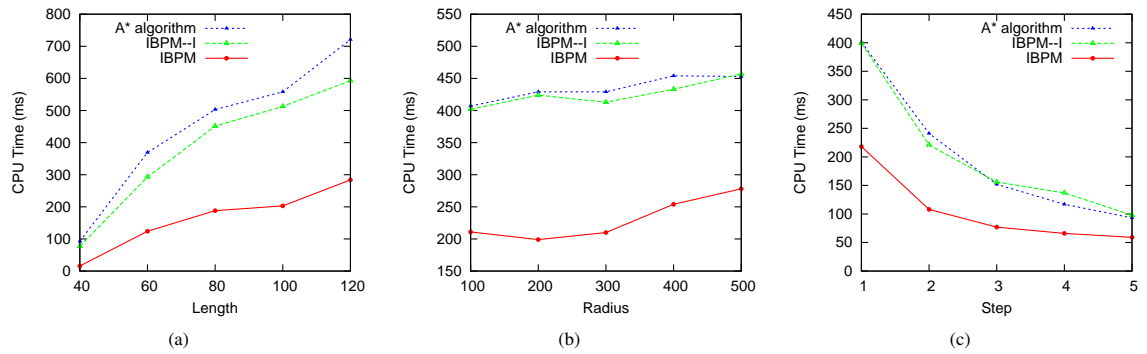


Figure 13: Performance of CPU Time in City of Oldenburg Road Network

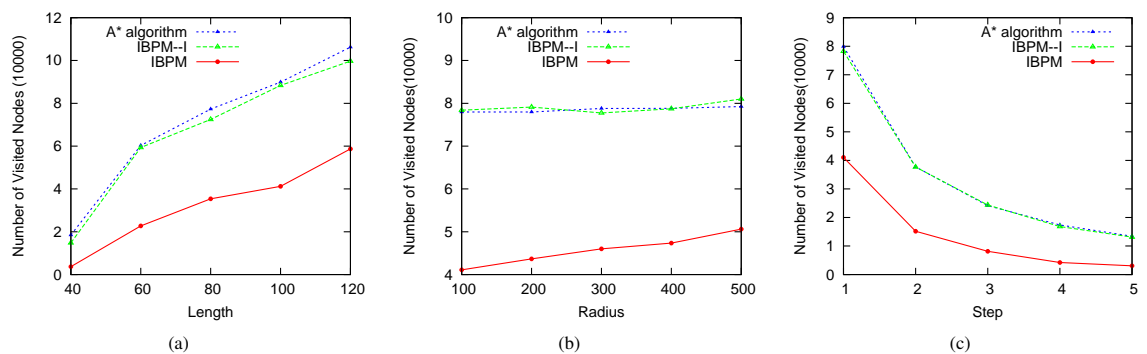


Figure 14: Performance of Visited Nodes in City of Oldenburg Road Network

# Towards Unifying Advances in Twig Join Algorithms

Nils Grimsmo

Truls A. Bjørklund

Department of Computer and Information Science  
Norwegian University of Science and Technology  
{nilsgri,trulsamu}@idi.ntnu.no

## Abstract

Twig joins are key building blocks in current XML indexing systems, and numerous algorithms and useful data structures have been introduced. We give a structured, qualitative analysis of recent advances, which leads to the identification of a number of opportunities for further improvements. Cases where combining competing or orthogonal techniques would be advantageous are highlighted, such as algorithms avoiding redundant computations and schemes for cheaper intermediate result management. We propose some direct improvements over existing solutions, such as reduced memory usage and stronger filters for bottom-up algorithms. In addition we identify cases where previous work has been overlooked or not used to its full potential, such as for virtual streams, or the benefits of previous techniques have been underestimated, such as for skipping joins. Using the identified opportunities as a guide for future work, we are hopefully one step closer to unification of many advances in twig join algorithms.

**Keywords:** Twig join, indexing, semi-structured data.

## 1 Introduction

Twig matching is the most heavily used building block for systems offering search in XML with languages like XPath and XQuery [13]. XML has become the de-facto standard for storage of semi-structured data, and the standard for data exchange between disjoint information systems. XPath is a declarative language, and XQuery is an iterative language which uses XPath as a building block. XPath queries can be evaluated in polynomial time [12].

Most academic work related to indexing and querying XML focuses on the *twig matching problem*, which is equivalent to a sub-set of XPath: Given a labeled data tree and a labeled query tree, find all matchings of the query nodes to the data nodes, where the data nodes satisfy the ancestor-descendant (a-d) and parent-child (p-c) relationships specified by the query tree edges.

The example in Figure 1 shows the relation between twig matching and XML search. The tree in part (a) is an abstraction of the XML document in (c). Real XML separates element (tag), attribute and text nodes, but in the abstract model there is only one

type of nodes. The XPath and XQuery examples in (d) both specify the same structure as the abstract twig query in (b), where double edges symbolize a-d relationships.

This work focuses on twig matching in indexed data trees. In a typical setting, all data nodes with the same label are stored together, using some encoding which specifies tree positions. To evaluate a query, one stream of data nodes with matching label is read for each query node, and are joined to form twig matches.

This paper gives a structured analysis of recent advances in twig join algorithms, which leads to the identification of a number of opportunities for further improvements. Some direct improvements are identified, such as reduced memory usage in bottom-up algorithms and stronger top-down filters. We highlight cases where new combinations of competing and orthogonal techniques would have clear advantages, but also cases where important previous work has been has been compared to unfairly in our view. We note some open challenges, such as updatability in strong structural summaries, and more efficient detection of cases where simpler and faster algorithms can be used (Section 3.7).

The analysis explores techniques for avoiding redundant computations (Section 3.1), schemes for intermediate result management (Section 3.2), top-down filters for bottom-up algorithms (Section 3.3), skipping joins (Section 3.4), refined access methods (Section 3.5) and virtual streams (Section 3.6).

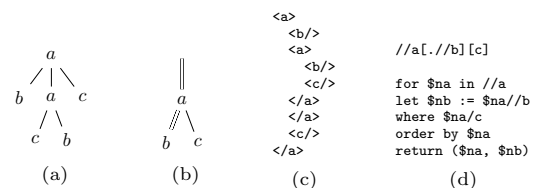


Figure 1: XML and twig matching relation. (a) Abstract data tree. (b) Twig query. (c) XML Data. (d) XPath (above) and XQuery (below).

## 2 Background: Concepts and Techniques

This section goes through some fundamental concepts and techniques which are useful for the understanding of later algorithms. First we formally define the problem.

**Definition 1** (Twig matching). Given a rooted unordered labeled query tree  $Q$  and a rooted ordered labeled data tree  $D$ , find all complete matchings of the nodes in  $Q$ , such that the matched nodes in  $D$  follow the structural requirements given by the a-d and p-c edges in  $Q$ .

First author supported by the Research Council of Norway under the grant NFR 162349.

Copyright ©2010, Australian Computer Society, Inc. This paper appeared at the Twenty-First Australasian Database Conference (ADC2010), Brisbane, Australia, January 2010. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 104, Heng Tao Shen and Athman Bouguettaya, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

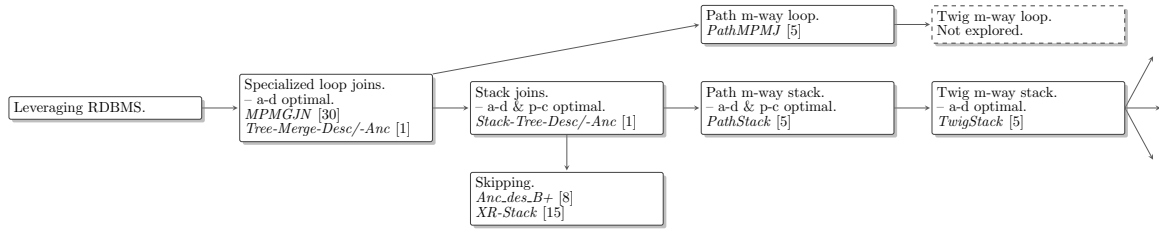


Figure 2: The early history of twig joins. Continued in Figure 8.

Note that there is a slight difference between the semantics of twig matching and XPath. A twig query returns all legal combinations of node matches, while in XPath there is a single query return node. The generality of returning all legal combinations of matches in twig matching may have been the academic focus point because it is useful for the flexibility in XQuery. XPath can also specify more than a-d and p-c relationships, but a majority of XPath queries in practice use only the a-d and p-c axis [13].

Many early approaches to search in semi-structured data used combinations of indexing and tree navigation, but the main focus the last decade has been on indexing with inverted lists and structural joins of streams of query node matches [1]. This paper only considers twig join algorithms.

**Indexing and node encoding** is critical for the efficiency of twig joins. Usually data nodes are indexed (partitioned) on node labels, using for example inverted lists. Two aspects of how data is stored inside partitions are important: How the position of a node is encoded, and how the nodes in the partition are ordered. For most algorithms nodes are stored in depth first traversal pre-order, such that ascendants are seen before descendants. The positional information which follows nodes must allow decision of a-d and p-c relationships. The most common is the regional *begin, end, level* (BEL) encoding [30], which is used in the data extraction example in Figure 3. It reflects the positions of opening and closing tags in XML (see Figure 1c). The *begin* and *end* numbers are not the same as pre- and post-order traversal numbers, but give the same sorting orders.

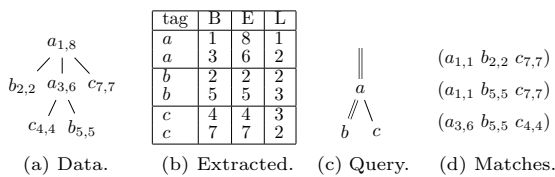


Figure 3: Tree indexing and querying example.

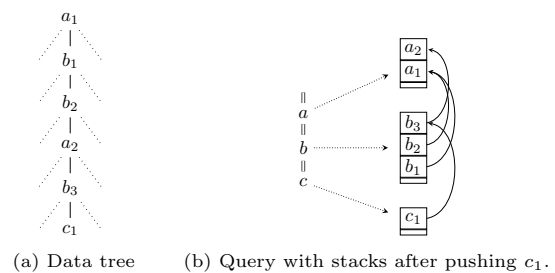
In the following, let  $T_q$  denote the stream of matches for query node  $q$ , and  $C_q$  denote the current data node in this stream. For simplicity, polynomial factors in the size of the query are ignored in asymptotic notation.

**The early history** of twig joins is shown in Figure 2. An early approach for schema agnostic XML indexing was to store nodes with BEL encoding in an RDBMS, and specify query node relations as a number of inequalities. But these theta-joins are expensive. Specialized loop structural joins which leveraged the knowledge that the data encoded is a tree where introduced [30, 1]. These have  $\mathcal{O}(I + O)$  cost for evaluating an a-d relationship, where  $I$  and  $O$  are the sizes of the input and output streams, but quadratic worst-case for p-c relationships. Stack joins were introduced to get optimal evaluation for all binary structural joins.

A problem with combining the evaluation of a number of binary relationships to answer a query, is that the intermediate results may be of size exponential in the query, even if the output is small. This lead to the introduction of multi-way join algorithms.

**Stacks are key data structures** in most modern twig join algorithms. Their use here is motivated by their use in depth first tree traversals. To join streams of ascendants and descendants, a stack of currently nested ancestor nodes is maintained. Nodes are popped off the ancestor stack when a non-contained (disjoint) node is seen in either stream. In a path or twig multi-way algorithm, there must be one stack  $S_{q_i}$  for each internal query node  $q_i$ . The matches for different query nodes must be processed in total pre-order, to ensure that ancestor nodes are added before descendants need them.

In each step in the *PathStack* algorithm [5], the current data node is used to clean all stacks by popping non-containing nodes, before it is pushed on stack. Figure 4b shows the stacks for a query when evaluated on the data in Figure 4a, right after the node  $c_1$  has been pushed. When the current query node is a leaf, all related matches are output. To enable linear time enumeration of the matches encoded in the stacks, each data node pushed onto a stack has a pointer to the closest containing data node in the parent stack, which would be the top of the parent stack as the data node was pushed. Nodes above on the parent stack cannot be ascendants, as the data nodes are read in pre-order. In the example,  $b_2$  and  $a_2$  are not usable together. Because a stack only contain nested nodes, the space needed is  $\mathcal{O}(d)$ , where  $d$  is the maximal depth of the data tree.

Figure 4: Data structures for *PathStack*.

A technique for getting path matches sorted on higher query matches first is critical for the efficiency of *TwigStack* and other twig multi-way algorithms. Delaying out-of-order output is achieved by maintaining so-called self- and inherit-lists for each stacked node [1]. The lists for the data and query in Figure 4 is shown in Figure 5.

As a node is popped off stack, the contents of its lists are appended to the inherit-lists of the node below on the same stack, if there is one. This is to maintain correct output order. See for example the lists for  $b_2$  and  $b_1$  in the example. But if the popped node can use some ancestor node in the parent stack, which the node below in its own stack cannot, the contents

of the lists must be appended to the self-lists there. This is decided from the inter-stack pointers. In the example, popping node  $b_3$  results in adding  $(a_2b_3c_1)$  to the self-list of  $a_2$ . *PathStack* has  $\mathcal{O}(I + O)$  complexity both with and without delaying output, where  $I$  is now the total input size.

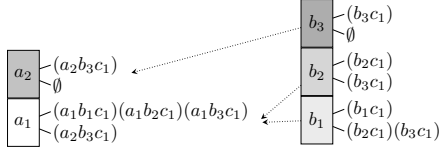


Figure 5: Stack nodes with final self- and inherit lists for the data and query in Figure 4. Darker nodes popped first.

**TwigStack** [5] was the first holistic twig join algorithm. Using *PathStack* on each root-to-leaf path in a twig query and merging the matches, may lead to many useless intermediate results, because path matches need not be part of complete matches. *TwigStack* improved on this, and achieved  $\mathcal{O}(I + O)$  complexity for queries with a-d edges only.

It is a two-phase algorithm, where the first phase outputs matches for each root-to-leaf path, and the second phase merge joins the path matches. The first phase does two things which are critical for the performance of the algorithm: It only outputs path matches which possibly are part of some complete query match, and outputs paths sorted on higher query nodes first, using the technique from [1]. This allow a linear merge in phase two.

*TwigStack* does additional checking before pushing nodes on stack compared to *PathStack*. The data node at the head of the stream for a query node  $q$  is not pushed on stack before it has a so called “solution extension”, which means that the heads of the streams of all child query nodes are contained by  $C_q$ , and that child nodes recursively satisfy this property. Also, a node is not pushed on stack unless there is a useable ancestor data node on the stack for the parent query node.

Pseudo-code for *TwigStack* is shown in Algorithm 1 (adapted from [5]). It is included here to ease the depiction of the improvements discussed in the following sections. Each query node  $q$  has an associated stream  $T_q$  with current element  $C_q$ , and a stack  $S_q$ . The algorithm revolves around a recursive function *getNext*( $q$ ), which returns a (locally) uppermost query node in the subtree of  $q$  which has a solution extension. If the parent of the returned  $q$  has a usable ancestor data node on stack, this means  $C_q$  is part of a full solution extension identified earlier, and  $C_q$  is pushed on  $S_q$ . A path match is found when a leaf node is pushed on stack, but output is delayed to make sure paths are ordered on the query nodes top down (called “blocking” in [5]). Note that actually pushing a leaf node on stack is unnecessary, as it will be popped right off.

The *getNext*() traversal is bottom up, and is short cut if some node does not have a solution extension (see line 20). Leaves trivially have solution extensions. The traversal has the side effect of advancing the treated query node at least until it contains all its children (line 23). If it does not contain all children at this point, the child currently with the first pre-order data node (lowest *begin* value) is returned to be forwarded in line 12.

Figure 6 shows the state of the algorithm when evaluating the query in Figure 3c, right after node  $b_{5,5}$  has been processed. After the first call to *getNext*( $a$ ), when all the streams were at their start position,

### Algorithm 1 TwigStack

```

1: function TwigStack( $Q$ )
2:   while not atEnd( $Q$ )
3:      $q := \text{getNext}(Q.\text{root})$ 
4:     if not isRoot( $q$ )
5:       cleanStack( $S_{\text{parent}(q)}, C_q$ )
6:       if isRoot( $q$ ) or not empty( $S_{\text{parent}(q)}$ )
7:         cleanStack( $S_q, C_q$ )
8:         push( $S_q, C_q, \text{top}(S_{\text{parent}(q)})$ )
9:         if isLeaf( $q$ )
10:          outputPathsDelayed( $C_q$ )
11:          pop( $S_q$ )
12:         advance( $T_q$ )
13:   mergePathSolutions()

14: function getNext( $q$ )
15:   if isLeaf( $q$ )
16:     return  $q$ 
17:   for  $q_i \in \text{children}(q)$ 
18:      $q_j := \text{getNext}(q_i)$ 
19:     if  $q_j \neq q_i$ 
20:       return  $q_j$ 
21:    $q_{\min} = \min \arg_{q_i \in \text{children}(q)} \{C_{q_i}.\text{begin}\}$ 
22:    $q_{\max} = \max \arg_{q_i \in \text{children}(q)} \{C_{q_i}.\text{begin}\}$ 
23:   while  $C_q.\text{end} < C_{q_{\max}}.\text{begin}$ 
24:     advance( $C_q$ )
25:   if  $C_q.\text{begin} < C_{q_{\min}}.\text{begin}$ 
26:     return  $q$ 
27:   else
28:     return  $q_{\min}$ 

```

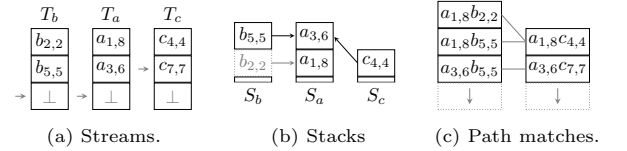


Figure 6: TwigStack state when evaluating query in Figure 3c, after processing node  $b_{5,5}$ .

$a$  itself was returned as it had a solution extension, and  $C_a = a_{1,8}$  was pushed on stack. For the second call to *getNext*( $a$ ), this was not the case, and  $b$  was returned, with head  $C_b = b_{2,2}$ . Since  $b_{2,2}$  had a usable ancestor  $a_{1,8}$  on the parent stack  $S_a$ ,  $a_{1,8}$  must have had a solution extension, in which the subtree rooted at  $b_{2,2}$  was usable. So  $C_b = b_{2,2}$  was pushed on its own stack  $S_b$ , and since it was a leaf, the path matching  $(a_{1,8}, b_{2,2})$  was output. After all paths have been found they are merge joined.

**TwigStack suboptimality** for mixed a-d and p-c queries comes from having to output path matches without knowing whether the data nodes used can satisfy all their p-c relationships. The algorithm cannot always decide this from the nodes on the stacks and the heads of the streams. For the example in Figure 7 it cannot be decided if the path matches  $(a_1, b_1), \dots, (a_1, b_n)$  are part of a full match before the node  $c_{n+1}$  is seen.

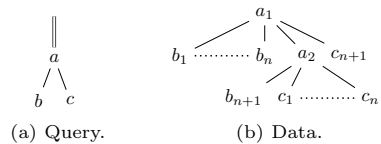


Figure 7: Bad case for *TwigStack*.

For a-d only queries, queries where a p-c edge never follows an a-d edge [25], or on data with non-recursive schema [10], twig joins can be solved with linear cost in the size of the input and the output, using  $\mathcal{O}(d)$  memory. Sadly, recursive schema, where nodes with a given label may nest nodes with the

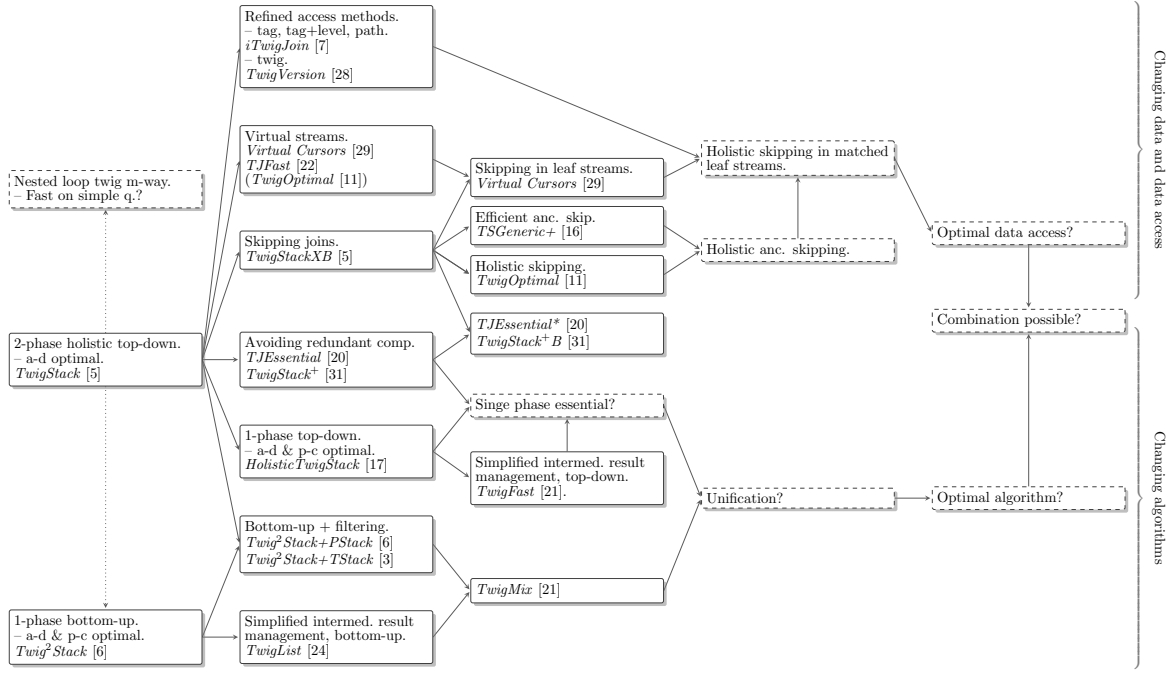


Figure 8: Advances and opportunities in twig joins.

same label, are common in XML in practice [9], and so are mixed queries of the type mentioned above. No algorithm can solve the general problem given tag streaming, linear index size, and a query evaluation memory requirement of  $\mathcal{O}(d)$  [25]. One alternative is storing multiple sort orders on disk, instead of only tree pre-order. This would require  $\Omega(m^{\min m, d} \cdot D)$  disk space in the worst case, where  $m$  is the number of structurally recursive labels and  $D$  is the size of the document [10]. Another alternative is to do multiple scans of the streams, but this would require  $\Omega(d^t)$  passes in the worst case, where  $t$  is a linear metric on the complexity of the query [10]. So, the only viable alternatives left seem to be relaxing the  $\mathcal{O}(d)$  space requirement, or using something different than tag partitioning. The following section investigates this, but also many practical speedups to *TwigStack*.

### 3 Advances

A multitude of different improvements have been presented after the introduction of *TwigStack*. Figure 8 gives an overview of these, with a separation between improved join algorithms and changes to how data is indexed and accessed. The rest of this paper is devoted to a structured review of these advances. Our goal is to identify further improvements, and to shed light on whether it is likely that combining these advances is possible and beneficial.

#### 3.1 Avoiding Redundant Computation

*TwigStack* may perform many redundant checks in the calls to *getNext()*. Each time a node is returned, the full subtree below has been inspected. The *TJEssential* [20] algorithm improved three specific deficiencies, exemplified in Figure 9.

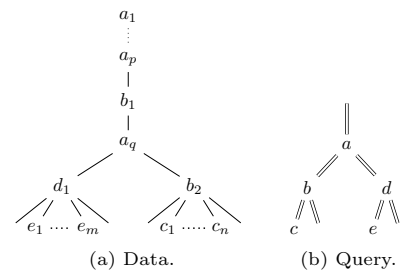
The first deficiency is from self-nested matching nodes. For query node  $a$  and data nodes  $a_1$  to  $a_p$  in the example, it is unnecessary to recursively check the full subtrees below  $b$  and  $d$  in each round while pushing the nodes onto  $S_a$ . The usefulness of  $a_2, \dots, a_p$  can be seen from the fact that  $a_1$  had a new solution extension, and that  $a_2, \dots, a_p$  contains  $b_1$  and  $d_1$ , the heads of the streams of  $a$ 's children.

The second observation is on the order in which child nodes are inspected. If the child  $b$  is inspected before  $d$  in line 18 of Algorithm 1, *getNext(a)* will call *getNext(b)* before *getNext(d)* shortcuts the search. There will be  $m - 1$  redundant calls *getNext(b)* while forwarding the leaf node  $e$ .

The third observation is that many useless calls could be made after a stream has reached its end. Assuming that  $b_2$  was the last  $b$ -node in Figure 9, no  $a$  node later in the tree order would ever be pushed onto stack, and  $T_a$  could be forwarded to its end. Also, if  $S_b$  was empty, any descendant of  $b$  in the query could have their stream forwarded to the end, as the remaining nodes could not be part of a solution.

*TJEssential* is a total rewrite of *TwigStack*, and is more complex than the original algorithm. *TwigStack+* [31] is a less involved modification, which only changes the *getNext()* procedure, such that it does not return before a solution is found. *TwigStack+* does not catch any of the tree above cases, but reduces computation for scattered node matches in practice.

**Opportunity 1** (Removing redundant computation in top-down one-phase joins). The improvement of *TwigStack+* can trivially be ported to recent algorithms such as *HolisticTwigStack* and *TwigFast*, which improve other aspects of *TwigStack* (see Section 3.2). A challenge is to do the same for all the three improvements of *TJEssential*. Also, case three above could be extended to more efficient aligning for multi-document XML collections.

Figure 9: Giving redundant checks in *TwigStack*.



### 3.2 Top-down vs. Bottom-up

There are two main lines of algorithmic improvements over *TwigStack* which give optimal evaluation of mixed a-d and p-c queries by relaxing the  $\mathcal{O}(d)$  memory requirement: Bottom-up algorithms which read nodes in post-order, and later algorithms which go back to top-down and pre-order. Differences between these are illustrated in Figure 10.

**Twig<sup>2</sup>Stack** [6] generates a *single combined* stream with post-order sorting for all query node matches with the help of a single stack. With post-order processing it can be decided if an entire subtree has a match at the time the top node is seen.

Figure 10c shows the *hierarchies of stacks* built while processing a query. For each query node, a list of trees of stacks is maintained. A data node strictly nests all nodes *below* in the stack, and all nodes in child stacks in the tree. The lists of trees are stored sorted in post-order, and are linked together by a common root if an ancestor node is processed. From the post-order, the nodes to be linked will always be found at the end of the list, and the new root will always be put at the end. The order naturally maintains itself, and good locality is achieved.

Instead of each node on stack having a pointer to an ancestor node on a parent stack as in *TwigStack*, each stacked data node has for each related child query node, a list of pointers to top stack nodes matching the query axis relationship. Nodes are only added if a-d and p-c relationships can be satisfied, and p-c pointers are only added when levels are correct, as seen for the *a* and *c* nodes in the example.

**TwigList** [24] is a simplification of *Twig<sup>2</sup>Stack* using simple lists and intervals given by pointers, which improves performance in practice. For each query node, there is a post-order list of the data nodes used so far. Each node in a list has, for each child query node, a single recorded interval of contained nodes,

as shown in Figure 10d. Interval start and end positions are recorded as nodes are pushed and popped on and off the global stack. All descendant data nodes are processed in between. Compared with the list of pointers in *Twig<sup>2</sup>Stack*, enumeration of matches is not as efficient for p-c edges, but sibling pointers can remedy this.

**HolisticTwigStack** [17] is a modification of *TwigStack* which uses pre-order processing, but maintains complex stack structures like *Twig<sup>2</sup>Stack*. The argument against *Twig<sup>2</sup>Stack* was a high memory usage, caused by the fact that all query leaf matches are kept in memory until the tree is completely processed, as they could be part of a match. *HolisticTwigStack* differentiates between the top-most branching node and its ancestors, for which a regular stack is used, and lower query nodes, which have multiple linked lists of stacks, as shown in Figure 10e. Each query node match has one pointer to the first descendant in pre-order for each child query node. For “lower” query nodes, new data nodes are pushed onto the current stack if contained, otherwise a new stack is created and appended to the list. As a match for an “upper” query node is popped, the node below on stack must inherit the pointers. Node *a*<sub>1</sub> would inherit the pointers from both *a*<sub>2</sub> and *a*<sub>4</sub> in the example in Figure 10e, and the related lists of child matches would be linked.

**TwigFast** [21] is a simplification of *HolisticTwigStack* similar to *TwigList*. There is one list containing matches for each query node, naturally sorted in pre-order, and data nodes in the lists have pointers giving the interval of contained matches for child query nodes, as shown in Figure 10f. Each data node put into the list has a pointer to its closest ancestor in the same list, and there is a “tail pointer”, which gives the last position where a node can be the ancestor of following nodes in the streams. These pointers are used for the construction of the intervals.

**Different advantages** of top-down and bottom-up algorithms can be seen in Figure 10. A top-down algorithm can avoid storing *b*<sub>1</sub> and *c*<sub>2</sub>, while a bottom-up algorithm is unable to decide that these nodes cannot be part of a solution. On the other hand, a bottom-up algorithm can decide that *a*<sub>2</sub> is not usable, because it cannot satisfy the p-c relationship between *a* and *c*. Both approaches can decide that *a*<sub>3</sub> is not useful because it does not have a *b* descendant.

The worst case space complexity of twig pattern matching is an open problem, and the known bounds are  $\Omega(\max d, u)$  and  $\mathcal{O}(I)$ , where *u* is the number of nodes which are part of a solution [25]. However, practical space savings are possible.

**Opportunity 2** (Top-down memory usage). *TwigStack* treats queries as a-d only in the stack construction part of phase one. A node returned from *getNext()* is pushed on stack if it has a usable ancestor on the parent stack, even if the query specifies a p-c relationship. For example does not *c*<sub>3</sub> have to be pushed on stack in Figure 10e, because it does not have a usable parent. Strictly checking p-c relationships before adding intermediate results would reduce memory usage in practice. This optimization was identified for *TJFast* [22] (see Section 3.6), but the later *HolisticTwigStack* and *TwigFast* do not take advantage of this opportunity.

**Opportunity 3** (Bottom-up memory usage). Assume a query node *q* with a p-c relationship to the parent query node. If a candidate match for *q* is pushed onto a stack in *Twig<sup>2</sup>Stack*, and the data node below on the stack does not have an incoming pointer, this means the node below will never get a matching parent, and can be popped off stack. For example

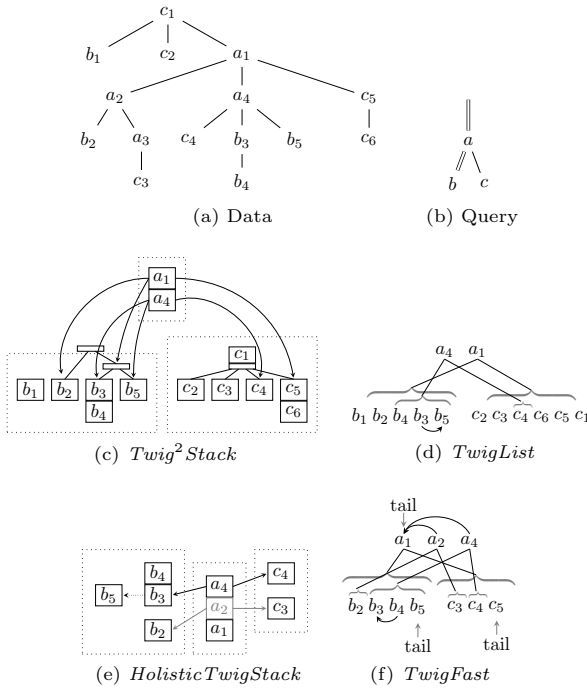


Figure 10: (a) Data. (b) Query. (c) Hierarchies of stacks for *Twig<sup>2</sup>Stack*. (d) Intervals for *TwigList*. Curved arrows are sibling pointers. (e) Lists of stacks for *HolisticTwigStack* right before *c*<sub>5</sub> is processed. Previously popped nodes shown in gray. (f) Intervals for *TwigFast* after *c*<sub>5</sub> has been processed. Curved arrows are ancestor pointers.

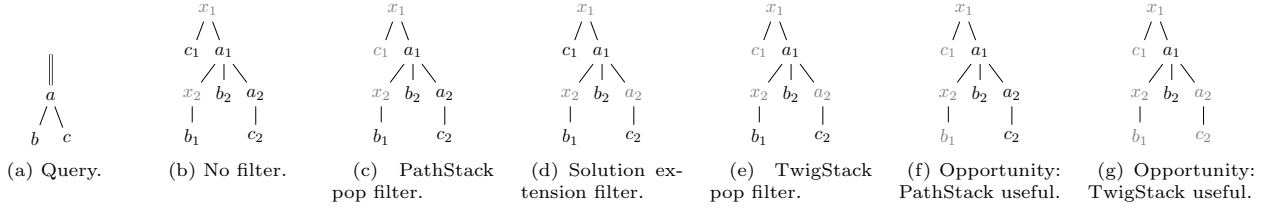
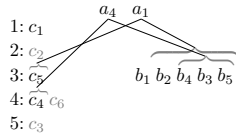


Figure 11: Filtering approaches for bottom-up up processing. Filtered nodes shown in gray.

could the node  $c_6$  be dropped in Figure 10c. Also, when stack trees for  $q$  are merged, some ancestor data node  $a_i$  is seen. Then all the stack trees which do not get or have an incoming pointer can be dropped, as all later candidates for the parent query node will be after in the post-order. In the example, the stack trees containing the single nodes  $c_2$  and  $c_3$  could be dropped when  $a_1$  is seen.

Note that improvements on this is hard to transfer directly to *TwigList* unless the lists are implemented as linked lists. But this is by far inferior to using arrays and array doubling on modern hardware, as done in *TwigList* [23]. Another solution is to keep one list for each level for query nodes which have a p-c relationship to the parent query node. Siblings would then be stored contiguously, and interval pointers would implicitly be to a list on a given level. When the first ancestor of a segment of nodes in need of a parent is seen, the useless nodes can be *over-written*. This modification would also make sibling pointers unnecessary and improve efficiency of result enumeration. Figure 12 shows the proposed approach for the data and query in Figure 10, where gray list items can be overwritten.

Figure 12: Proposal for multi-level lists for *TwigList*.

### 3.3 Filtering

Low memory top-down approaches have been used as filters to bottom-up algorithms to reduce space usage by avoiding useless nodes. Note that this does not result in a perfect solution. Assume that node  $a_1$  in Figure 10a had a different label. A  $\mathcal{O}(d)$  space top-down pre-order approach could not decide that  $b_2$  in the example was not part of a match, and a bottom-up algorithm would have to keep it in memory until the entire tree was read. Figure 11c-e shows the effects of different previously proposed filters. **PathStack Pop Filter.** In the original *Twig<sup>2</sup>Stack* paper [6], *PathStack* was proposed as a pre-filter to allow early result enumeration. *PathStack* is run as usual, but without its result enumeration. As disjoint nodes are popped off their stacks, they are passed to *Twig<sup>2</sup>Stack*. When the bottom node is popped from the stack of the root query node, all results can be output, and the hierarchical stacks destroyed. A side effect of this procedure is that only nodes that are part of some prefix path match are used (these are not necessarily part of a full root-to-leaf path match). In Figure 11c, node  $c_1$  is avoided. Note that one data node may result in the popping of multiple nodes on multiple stacks, and that *Twig<sup>2</sup>Stack* must receive descendants before ascendants.

**Solution Extension Filter.** *TwigMix* [21] is an algorithm which combines the simplified data structures in *TwigList* with the *getNext()* function from *TwigStack* as a filter. This combination gives efficient evaluation for queries involving p-c edges, and reduced memory usage in practice. An advantage of this approach over *Twig<sup>2</sup>Stack+PathStack* is that there is no overhead of maintaining an extra set of stacks, and that internal nodes are filtered holistically. The downside is that nodes are added without even having a possible parent or ancestor. Figure 11d shows that node  $a_2$  is filtered, because it never has a solution extension (misses  $b$  node below), while nodes  $c_1$  and  $b_1$  are not filtered.

**TwigStack Pop Filter.** *TwigStack* can also be used as a filter for *Twig<sup>2</sup>Stack* [3]. A node is never added to the hierarchical stacks if it is not popped from a top-down stack in *TwigStack*. As a node is never pushed on stack if it does not have a usable ancestor, which again has a solution extension, this gives additional filtering, at the cost of maintaining the top-down stacks. Figure 11e shows the improvements both over *PathStack* and solution extension as filters. An issue is that *Twig<sup>2</sup>Stack* expects the stream of nodes to be in post-order, and that *TwigStack* may pop nodes off stacks out of this order. When a node is returned from *getNext()*, only the related stack and the parent stack are inspected. Also, *TwigStack* does not keep leaf matches on stack, but nested leaf matches may arrive later. In [3] this is solved by keeping an extra queue of data nodes into which popped nodes are placed if the algorithm decides later popped nodes may precede them.

A different solution could be to allow nested nodes on query leaf stacks, and to inspect all stacks when popping disjoint nodes to ensure post-order, as with the *PathStack* filter. Also, *Twig<sup>2</sup>Stack* actually does not need to see nodes in strict post-order, but only to see descendants before ascendants. Hence, not all stacks in the query would have to be inspected, only ascendant and descendant stacks of the current node.

**Opportunity 4 (Stronger filters).** There are further possibilities for filters with  $\mathcal{O}(d)$  space usage. Instead of using all nodes popped off stacks in *PathStack*, one could use the nodes which would be *used* in full path match. As leaf nodes are pushed on stack, a simplified enumeration algorithm could be run, tagging nodes which take part in solutions. As can be seen in Figure 11f, this is an improvement over the previous *PathStack* filter, but only partially over the solution extension filter, which to a greater extent filters matches for higher query nodes. Leaves trivially have solution extensions. The “*PathStack useful*” filter works well on lower query nodes. Note that as the bottom-up algorithms to a greater extent handle upper nodes themselves, a filter is of most use if it removes lower query node candidates effectively. An even stronger filter would be to only use nodes which would have been output as parts of path matches in *TwigStack*, as shown in Figure 11g. None of [6, 21, 3] compare with using any other type of filter. A thorough comparison should compare both the practical

space reductions filters give, their absolute costs, and how their use affect the total computational cost.

**Opportunity 5** (Unification or assimilation). When comparing absolute performance gains presented in the respective papers, *TwigFast* is the winner on performance for pure tag streaming. As this is a very important result, it should be verified independently. Before *TwigFast* is picked as the method of choice, at least the following should be answered: (i) Can the improvements discussed in Section 3.1 be applied? (ii) Is it superior to improved top-down and bottom-up combinations? (iii) Does the picture change when random access gets more expensive compared to computation? [21] does not comment on the spatial locality of memory access patterns in the intermediate result data structures in *TwigFast*, while they are very good for *TwigList* [24].

### 3.4 Skipping Joins

Skipping is a useful technique when the streams to be joined have very different sizes. Skipping is used to jump forward in a stream to avoid reading and processing parts of the streams which cannot contain useful nodes. Figure 13 shows cases where different skipping techniques and data structures can be used.

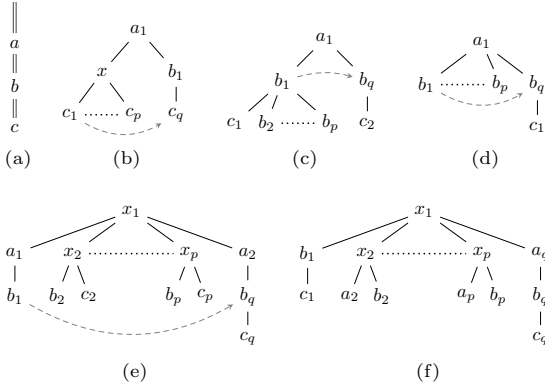


Figure 13: Benefits of skipping techniques. (a) Query. (b) Descendants easily skipped with B-tree. (c) Skip past discarded ascendant. (d) XR-tree needed to skip ascendants. (e) Holistic skipping preferred. (f) Holistic skipping with XR-tree needed.

**Simple B-tree skipping** can be used to skip in descendant streams, and to some extent in ancestor streams. It is trivial to skip in the descendant stream to find the first possible contained node, which is the first node with a larger *begin* value. In Figure 13b,  $T_c$  is forwarded from  $c_1$  to  $c_q$  to find the first possible ascendant of  $b_1$ .

But skipping to find the next ascendant of a node using the same approach is not effective, as any node with a lower *begin* value may be a match. A trick for ancestor skipping was introduced in [8]. If a node  $b_i$  is popped off stack  $S_b$  due to disjointness with the current data node in some query node,  $T_b$  is forwarded to the first node not contained by the popped node, a  $b_j$  such that  $b_i.end < b_j.begin$ . An example of this can be seen in Figure 13c. If  $c_2$  pops  $b_1$  off stack,  $T_b$  can be forwarded beyond  $b_1$  to  $b_q$ , because no descendant of  $b_1$  could be useful.

**XR-trees** enable ancestor skipping in the general case [15]. Figure 13d shows an example where the above trick cannot be used. The XR-tree is B-tree variant which can retrieve all  $R$  ancestors or descendants of a node from  $N$  candidates in  $\mathcal{O}(\log N + R)$  time. Typically one tree is built for each tag. To find all  $a$  ascendants of a node  $d_k$ , find the node  $a_i$

with the nearest preceding *begin* value, and then all  $a$  ascendants of  $a_i$  in the XR-tree for  $a$ . Conceptually the XR-tree contains for each node, the list of ascendants, which gives quadratic space usage when implemented naively. Linear space usage is achieved by not storing information redundantly internally in XR-tree nodes, and by storing common information in internal XR-tree nodes.

*TSGeneric+* (also called *XRTwig*) [16] extends the use of the XR-tree to *TwigStack*, and does two major modifications to the algorithm. The first is to skip forward to containment of the first child in the *getNext()* procedure (see line 23 in Algorithm 1). The second change is more involved. Before calling *getNext()* on all children in line 18, a “broken” edge in the query sub-tree is repeatedly picked, and the two related nodes are “zig-zag” forwarded until they match. This is only done if the query node does not have data nodes on the stack. Choosing which edge to fix is either top-down, bottom-up or by statistics.

**Holistic skipping** was introduced in the *TwigOptimal* [11] algorithm, which uses B-trees. Figure 13e shows a case where the approach from *TSGeneric+* would be very expensive, reading all nodes  $b_2-b_p$  and  $c_2-c_p$  to fix the edge between  $b$  and  $c$ . *TwigOptimal* processes the query bottom-up then top-down. In the bottom-up phase, nodes are forwarded to contain their descendants, and in the top-down phase, nodes are forwarded until they are contained by their parent. To avoid as many data structure reads as possible, nodes are forwarded to “virtual positions”, which have only *begin* values. When a full traversal did not forward any node, the node with the minimal current *begin* value is forwarded to a real data node.

The name of the *TwigOptimal* algorithm may be slightly misleading, as the optimality is given skip structures on *begin* values only. Only *TSGeneric+* using simple B-trees is compared with. The effects of the two contributions, holistic skipping and the virtual positions, are not separately tested. *TwigOptimal* would not be efficient neither on the example in Figure 13c nor 13d. The approach is best when there are more matches for lower query nodes. An common exception from this is queries with leaf value predicates in XML. [11] mentions skipping to the closest ancestor and then backtracking to the first ancestor as a possible practical speed-up.

**Opportunity 6** (Holistic effective ancestor skipping). Figure 13f shows a case where both *TSGeneric+* with XR-trees and *TwigOptimal* would fail to be efficient. The former would zig-zag join  $a_2-a_p$  and  $b_2-b_p$ , and the latter would be unable to forward  $T_a$  to  $a_q$  without checking at least all of  $a_2-a_p$  for ancestry of  $c_q$ . Combining holistic skipping and data structures for efficient ancestor skipping is required in a robust solution.

**Opportunity 7** (Simpler and faster skipping data structures). The XR-tree is a dynamic data structure which supports insertions and deletions [15]. In regular keyword search engines, simpler data structures are usually preferred to the heavier B-trees when the data is static or semi-static. Similar simpler data structures should also be created for efficient ancestor skipping. If their use is still expensive, techniques similar to the trick used to skip past discarded ascendants should be applied when possible.

### 3.5 Refined Access Methods

There are alternatives to indexing and accessing data by node labels, such as using label and level, or the root-to-node path strings of labels (called *tag+level* and *prefix path streaming* [7]). With refined partitioning some method must be used to identify the

useful partitions for each query node. For prefix path streaming this would be the partitions with data paths matching the root-to-node downward paths in the query.

**Structural summaries** are directory structures used to classify nodes based on their surrounding structure. They were first used in combination with tree traversals, but have later been integrated with pure partitioning schemes [19]. The most common is a *path summary*, which is a minimal tree containing all unique root-to-node label paths seen in the data. The data nodes associated with a summary node is called the extent of the node. Figure 14a shows the path summary for the data in Figure 14b, and the extents are shown in Figure 14d.

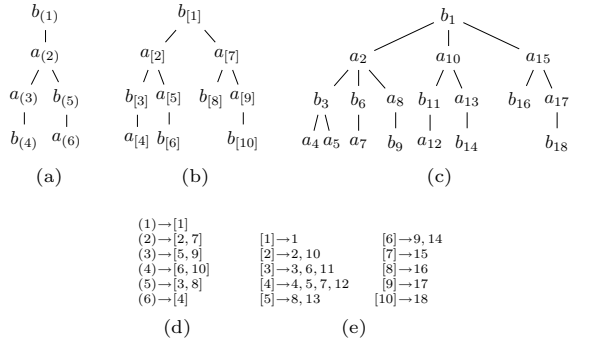


Figure 14: Structural partitioning example. (a) is path summary for (b), with extents shown in (d). (b) is F&B summary for (c), with extents shown in (e).

Many alternative summary structures have been devised for general graphs. A structure which is also directly useful for trees is the stronger F&B-index [18], where two nodes are in the same equivalence class if they have the same prefix path, *and* have children of the same equivalence classes. In the example, (b) is the F&B summary of the tree in (c). For graphs, the F&B index can be found in  $\mathcal{O}(m \log n)$ , where  $m$  and  $n$  are the number of edges and nodes in the graph. It is not known whether the F&B index can be found more efficiently for trees.

**Opportunity 8** (Updates in stronger summaries). Simple path summaries are usually small, and are easily updateable. When traversing a data tree for indexing, the path summary is used as a deterministic automaton, where new nodes are added on the fly when needed. Data nodes can be put in the correct extents immediately. If a data tree is updated, only the data nodes whose extent changes are affected. An interesting question is the updatability of stronger structural summaries. In the worst case for the F&B index, the structure of an entire *containing subtree* below the global root could change if a data node is added or removed, by causing or removing equivalence with another subtree. What are the implications of strategies lessening the restrictions on the F&B index? Would this give critical “fragmentation effects” in practice? And are updates cheaper in coarser variants, such as the F+B-index [18]?

**Opportunity 9** (Hashing F&B summaries). In some search scenarios, there are many small documents, which are parts of a virtual tree. Document updates can be implemented as document deletes and inserts. With simple path summaries, documents can be added with cost linear in the document size, by traversing the summary deterministically. However, more refined summaries are not deterministic. Are stronger summaries like the F&B index suitable in

this model? A challenge is that matching a new document in the F&B index has cost linear in the size of the *summary* in the worst case, not the document. Assume now that  $a_2$ ,  $a_{10}$  and  $a_{15}$  in Figure 14c are document roots. The structure of each document is classified by a node on depth two in the F&B summary in Figure 14b. If a new document is added below  $b_1$ , it will either have the structure defined by  $a_{[2]}$  or  $a_{[7]}$ , or a new subtree will be added below  $b_{[1]}$ . One possibility is to index the F&B summary by *hashing* each level 2 subtree, as these represent full document structures. When a new document is indexed, a summary of the document structure can be built and hashed, to identify a match in the global F&B index.

*TwigVersion* [28] is a twig matching approach which introduces a novel two-layer indexing scheme, with an F&B summary of the data, and a path summary of the F&B summary. This reduces the expense of matching in the F&B index. But as they only compare to twig join algorithms which do not use structural summaries, and also introduce many other ideas, it is hard to assess the usefulness of the two-layer approach itself. They compare their two layer approach with a pure F&B index, but do not state how they search in it.

A common way to *use* path summaries is to match each individual root-to-leaf path, and *prune* away matches which cannot be part of a full match [7, 2]. Another solution, which is more robust for large path summaries, is to label partition the summary and run a full twig join algorithm on it. In [3] a novel combination of *Twig<sup>2</sup>Stack* and *TwigStack* is used for matching in large path summaries (see Section 3.3).

**Opportunity 10** (Exploring summary structures and how to search them). Many twig join algorithms have leveraged the benefits of path summaries. Stronger summaries like the F&B index are not as commonly used, maybe because of worst case size and implementational complexity. Using different algorithms to search various types and combinations of summaries has not been thoroughly explored. An evaluation should address the total benefits of different single- and multi-level strategies, but also detail the local cost of specific matching methods in specific summary types of different sizes.

**Multi-stream access** may be required for a single query node when partitioning on path, as there may be many path matches. One solution is to merge the streams. Another is to have a tag partitioned store, and filter the data nodes on matching path ID [29]. A speedup to this approach is to *chain* together nodes with the same path [19]. This is also useful when indexing text nodes on value and integrating structure information.

*S<sup>3</sup>* [14] is a twig matching system which takes all possible combinations of individual streams matching query nodes based on prefix path, and solves each combination separately, merging the results of each evaluation. This approach does not give polynomial worst-case bounds.

**Blocking** is the reason for the sub-optimality of *TwigStack*. When partial matches must be output to evaluate the data and query in Figure 15a, it is because  $b_1$  and  $c_1$  blocks each others access to  $c_2$  and  $b_2$  respectively.

Using more fine grained partitioning and streaming solves some blocking issues, because there are multiple heads of streams. A partitioning which refines another always inherits the reduced blocking [7]. In *tag+level streaming* there is no blocking when p-c edges only are used below the query root [7]. But in mixed queries, such as in Figure 15b, blocking can

still occur. There the stream for tag  $d$  level 3 is  $[d_1, d_2]$  and the stream for  $c$  level 4 is  $[c_1, c_2]$ . There are only two matches for the query, and data nodes  $c_1$  and  $d_1$  block each other.

*Prefix path streaming* results in no blocking when there is a single branching node in the query. It solves the case in Figure 15b optimally, but not the one in 15c. There the stream for the path  $abac$  is  $[c_1, c_2]$ , and the stream for the path  $ababe$  is  $[e_1, e_2]$ . Even though  $c_2$  is also usable in the match with root  $a_1$ , it cannot be known whether or not  $c_1$  is usable, because  $e_1$  blocks for  $e_2$ .

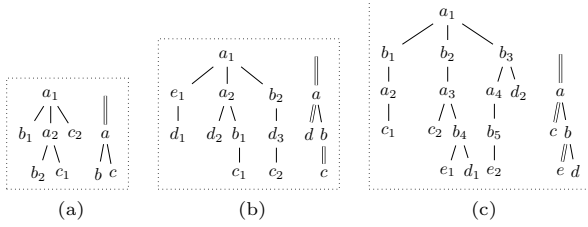


Figure 15: Cases of data and query blocking with (a) tag streaming, (b) tag+level streaming, (c) prefix path streaming. Adapted from [7].

*iTwigJoin* [7] uses a specialized approach for accessing multiple useful streams, which supports any partitioning scheme. In its variant of *getNext(q)*, it considers for each matching stream, the streams which are *usable together* with this stream, for each child of  $q$ . This reduces the amount of blocking when more fine grained partitioning is used. The space usage for *iTwigJoin* is  $\mathcal{O}(d)$ , and the running time is  $\mathcal{O}(t(I + O))$  when no blocking occurs, where  $t$  is the number of useful streams.

**Opportunity 11** (Access methods for multiple matching partitions). Strategies for accessing multiple useful streams include merging, filtering and chaining of input, informed merging access like in *iTwigJoin*, and merging the output of multiple joins. Many authors do not argue for the rationale of their choice of how to access multiple useful partitions for a node. A new access paradigm is presented in [7], but only tag streaming is compared with. The benefit of the method for accessing multiple matching streams is not separated from the benefit of reduced total input size. The technique reduces the number of intermediate paths output by phase one in *TwigStack*, and would undoubtedly reduce the amount of memory needed for intermediate results in time optimal top-down algorithms like *HolisticTwigStack* and *TwigFast*, but it is not certain if it is a win-win both on memory and speed in practice.

### 3.6 Virtual Streams

Another approach that can lead to reading less input is using “virtual streams” for internal nodes, by inferring the existence of nodes from their descendants. This requires a position encoding which allows ancestor position reconstruction [27], such as Dewey [26]. Ancestor label paths must also be infereable, and a path summary is an excellent tool for this. Consider the example in Figure 16, where streams of nodes matching leaf label paths are shown. For the node 1.2.1.2 with path (4)  $a.a.b.a$ , it can be inferred that one candidate for the query root is 1.2 with path (2)  $a.a$ .

*Virtual Cursors* is an implementation of virtual streams using Deweys and path summaries [29]. Generating a “next” match for an internal query node

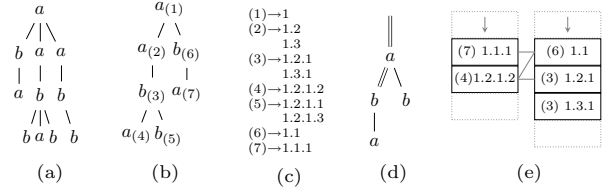


Figure 16: Virtual stream example. (a) Data. (b) Path summary. (c) Extents of summary nodes. (d) Query. (e) Leaf streams.

is done by going through the prefixes of a leaf node’s Dewey and path, and using those where the ending tag is correct. The search stops when the new Dewey is lexicographically greater than the previous, meaning later in the pre-order. [29] does not give details on how ancestor candidates are generated, but this can be done in time linear in the depth of the leaf match used.

Forwarding the entire query is done by repeatedly picking a leaf with a “broken” path, forwarding it to containment by the maximal ancestor, and then forwarding all ancestors virtually to contain the leaf. In the system described by [29], tag streaming with path ID filtering was used, and B-trees were used for skipping during leaf forwarding.

**Other virtual stream approaches** have later been introduced. *TJFast* [22] is an independently developed algorithm which does not use a structural summary, but stores with each data node the root-to-node label path and the Dewey encoding together in a compressed format. Label paths are matched for each node processed. An improvement over *Virtual Cursors* as described, is that path matching is also done when generating internal nodes, giving fewer useless candidates. Also, non-branching internal nodes can be ignored during query evaluation, because they are implicit from the path matchings of below and above nodes. *TJFast* does not produce streams for internal nodes, but maintains *sets* of currently possible candidates.

*TwigVersion* [28] and  $S^3$  [14] (see Section 3.5) are non-holistic approaches which combine structural summaries and inference of internal node matches. *TwigVersion* computes sets of matches bottom-up. Each child node query generates a set of candidates for its parent query node based on its own matches, and these sets are then intersected.  $S^3$  uses the potentially exponential number of ways a query matches the summary, and evaluates each such match, merging the results. For one summary matching, it looks at the query leaf nodes pairwise, and merge joins sets based on lowest common ancestor query nodes. This could give large useless intermediate results. The holistic skipping algorithm *TwigOptimal* [11] does partially implement virtual streams through its “virtual positions” (see Section 3.4).

**Opportunity 12** (Improved virtual streams). To reduce the number of matches and make it possible to ignore non-branching internal nodes, only structurally matching internal nodes should be generated. A structural summary can be used to avoid repeated matching of paths. However, how to store path matching information is not obvious. Given a matching path for a leaf query node, there may be an exponential number of combinations for matching the above query nodes. Should the matches be calculated on the fly as in *TJFast*, kept in independent sets for each node above a leaf match, or encoded in stacks? Or is it enough to store candidates for the lowest branching node above each leaf match, if the

query nodes on a path are processed bottom-up?

It is common to store Deweys in a succinct format to reduce space usage, but in addition, some scheme should also be devised to reduce the redundancy of using related Deweys in ascendant and descendant nodes. It is also preferable if node encodings do not have to be fully de-compressed to be compared during query evaluation, but that the compressed storage format allows for cheap direct comparisons.

**Opportunity 13** (Holistic skipping among leaf streams). In some sense, virtual streams *are* skipping by not generating unrelated matches for internal query nodes. The *Virtual Cursors* algorithm does perform skipping which is “path-holistic”, in the way broken root-to-leaf paths are fixed. The order in which leaves are picked is not specified [29], but query node pre-order could have been used. If the lexicographically largest broken leaf was picked, the skipping would become truly holistic.

The work in [29], in combination with some intermediate result handling method from Section 3.2, may be a suitable starting point in the hunt for the “ultimate” twig matching approach, but the work is not much compared with, or even referenced.

### 3.7 Query Difficulty Classes

As mentioned in Section 2, the “difficulty” of twig joins comes from mixture of a-d edges followed by p-c edges in queries, in combination with structurally recursive labels in the data. [25] shows that when an a-d edge is never followed by a p-c edge downwards in a query, it can be evaluated in linear time and  $\mathcal{O}(d)$  space. When there in addition is a single return node (as in XPath), it can also be evaluated in  $\mathcal{O}(1)$  space. If after combining all the advances listed in this paper, faster evaluation methods still exist for some classes of queries, practical implementations should take advantage of this.

**Opportunity 14** (Identifying and using difficulty classes). Can the correctness of using a simpler matching algorithm be decided not only from the query, but also from the query and the data? Structural summaries give possibilities for this. What happens if there are only single path candidates for some query nodes? What happens when the tree level for matches of a query node is fixed? What happens if data node matches with given paths for some query nodes fix path matches for other query nodes?

In [2] additional information is collected in a path summary, noting whether a node always has a given child, and whether there is at most one child. This information is used there to simplify query evaluation when there are non-return nodes in the query, such as in XPath. Could such statistics also allow detection of more cases where query evaluation can be simplified for general twig matching?

## 4 Conclusion

We have given a structured analysis of recent advances, and identified a number of opportunities for further research, focusing both on join algorithms and index organization strategies. Hopefully this has given an overview which has led us one step further towards unification of the numerous advances in this field.

One conclusion is that given its sheer volume, it seems nearly impossible to consider *all* related work when presenting new twig join techniques. The field would benefit greatly from an open source repository of algorithms and data structures.

## References

- [1] S. Al-Khalifa, H. Jagadish, N. Koudas, J. Patel, and D. Srivastava. Structural joins: A primitive for efficient XML query pattern matching. In *Proc. ICDE*, 2002.
- [2] A. Arion, A. Bonifati, I. Manolescu, and A. Pugliese. Path summaries and path partitioning in modern XML databases. In *Proc. WWW*, 2006.
- [3] R. Bača, M. Krátký, and V. Snášel. On the efficient search of an XML twig query in large DataGuide trees. In *Proc. IDEAS*, 2008.
- [4] R. Bača and M. Krátký. On the efficiency of a prefix path holistic algorithm. In *Proc. XSym*, 2009.
- [5] N. Bruno, N. Koudas, and D. Srivastava. Holistic twig joins: Optimal XML pattern matching. In *Proc. SIGMOD*, 2002.
- [6] S. Chen, H.-G. Li, J. Tatemura, W.-P. Hsiung, D. Agrawal, and K. S. Candan. Twig<sup>2</sup>Stack: bottom-up processing of generalized-tree-pattern queries over XML documents. In *Proc. VLDB*, 2006.
- [7] T. Chen, J. Lu, and T. W. Ling. On boosting holism in XML twig pattern matching using structural indexing techniques. In *Proc. SIGMOD*, 2005.
- [8] S. Chien, Z. Vagena, D. Zhang, V. Tsotras, and C. Zaniolo. Efficient structural joins on indexed XML documents. In *Proc. VLDB*, 2002.
- [9] B. Choi. What are real DTDs like. Technical Report MS-CIS-02-05, University of Pennsylvania, 2002.
- [10] B. Choi, M. Mahoui, and D. Wood. On the optimality of holistic algorithms for twig queries. In *Proc. DEXA*, 2003.
- [11] M. Fontoura, V. Josifovski, E. Shekita, and B. Yang. Optimizing cursor movement in holistic twig joins. In *Proc. CIKM*, 2005.
- [12] G. Gottlob, C. Koch, and R. Pichler. Efficient algorithms for processing XPath queries. In *Proc. VLDB*, 2002.
- [13] G. Gou and R. Chirkova. Efficiently querying large XML data repositories: A survey. *Knowl. and Data Eng.*, 2007.
- [14] S. K. Izadi, T. Härder, and M. S. Haghjoo. S<sup>3</sup>: Evaluation of tree-pattern XML queries supported by structural summaries. *Data Knowl. Eng.*, 2009.
- [15] H. Jiang, H. Lu, W. Wang, and B. C. Ooi. XR-tree: Indexing XML data for efficient structural joins. In *Proc. ICDE*, 2003.
- [16] H. Jiang, W. Wang, H. Lu, and J. Yu. Holistic twig joins on indexed XML documents. In *Proc. VLDB*, 2003.
- [17] Z. Jiang, C. Luo, W.-C. Hou, and Q. Z. D. Che. Efficient processing of XML twig pattern: A novel one-phase holistic solution. In *Proc. DEXA*, 2007.
- [18] R. Kaushik, P. Bohannon, J. F. Naughton, and H. F. Korth. Covering indexes for branching path queries. In *Proc. SIGMOD*, 2002.
- [19] R. Kaushik, R. Krishnamurthy, J. F. Naughton, and R. Ramakrishnan. On the integration of structure indexes and inverted lists. In *Proc. SIGMOD*, 2004.
- [20] G. Li, J. Feng, Y. Zhang, and L. Zhou. Efficient holistic twig joins in leaf-to-root combining with root-to-leaf way. In *Proc. Advances in Databases: Concepts, Systems and Applications*, 2007.
- [21] J. Li and J. Wang. Fast matching of twig patterns. In *Proc. DEXA*, 2008.
- [22] J. Lu, T. Ling, C. Chan, and T. Chen. From region encoding to extended Dewey: On efficient processing of XML twig pattern matching. In *Proc. VLDB*, 2005.
- [23] L. Qin. Personal correspondence, 2009.
- [24] L. Qin, J. X. Yu, and B. Ding. TwigList: Make twig pattern matching fast. In *Proc. DASFAA*, 2007.
- [25] M. Shalem and Z. Bar-Yossef. The space complexity of processing XML twig queries over indexed documents. In *Proc. ICDE*, 2008.
- [26] I. Tatarinov, S. D. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang. Storing and querying ordered XML using a relational database system. In *Proc. SIGMOD*, 2002.
- [27] F. Weigel. *Structural summaries as a core technology for efficient XML retrieval*. PhD thesis, Ludwig-Maximilians-Universität München, 2006.
- [28] X. Wu and G. Liu. XML twig pattern matching using version tree. *Data & Knowl. Eng.*, 2008.
- [29] B. Yang, M. Fontoura, E. Shekita, S. Rajagopalan, and K. Beyer. Virtual cursors for XML joins. In *Proc. CIKM*, 2004.
- [30] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman. On supporting containment queries in relational database management systems. *SIGMOD Rec.*, 2001.
- [31] J. Zhou, M. Xie, and X. Meng. TwigStack<sup>+</sup>: Holistic twig join pruning using extended solution extension. *Wuhan University Journal of Natural Sciences*, 2007.

# A Quantum Interpretation of the View-Update Problem

Christian Flender

Faculty of Science and Technology,  
Queensland University of Technology,  
Level 8 / 126 Margaret Street,  
Brisbane QLD 4000, Australia,  
Email: c.flender@qut.edu.au

## Abstract

The ANSI-SPARC architecture was proposed as a hierarchical model for the implementation of Database Management Systems (DBMS). A separation of external user views and shared base relations (conceptual schema) constitutes their logical independence, i.e., external views are immune to changes of the conceptual schema. Moreover, users can customize their views independent of the conceptual schema. However, all updates to a base relation should be immediately reflected in all views that reference the base relation. Vice versa, if a view is updated, then the underlying base relation should reflect the change. Keeping views and base relations in sync came to be known as the view-update problem. This paper argues that view updates require the user to cause a change. Prior to a view update user and view are entangled. Entangled states cannot be reduced to factual states of user and base relation. It will be shown that the view-update problem arises due to a view update (causation) being irreducible to a functional mapping between base relation and view (causality).

**Keywords:** *ANSI-SPARC architecture, view-update problem, quantum entanglement, causation, causality*

## 1 Introduction

As early as 1971, the Data Base Task Group (DBTG) appointed by the Conference of Data Systems and Languages (CODASYL) proposed a general architecture for database systems (CODASYL 1971). This proposal already recognized the need for several levels of abstraction in order to deal with the diverging needs of database stakeholders such as end-users, application developers and administrators (DBAs). Already hierarchical or first-generation DBMS divided their system into schema and subschema. The former was meant to provide a view for DBAs who were mainly concerned with the definition of database names, types of records and components of these records. The schema was separated from the so called subschemas, the part of the database as seen by users or application programs. Then in 1975, with the rise of relational or second-generation DBMS, which can be traced back to E.F.Codd's influential paper in 1970 (Codd 1970), the American National Standards Institute's (ANSI) Standards Planning and Requirement Committee (SPARC) produced a similar multi-level

architecture with the same purpose, i.e., the provision of an implementation independent layer to isolate programs from underlying representational issues (ANSI-SPARC 1975). The three-level ANSI-SPARC architecture is a *de facto* standard for the implementation of relational DBMS and its relevance to information systems continues to the present day.

The three-level architecture consists of an external level, a conceptual level, and an internal level (cf. Figure 1). The way users perceive data, their personalized view (top-down), belongs to the external level. The internal level abstracts from the way the operating system accesses data (bottom-up), i.e., via data structures and file organisations. The conceptual level provides a mapping between external and internal level and thereby mediates between customized user views and physical data implementation. Generally, at the conceptual level the concep-

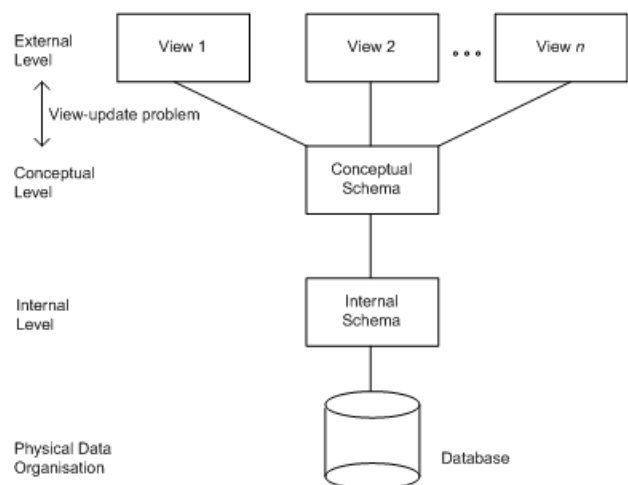


Figure 1: The ANSI-SPARC architecture divides a DBMS into three levels. At the top the external level consists of customized views and therefore interfaces to end-users and application developers. At the bottom the internal level abstracts from the physical data implementation. The conceptual level provides a mapping between external level and internal level and thereby mediates between views and physical data organisation.



Cartesian product, union and set difference, acting upon base relations. A separation of external user views and shared base relations constitutes their logical independence, i.e., external views are immune to changes of the conceptual schema. Users can customize their views at the external level independent of the conceptual schema. However, all updates to a base relation should be immediately reflected in all views that reference the base relation. Vice versa, if a view is updated, then the underlying base relation should reflect the change. Keeping views and base relations in sync came to be known as the view-update problem.

There are several approaches attempting to solve this problem of external-conceptual mapping (see related work in Section 4). However, none of these approaches has ever considered a view update at the external level as a quantum-like cause (conditions for change are necessary but not sufficient) being irreducible to a determinate functional mapping (conditions for change are necessary and sufficient) at the conceptual level. This article aims to offer a quantum interpretation of the view-update problem. It is argued that, prior to a view update, user and view are entangled (user and view cannot be separated). The change of a view itself is indeterminate, i.e., the state of a view is determined by the outcome of an update but not prior to it. Accordingly, changes being propagated to the conceptual level do not necessarily translate into determinate functional mappings, i.e., the application of one or more relational operators (selection, projection, Cartesian product, union and set difference). Entangled states of user and view at the external level are irreducible to states of user and base relation at the conceptual level. Hence, quantum entanglement offers a reasonable explanation of the view-update problem.

The contribution of this paper is to show that the interface between user and DBMS is not as clear cut as the logical and physical independence of the ANSI-SPARC architecture might suggest. Although the ANSI-SPARC architecture claims to separate logic from physics, it cannot separate the user from the DBMS. User-view entanglement inseparably couples user and DBMS. A view derives from a base relation; a view update, however, often cannot be reduced to changes at the conceptual level. What came to be known as the view-update problem has a natural and reasonable explanation. Views are not independent from their implementation but rely on the user's causation for their realization.

The next section introduces the view-update problem and shows where change propagation from external to conceptual level breaks down. Then, in Section 3, user-view relationships will be shown to be entangled and therefore quantum-like. Section 4 discusses existing work related to the view-update problem, in particular related to indeterminate updates (causation) at the external level and determinate updates (causality) at the conceptual level. Finally, Section 5 concludes the article and gives an outlook towards future work.

## 2 The View-Update Problem

The view-update problem arises at the interface between views at the external level and base relations at the conceptual level (cf. Figure 1). The latter constitutes a shared representation accessible by multiple users and independent from storage space allocation and index structures. At the conceptual level, the DBMS manages base relations. A base relation is a named relation, e.g., *Staff*, corresponding to an entity in the real world. Concrete instances of a base rela-

tion, e.g., *Staff* member 001, are tuples and stored physically in the database. A view, on the other hand, is defined at the external level. It is the dynamic result of one or more relational operations<sup>1</sup>, i.e., selection, projection, Cartesian product, union and set difference, acting upon a base relation to produce another relation, e.g., *Manager* as a subset of *Staff*. Hence, a view is a virtual or derived relation in the sense that it belongs to the external level and can be produced upon request by a particular user.

Views offer flexibility and security by hiding certain parts of the database from certain users, e.g., the view *Manager* doesn't contain information about board members. Furthermore, users can customize their information needs. The same data can be seen in different ways at the same time. Moreover, views can simplify complex operations upon base relations. Since changes of a base relation, e.g., adding attributes or tuples, do not necessarily require changes of views, they are considered to be logically independent. Nevertheless, all updates of a base relation should be immediately reflected in all views that reference the base relation. Vice versa, if a view is updated, then the underlying base relation should propagate the change. Here, the view-update problem arises due to the fact that there are view updates which do not translate into base relation updates so that, eventually, the changed view equals the view of a changed base relation. An update mapping does not always exist, and even when it does exist, it may not be unique (Codd 1975). Therefore, a change in the view may not be reflected unambiguously by equivalent changes in the base relation. In formal terms,

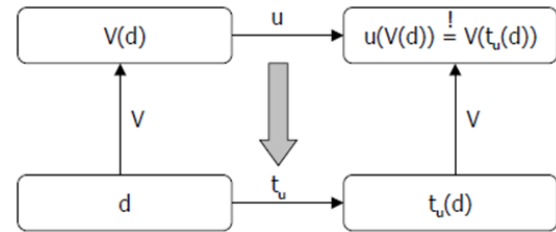


Figure 2: The view-update problem refers to the synchronization of external level and conceptual level. A view is derived from a base relation by means of operators as defined in relational algebra. If a user causes a view update this has to be propagated to the conceptual level such that the changed view equals the view projection of the changed base relation.

consider the state  $d$  of a base relation (cf. Figure 2). This state is projected onto the external level by a mapping  $V$ .  $V$  is functional, i.e., conditions for the mapping are causally necessary and sufficient. Effectively,  $V$  refers to one or more relational operations, i.e., selection, projection, Cartesian product, union and set difference, acting upon the base relation  $d$  to produce a view  $V(d)$ . If a user carries out a view update  $u$ , then this results in the changed view  $u(V(d))$ . Accordingly, at the conceptual level, there must be a base relation update  $t_u$  such that it reflects the change at the external level. Hence, the challenge is to find a transformation

$$u \rightarrow t_u, \quad (1)$$

<sup>1</sup>For a proof showing that operators of relational algebra are essentially equivalent in expressive power to relational calculus and thus first-order logic see (Codd 1970).



where  $u(V(d)) = V(t_u(d))$ . However, according to the view-update problem,  $u \rightarrow t_u$  is indeterminate for arbitrary mappings  $V$ . Put in another way, there are view-updates  $u$  and mappings  $V$  for which external level and conceptual level cannot be synchronized, and, given  $u$  and  $V$ , there is no general rule for determining synchronizability.

The following example illustrates the view-update problem. Consider a simple base relation *Staff*. The current state of the base relation is  $d = \text{Staff}\{(001,B1),(002,B2)\}$ . The state  $d$  of the base relation is composed of two tuples; the first value of each tuple stands for the identification number of a staff member. The second entry represents the department number. A selection  $V = \text{'get Staff where depNo=B1'}$  defines the mapping that generates a view. The view is meant to be a subset of all staff members, in particular managers working in department B1. Consequently,  $V(d) = \text{Manager}\{(001,B1)\}$ .

Now assume a user updates this view. For instance, if he carries out  $u = \text{'update Manager set depNo=B3 where ID=001'}$ , then this will change the view such that  $u(V(d)) = \text{Manager}\{(001,B3)\}$ .

Accordingly, changes need to be done at the conceptual level. A transformation  $u \rightarrow t_u$  is needed which satisfies the constraint  $u(V(d)) = V(t_u(d)) = \text{Manager}\{(001,B3)\}$ . However, there is no transformation  $u \rightarrow t_u$  where  $u(V(d)) = V(t_u(d))$  if the view update  $u$  'interferes' with the view mapping  $V$ . In other words, the projection  $V$  defines a view  $V(d)$  by operating upon a base relation  $d$ . If an update  $u$  upon  $V(d)$  alters the view mapping  $V$ , then view and base relation cannot be synchronized. For example, consider the transformation for which the view update  $u$  is applied at the conceptual level such that  $t_u = u = \text{'update Staff set depNo=B3 where ID=001'}$ . It follows that the base relation changes to  $t_u(d) = \text{Staff}\{(001,B3),(002,B2)\}$ . However,  $V(t_u(d)) = \text{Manager}\{\emptyset\}$  and therefore  $u(V(d)) \neq V(t_u(d))$ . If view update  $u$  and view mapping  $V$  interfere, then there is no transformation  $u \rightarrow t_u$  such that  $u(V(d)) = V(t_u(d))$ . Interference is nothing beyond reasoning. In fact, it is well known in quantum theory and thus the view-update problem suggests a quantum interpretation.

### 3 User-View Entanglement

Recently quantum formalisms have been adopted to problem descriptions outside of physics (Bruza, Lawless, van Rijsbergen, Sofge, Coecke & Clark 2008, Bruza et al. 2009). Examples related to databases are models in information retrieval (IR) (Piwowarski & Lalmas 2009b,a) or the design of part-whole relationships using the Entity-Relationship (ER) notation (Flender et al. 2009). One of the selling points of quantum models is their ability to represent effects like context-dependence and emergence (Kitto 2008). Phenomena like human-computer interactions are observer-relative and therefore context-dependent. An observation, action, or change, can be modelled as context. For instance, the meaning of a word like *Bat* depends on the context in which it is used. It might be understood as an animal or a sports utility dependent on its evocation (Bruza, Kitto, Nelson & McEvoy 2008). Similarly, emergent phenomena are often conceptualized as entangled or nonseparable states. For instance, an instance of a combined concept like *Pet Fish* can be modelled as an entangled or nonseparable state that emerges with dependence upon context (Aerts & Gabora 2005b,a). Experiments have shown that users generally associate instances like 'guppy' neither as a good example of the word *Pet* nor of the word *Fish*. However, their rating

in the context of *Pet Fish* reveals a strong association. Therefore, 'guppy' is modelled as an emergent, entangled or nonseparable state of *Pet Fish*. Modelling emergent and context-dependent states requires operators which do not exist in relational algebra. The two operators relevant to the view-update problem are the measurement function and the Tensor product. The latter shall be introduced later in this section.

The quantum formalism defines an actualization function which corresponds to a measurement in quantum physics. This function is indeterminate in the sense that the resulting state of an observed system is determined by the outcome of the measurement but not prior to it. An observer causes the system to change (causation) but he will never be able to give necessary and sufficient conditions for a change to occur (causality). The probability involved is non-classical since there is no underlying deterministic assumption. Consequently, prior to a change or update, there are no necessary and sufficient conditions, i.e., the state in which a system will be found by chance was not determined at the time the change was made. In contrast, classical probabilistic functions assume a system to be in one or another state, i.e., the state of a system, though not yet determined, is in a definite state. Accordingly, relational operations as physically instantiated by a DBMS always presuppose views and base relations to be in a definite state. To make this more precise, we need to have a look at the quantum formalism, in particular vector spaces and operators.

A vector space is a multi-dimensional space. A Hilbert space is an abstract or infinite-dimensional vector space over the set of real or complex numbers. It is equipped with inner products, i.e., rules to measure distances and angles between vectors. Vectors have a magnitude denoting their relative size or length in space. Consider the orthonormal basis  $B$  of a two-dimensional Hilbert space (cf. Figure 3). For an or-

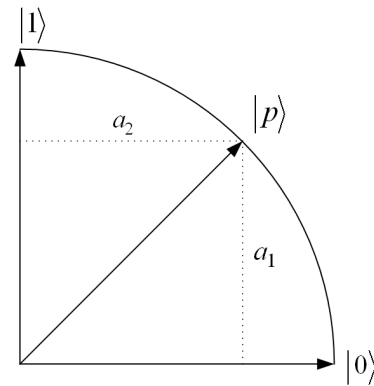


Figure 3: The figure shows a two-dimensional Hilbert space. It is equipped with inner products, i.e., rules to measure distances and angles between vectors. The orthonormal basis represents a view  $B$  which is composed of two values or qubits  $|1\rangle$  and  $|0\rangle$ . After an update  $u$ , the view is in a state  $|0\rangle$  where  $u(V(d)) \neq V(t_u(d))$  or in a state  $|1\rangle$  where  $u(V(d)) = V(t_u(d))$ .

thonormal basis  $B$ , basis vectors are both normalized and orthogonal to each other. Two vectors<sup>2</sup>  $|0\rangle$  and  $|1\rangle$  are orthogonal if their angle is  $90^\circ$  or their cosine equals 0. The dot product or inner product  $\langle 0|1\rangle$  returns a scalar. If vectors are normalized this scalar equates the cosine and hence measures the angle or

<sup>2</sup>Note that besides (Brak-)ket notation ( $| \rangle$  and  $\langle |$ ) vectors can be written in terms of linear algebra or coordinate vectors.

distance between them. Basis vectors are normalized if their length equals 1 and so they are called unit vectors. A vector of arbitrary length can be divided by its length to create a unit vector. The length or magnitude of a vector is, according to Pythagoras, the square root of the sum of all squared vector components.

Each vector  $|p\rangle$  can be written as a linear combination of orthogonal vectors.

$$|p\rangle = a_1|0\rangle + a_2|1\rangle, \quad (2)$$

where  $|a_1|^2 + |a_2|^2 = 1$ . The vector  $|p\rangle$  represents a superposition or potentiality state. To illustrate the relationship between superposition states and actual states (eigenstates), consider the classical example of wave-particle complementarity.

Generally, matter and energy exhibit both wave-like and particle-like properties but not both at the same time, i.e., not within the same context. In different contexts or experimental arrangements some matter seems more particle-like than wave-like. With reduced values of energy (change of context) the same matter will be more likely to show wave-like qualities than particle-like properties. All the information about a particle is encoded in its wave function, which is analogous to the amplitude of a wave at each point in space. This function evolves according to a differential equation (the Schrödinger equation) and so gives rise to interference. Interference occurs when the interaction of two or more waves, e.g., one wave representing observer and the other one standing for the observed system, influences their direction of propagation characterized by crests and troughs. When two or more waves reach the same point in space at the same time, they either add up (the crests arrive together which is called in-phase) or cancel each other out (the crest from one wave meets a trough from another wave which is called out-of-phase). The state of a wave-like property is called superposition or potentiality state and represented as a vector  $|p\rangle$ . Its linear combination, the superposition or addition of two or more states, resembles an interference pattern typical of waves. If an observer measures the location

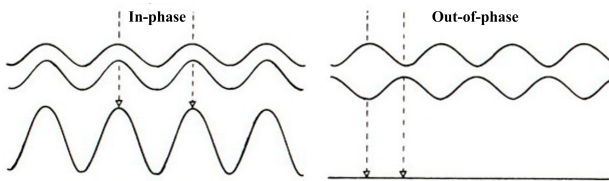


Figure 4: Interference occurs when the interaction of two or more waves, e.g., one wave representing observer and the other one standing for the observed system, influences their direction of propagation characterized by crests and troughs. When two or more waves reach the same point in space at the same time, they either add up (the crests arrive together which is called in-phase) or cancel each other out (the crest from one wave meets a trough from another wave which is called out-of-phase).

of the particle encoded in  $|p\rangle$ , the wave-function will randomly collapse to a well-defined position, a state like  $|1\rangle$  or  $|0\rangle$  traditionally associated with particles. The act of measurement, or context, of a particular situation is modelled as a choice of basis where the eigenstate of the system is chosen from the set of possibilities in such a way that it is compatible with the choice of action, i.e., context, to be performed upon

it. Expectation values  $|a_1|^2$  and  $|a_2|^2$  are related to the probability  $P(|0\rangle)$  and  $P(|1\rangle)$  of the system being eventually found in their respective eigenstates  $|0\rangle$  and  $|1\rangle$ . Put in another way, the likelihood of any particular location, or eigenstate, equals the squared amplitude  $|a_1|^2$  and  $|a_2|^2$  of the wave-function in this location.

$$P(|0\rangle) = |a_1|^2 \text{ and } P(|1\rangle) = |a_2|^2 \quad (3)$$

How does this actualization function translate into the ANSI-SPARC architecture? As discussed in the previous section, external views constitute the interface to the user. Whether end-user or application developer, at one stage he or she will update a view. Although database operations are physically instantiated causal functions, the actual change requires causation. Causation means the user makes an effort to change a view for which he may give reasons, e.g., a customer wants a modified view on his data, the program code must be changed, etc. However, such reasons will never be causally necessary and sufficient. Causation is not causality. Therefore, prior to an update  $u$ , a view  $V(d)$  is always superposed in a state  $|p\rangle$ . After an update  $u$  the view is in a state  $|0\rangle$  where  $u(V(d)) \neq V(t_u(d))$  or in a state  $|1\rangle$  where  $u(V(d)) = V(t_u(d))$ . Now, according to the view-update problem,  $u \rightarrow t_u$  is indeterminate for arbitrary mappings  $V$ . Put in another way, there are view-updates  $u$  and mappings  $V$  for which it cannot be determined whether a)  $|p\rangle$  collapses to  $|1\rangle$ , i.e., there is a transformation  $u \rightarrow t_u$  such that  $u(V(d)) = V(t_u(d))$ , or b)  $|p\rangle$  collapses to  $|0\rangle$ , i.e., there are only transformations  $u \rightarrow t_u$  such that  $u(V(d)) \neq V(t_u(d))$ . Therefore, we need a superposition state which cannot be reduced to  $|0\rangle$  or  $|1\rangle$ . To explain this irreducible indeterminism inherent in the view-update problem, the user has to be modelled. A situation in which a user cannot give causally necessary and sufficient conditions for a view  $V(d)$  or state  $|p\rangle$  to be reduced to a state  $|1\rangle$  where  $u(V(d)) = V(t_u(d))$  or a state  $|0\rangle$  where  $u(V(d)) \neq V(t_u(d))$  refers to the entanglement of user and view.

Entanglement requires the combined system user *plus* view to be in a superposition state that is neither reducible to the user nor to the view. Prior to a view update user and view are entangled. Moreover, views are virtual relations derived from the conceptual schema, i.e., the community view. Therefore, views, though not reducible to base relations, depend on the conceptual schema nevertheless. Entangled states of user and view refer to one integrated whole. More technically, an entangled state of user *plus* view cannot be written as a Tensor product of the user's current state and the view's current state. The following paragraph formalizes this situation.

The Tensor product is an outer product of matrices for which states can be easily shown to be entangled. In contrast to the Cartesian product of relational algebra, the Tensor product operates within combined Hilbert spaces. Tensor products are outer products of vectors and generate states within combined systems. The dimensionality of a composite state space is  $I^J$  where  $I$  is the number of single-body systems, e.g., 2 for user and view, and  $J$  is the sum of vector components, e.g., 2 for success and failure. For instance, the tensor product of two vectors associated with two bases  $B_1$  (user) and  $B_2$  (view) operates within a four-dimensional vector space  $B = \{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ . Each vector  $\psi$  in this space is regarded as corresponding to a possible state of the associated pair of two-state systems. If such a state can be factorized into states of the single bases involved it is separable, i.e., it is a product state. However, it is easy to find vectors which

cannot be expressed as a tensor product of a pair of state vectors from the associated quantum systems. Such vectors are entangled states. In terms of the

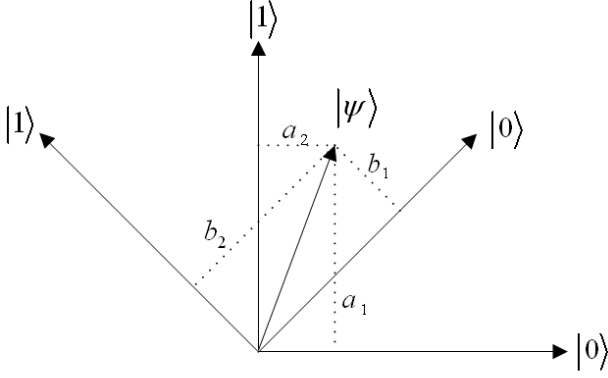


Figure 5: The figure shows a combined system composed of two qubit bases  $B_1 = \{|0\rangle, |1\rangle\}$  (user) and  $B_2 = \{|0\rangle, |1\rangle\}$  (view). Vectors  $\psi$  generated within that space represent possible states of the system.  $\psi$  is a superposition which is either a separable or an entangled state.

view-update problem, a product or separable state  $|\psi\rangle$  refers to the combined state of user  $|p_1\rangle$  and view  $|p_2\rangle$  (or  $V(d)$ ) for which the outcome of  $u$  has determined whether a) there is a transformation  $u \rightarrow t_u$  such that  $u(V(d) = V(t_u(d))$ , or b) there are only transformations  $u \rightarrow t_u$  such that  $u(V(d) \neq V(t_u(d))$ . If user  $B_1$  and view  $B_2$  disentangle, then  $\psi$  is separated. *A posteriori* the state of  $u(V(d))$  is either  $|0\rangle$  or  $|1\rangle$  determined by the outcome of  $u$ .

$$\begin{aligned} |\psi\rangle &= |p_1\rangle \otimes |p_2\rangle \\ &= (a_1|0\rangle + a_2|1\rangle) \otimes (b_1|0\rangle + b_2|1\rangle) \\ &= a_1b_1|00\rangle + a_2b_1|10\rangle + a_1b_2|01\rangle + a_2b_2|11\rangle, \end{aligned} \quad (4)$$

where  $|a_1b_1|^2 + |a_2b_1|^2 + |a_1b_2|^2 + |a_2b_2|^2 = 1$ .

Prior to an update  $u$ , however, user and view are entangled. If  $|\psi\rangle$  is entangled, there is no  $|p_1\rangle$  and  $|p_2\rangle$  such that  $|\psi\rangle = |p_1\rangle \otimes |p_2\rangle$ . In contrast to product states, entangled quantum states are not separable. If two or more quantum states entangle, they interact so that a nonseparable state emerges. The joint probability that emerges cannot be factorized into single probabilities. In physics, the empirical setting for this phenomenon involves two spatially separated (non-local) measuring devices performing spatially separated measurements. Eventually, one of the measurements disentangles  $B_1$  (user) and  $B_2$  (view) and the outcome determines the eigenstates in which the system will be found.

$$|\psi\rangle = x|00\rangle + y|11\rangle, \quad (5)$$

where  $|x|^2 + |y|^2 = 1$ .

In terms of the view-update problem, user  $B_1$  and view  $B_2$  interact in a way that it cannot be determined whether an update  $u$  applied to the view  $V(d)$  translates into an update  $t_u$  applied to the base relation  $d$  such that  $u(V(d)) = V(t_u(d))$ . The entangled state  $|\psi\rangle$  of the user-view whole cannot be reduced to a failed state  $|0\rangle \in B_2$  (there is no transformation) or a successful state  $|1\rangle \in B_2$  (there is a transformation).

Reconsider Equation 4 and Equation 5. Since  $x > 0$  it follows that  $a_1 > 0$  and  $b_1 > 0$ . Moreover,

there is no coefficient for  $|10\rangle$  and  $|01\rangle$  and therefore  $a_2 = 0$  or  $b_2 = 0$ . However,  $y|11\rangle$  implies that  $a_2 > 0$  and therefore it contradicts the fact that  $a_2 = 0$ . Consequently, there are no two states each belonging to one of the bases  $B_1$  (user) and  $B_2$  (view) and thus  $\psi$  is a nonseparable or entangled state.

#### 4 Related Work

So far it has been shown that a quantum interpretation of the view-update problem gives a reasonable answer to the following questions. Why is there no rule for determining synchronizability? In other words, why is there no rule for determining whether view updates translate into base relation updates for arbitrary view mappings or not? A quantum interpretation of the view-update problem explains this indeterminism. User and view are entangled and therefore, prior to a view update, it is not determined whether this update translates into changes of a base relation or not. Due to the irreducibility of user and view there may be ambiguous or interfering changes of the base relation which cannot be determined prior to the user's update. Put in another way, the user's causation is irreducible to relational operators and thus to causal functions as physically instantiated by the DBMS. Moreover, there is no independence between user and DBMS. The physical realization of a view update depends on the user's causation. The remainder of this section shall discuss causation and causality in relation to existing work on the view-update problem.

First attempts to solve the view-update problem can be traced back to the early 1980s (Bancilhon & Spyratos 1981, Dayal & Bernstein 1982). Several classes have been defined for which views are theoretically updatable, theoretically not updatable, and partially updatable. However, given an update  $u$ , for arbitrary  $V$ , there is no general rule that determines synchronizability (see (Furtado & Casanova 1985) for a survey on updating views). An update mapping  $u \rightarrow t_u$  does not always exist, and even when it does exist, it may not be unique (Codd 1975). A change in the view may not be reflected unambiguously by equivalent changes in the base relation. Nevertheless, many attempts to synchronize external and conceptual level have been made; attempts to compute  $u \rightarrow t_u$  such that  $u(V(d)) = V(t_u(d))$ . To deductively derive unambiguous transformations  $u \rightarrow t_u$ , view updates  $u$  are usually constrained to certain views  $V(d)$ , in particular those views  $V(d)$  that can be produced by the view projection  $V$ . Views are derived or virtual relations and therefore restricted to their creation using relational operators of relational algebra. Stringent conditions have to be introduced in order for a view update to translate into a base relation update such that the changed view equals the view of a changed base relation. Otherwise, undesirable side effects may occur effectively rendering external level and conceptual level inconsistent. Constraints can be realized by limiting updates to views satisfying several conditions. For instance, Dayal and Bernstein (1982) propose conditions under which a view update translates into a base relation update (Dayal & Bernstein 1982). Conditions were derived in terms of base relation instances and view instances using functional dependencies, keys, and subset constraints. The authors define simple syntactic translation procedures and derive checkable conditions that characterize when the translations produced by these procedures will satisfy the various correctness criteria. Accordingly, most commercial relational DBMS implementing the Standard Query Language (SQL) constrain updates to particular views. For instance,

views should be defined using simple queries involving a single base relation and the primary key or a candidate key of that base relation. Hence, updates are not allowed through views involving the Cartesian product over multiple base relations. Moreover, views derived from complex operations like aggregate functions and groupings are prohibited.

Other approaches have attempted to compute a transformation  $u \rightarrow t_u$  using view complements (Bancilhon & Spyrtos 1981). View complements represent missing data sources being complementary to the data of a view. Such complements are used in conjunction with a view update  $u$  in a way sufficient to deductively derive changes  $t_u$  of the base relation  $d$ . Hence, complementary information is exploited to translate view updates to base relation updates. The base relation can be reconstructed from the view and its complement. However, complements are not unique and the choice of an optimal complement requires deductive rules which haven't been found. In fact, it has been argued that there is no way to compute optimal complements within relational algebra if views derive from projection operators (Lechtenböcker & Vossen 2003).

In summary, deductive attempts to solve the view-update problem enforce constraints upon possible view changes. In this way, a DBMS defines requirements, metaphorically speaking a straitjacket, for updates  $u$ . In terms of the ANSI-SPARC architecture, requirements are imposed bottom-up. Physically instantiated functions dictate the range of possible views. Hence, restricting the user in his possibilities is not really a solution to, or explanation of, the view-update problem. Rather it is an educational program that enforces the user to speak the language of relational algebra, and thus first-order logic. The fact that, for practical purposes, such languages get a face-lift, e.g., SQL and other declarative or procedural languages, doesn't avoid the enforcement of constraints upon possible view changes.

Another interesting attempt to synchronize views and base relations uses abductive reasoning (Kakas & Mancarella 1990). Abductive reasoning is a kind of backward reasoning, i.e., the inverse of modus ponens. According to modus ponens:  $A \rightarrow B, A \vdash B$ . For instance, if someone is a human ( $A$ ), then he is mortal ( $B$ ). Christian is a human, therefore he is mortal. In contrast to this forward chaining which starts with the antecedent condition  $A$ , abductive reasoning starts with an observation  $B$ . It looks for possible explanations of this observation. According to a given theory, there are several explanations of someone being mortal, e.g., he is a man or she is a woman. In terms of the view-update problem, a changed view  $u(V(d))$  represents the conclusion or observation  $B$  which needs affirmation. The projection  $V$  is meant to be an assumed theory and the possible states  $d$  of a base relation represent abducibles required to explain the changed view. Abductive reasoning now generates alternative explanations, i.e., possible changes  $t_u$  of the base relation, in order to match the changed view  $u(V(d))$ . This method can be useful as a heuristic to find a good explanation. However, since there are multiple views  $V(t_u(d))$  as possible explanations, an exact solution would either require deductive reasoning in order to draw or derive the conclusion or an exhaustive inductive affirmation. Moreover, abductive reasoning is subject to the fallacy that an observation  $B$  is solely based on the order of events for which  $A$  is causally necessary and sufficient (*Post hoc ergo propter hoc*).

In summary, causation can neither be reduced to abductive reasoning nor deductive rules. In fact, user-view entanglement inseparably links user and DBMS and, unlike physically instantiated causal functions,

there are no necessary and sufficient conditions for a view update to occur. According to the quantum interpretation presented in this article, failed attempts to deductively or abductively derive base relation updates from view updates have a reasonable explanation. Practically, there is not *one* solution to the view-update problem as traditionally conceived. It rather dissolves if one accepts that external views are literally perspectives inseparably bound to their users.

## 5 Conclusion and Outlook

A quantum explanation of the indeterminate transformation  $u \rightarrow t_u$  draws from the user's causation. The user cannot give necessary and sufficient conditions for a view update to occur. Accordingly, causation cannot be reduced to relational operators physically implemented as causal functions. User-view states are represented as entangled states  $\psi$  which require causation to become actual and thus determinate. Such nonseparable states  $\psi$  cannot be reduced to an updated view state  $|1\rangle \in B_2$  for which there is a transformation  $u \rightarrow t_u$  such that  $u(V(d)) = V(t_u(d))$  or a state  $|0\rangle \in B_2$  for which there are only transformations  $u \rightarrow t_u$  such that  $u(V(d)) \neq V(t_u(d))$ . Therefore, there is a natural and reasonable explanation for the indeterminacy of  $u \rightarrow t_u$  and thus for the view-update problem.

What conclusions can be drawn with regard to the ANSI-SPARC architecture and information systems in general? Generally, it must be acknowledged that persistent storage of data is the backbone of most application programs. Accordingly, user interfaces comply with the general idea of abstracting from the underlying physical data organisation as the ANSI-SPARC architecture illustrates for relational databases. However, and this is the main upshot of this article, one can argue that external views gain a new quality due to their inseparability from the user. The user-view whole, an indeterminate, irreducible and integrated set of states, poses exciting questions about the nature of such states. Causation, i.e., the user's effort to actively change his view and thereby the view as provided by the information system, transcends a clear distinction between user and view. The source of data for modelling user and view as an integrated and undifferentiated realm becomes less an analytical task of separation or decomposition. Rather it turns into a challenge of describing the synthesizing ways, or modes, such views change as a function of causation. There is much to learn from the cognitive sciences, an interdisciplinary field comprising subjects like psychology, artificial intelligence and philosophy. Here, recent theories conceptualize the user as an embodied agent whose perspectives change as a function of movements or motor habits. Taking the user as an embodied agent, who is integrated with external views of different representational formats, e.g., graphical, formal or textual, constitutes a rich and promising data source. The user's experiential episodes, carefully described from the background of his expertise in doing so, could possibly lead to a better understanding of the user's practical engagement with information technology. A paradigmatic example of practical reasoning is absorbed skilful coping. Here, users generally do not adopt a scientific attitude towards a subject matter, e.g., they do not analyse their engaged activity like they would construct a logical argument. People dealing with all sorts of artefacts usually immerse themselves in such a way that they gain a maximal grip on the contingencies in their environment. Simply presupposing a clear separation between user and computational de-

vice distorts the phenomenon of what it is actually like to deal with such a device. Although user-view entanglement is an undifferentiated realm for analysis this is not to say that practical reasoning is indifferent to synthesis. Several aspects of human-computer interactions, from simple viewings of pictures to deliberate reflections upon screen content, are hidden, or opaque, and therefore require special treatment. For instance, revealing aspects like the apprehension of a pictorial subject through perception of a pictorial image requires users to attend to the activity of picture viewing. Correlative to the actual content of a picture, picture viewing is a synthetic unity of perceptual and imaginative aspects. During the last century, several methods for describing the synthetic unity of such aspects have been worked out. They stem from the tradition of continental philosophers like Edmund Husserl, Martin Heidegger and Maurice Merleau-Ponty.

## References

- Aerts, D. & Gabora, L. (2005a), 'A State-Context-Property model of concepts and their combinations I: The structure of the sets of contexts and properties', *Kybernetes* **34**(1&2), 167–191.
- Aerts, D. & Gabora, L. (2005b), 'A State-Context-Property model of concepts and their combinations II: A Hilbert space representation', *Kybernetes* **34**(1&2), 192–221.
- ANSI-SPARC (1975), Data base management systems: Interim report., Technical report, FDT, ACM SIGMOD bulletin. Volume 7, No. 2.
- Bancilhon, F. & Spyrtos, N. (1981), 'Update semantics of relational views', *ACM Transactions on Database Systems* **6**(4), 557–575.
- Bruza, P., Kitto, K., Nelson, D. & McEvoy, K. (2008), Entangling words and meaning, in 'Proceedings of the Second Quantum Interaction Symposium', University of Oxford.
- Bruza, P., Lawless, W., van Rijsbergen, C., Sofge, D., Coecke, B. & Clark, S., eds (2008), *Proceedings of the Second Quantum Interaction Symposium*, College Publications, University of Oxford.
- Bruza, P., Sofge, D., Lawless, W., van Rijsbergen, C. & Klusch, M., eds (2009), *Proceedings of the Third Quantum Interaction Symposium, University of Saarbrücken*, Vol. 5494 of *Lecture Notes in Artificial Intelligence*, Springer, Saarbrücken.
- CODASYL (1971), Feature analysis of data base management systems, Technical report, ACM, New York.
- Codd, E. (1970), 'A Relational Model of Data for Large Shared Data Banks', *Communications of the ACM* **13**(6), 377–387.
- Codd, E. (1975), 'Recent investigations in relational database systems', *ACM Pacific* pp. 15–20.
- Dayal, U. & Bernstein, P. (1982), 'On the correct translation of update operations on relational views', *ACM Transactions on Database Systems* **8**(3), 381–416.
- Flender, C., Kitto, K. & Bruza, P. (2009), Beyond Ontology in Information Systems, in 'Proceedings of the 3rd International Symposium on Quantum Interaction', Springer, pp. 276–288.
- Furtado, A. & Casanova, M. (1985), 'Updating relational views', *Query Processing in Database Systems* pp. 127–144.
- Kakas, A. & Mancarella, P. (1990), Database updates through abduction, in 'Proceedings of the 16th International Conference on Very Large Databases', pp. 13–16.
- Kitto, K. (2008), Why quantum theory?, in 'Proceedings of the Second Quantum Interaction Symposium', University of Oxford.
- Lechtenbörger, J. & Vossen, G. (2003), 'On the computation of relational view complements', *ACM Transactions on Database Systems* **28**(2), 175–208.
- Piwowarski, B. & Lalmas, M. (2009a), A Quantum-based Model for Interactive Information Retrieval, in 'Proceedings of the 2nd International Conference on Theory of Information Retrieval', Springer, pp. 232–240.
- Piwowarski, B. & Lalmas, M. (2009b), Structured Information Retrieval and Quantum Theory, in 'Proceedings of the 3rd International Symposium on Quantum Interaction', Springer, pp. 289–298.



# Counting Distinct Objects over Sliding Windows

Wenjie Zhang<sup>1,2</sup>Ying Zhang<sup>1</sup>Muhammad Aamir Cheema<sup>1</sup>Xuemin Lin<sup>1,2</sup><sup>1</sup> the University of New South Wales {zhangw,yingz,macheema,lxue}@cse.unsw.edu.au<sup>2</sup> NICTA

## Abstract

Aggregation against distinct objects has been involved in many real applications with the presence of duplicates, including real-time monitoring moving objects. In this paper, we investigate the problem of counting distinct objects over sliding windows with arbitrary lengths. We present novel, time and space efficient, one scan algorithms to continuously maintain a sketch so that the counting can be approximately conducted with a relative error guarantee  $\epsilon$  in the presence of object duplicates. Efficient query algorithms have also been developed by effectively utilizing the skyband property. Moreover, the proposed techniques may be immediately applied to the range counting aggregation and heavy hitter problem against distinct elements. A comprehensive performance study demonstrates that our algorithms can support real-time computation against high speed data streams.

## 1 Introduction

Many recent applications in real-time monitoring moving objects require on-line counting of distinct objects. For instance, in real-time traffic management it is desirable to monitor the traffic volume of an area over a time frame; this is usually done by counting the number of distinct objects. In wireless communication management, the number of distinct users at a station is a key measurement of “popularity” of the area covered by the station. Counting distinct objects is also required in many other applications. For instance, in a stock market surveillance system, it is important to monitor the number of distinct clients in real-time in addition to the total number of transactions, the prices, etc. Moreover, counting distinct objects is a key component in an estimation of join results in join processing optimization; consequently on-line counting distinct objects may be applied to the join processing optimization among data streams.

In the above applications, datasets may be massive in size and fast in update speed. Therefore, in the context of real-time monitoring regarding the applications above, it is desirable to read database only once (i.e., one scan). Nevertheless, the challenge is that it is impossible to count the number of distinct elements by only one-scan of a dataset unless the whole dataset fits in memory. This makes it impractical to exactly counting distinct objects in real-time.

Flajolet and Martin [7] provide the first one-scan technique to build a sketch so that approximately counting distinct objects has the precision guarantee  $\frac{|n' - n|}{n} \leq \epsilon$  ( $\epsilon$ -approximation) with the confidence  $1 - \delta$  ( $\delta > 0$ ) and the space of  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log m)$  bits where  $n'$  is an approximation of  $n$  distinct elements, and  $m$  is the object domain size. Bar-Yossef *et al.* in [1] improves the efficiency of the technique of [7].

While the techniques cited above have been shown very space and time efficient to achieve  $\epsilon$ -approximation, they do not deal with the concept of *aging*. They are not applicable to *sliding windows*. On the other hand, there are a broad spectrum of applications where data objects observed early could be outdated and counting the most recently observed

(sliding windows) distinct data elements is more important. For instance, in a real-time traffic management, counting distinct vehicles over the most recent traffic may provide more accurate prediction towards traffic volume changes than that over all observed vehicle movements as traffic changes from time to time. Similarly, sliding windows are also crucial in the other two applications above.

Datar *et al.* [5] provide the technique of *exponential histogram* to approximately counting the number of objects over sliding windows with variable lengths. It requires the space  $O(\frac{1}{\epsilon} \log w)$  to enforce  $\epsilon$ -approximation where  $w$  is the maximum number of objects encountered by a sliding window. While space and time efficient, the technique is not applicable to counting distinct elements.

An efficient off-line technique is developed by Tao *et al.* in [13] to build space effective techniques to approximately count distinct objects against time intervals. In order to accommodate any time interval, it requires to create a sketch at each time-stamp which has a different sketch than the last sketch. Consequently, a huge storage space may still be required if there are massive sketch changes; thus, the techniques are applicable only to off-line computation against historical data. Gibbon [8] develops a novel one-scan *distinct sampling technique* to estimate the result size of an arbitrary query. Applying it to counting distinct elements over sliding windows requires a pre-fixed sample size of  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \sqrt{m})$  to enforce  $\epsilon$ -approximation with  $(1 - \delta)$  confidence, according to Theorem in [8].

Motivated by this, in this paper we present novel, time and space efficient, one-scan algorithms to continuously maintain sketches to approximately count the number of distinct objects against sliding windows of arbitrary lengths with  $\epsilon$ -approximation. Our contributions may be summarized as follows:

1. We develop novel, one-scan, space and time efficient sketch techniques, based on FM algorithms [7]. Our techniques guarantee  $\epsilon$ -approximation by  $1 - \delta$  confidence with the space of  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log m)$  machine words; this greatly reduces the space requirement in [8]. Consider that approximately counting distinct elements over sliding windows is much more sophisticated than that over whole streams. The space requirement of our techniques is near optimal as even counting a whole stream needs the space of  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log m)$  bits [1, 7] to ensure  $\epsilon$ -approximation with  $1 - \delta$  confidence.
2. The sketch technique above has  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$  sub-sketches and a query algorithm has to process every sub-sketch. Consequently, our query algorithm, against FM-based sketches above runs in  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log \log m)$  time. To speed up query processing time, we develop another time and space efficient sketch techniques based on the algorithm in [1]. To enforce  $\epsilon$ -approximation with  $1 - \delta$  confidence, our second sketch technique only requires  $O(\log \frac{1}{\delta})$  sub-sketches. By converting the sketch problem to the “ $k$ -skyband” problem [11], novel and efficient techniques have been developed to continuously maintain  $k$ -skyband in a  $2-d$  space when  $k$  is large. By effectively utilizing the property of a skyband, we show that our query algorithm runs in time  $O(\log \frac{1}{\delta} \log M)$  where  $M$ , the maximum size of a subsketch, is (expected)

$O(\frac{1}{\epsilon^2} \log n)$ . Consequently, our second query algorithm runs in (expected)  $O(\log \frac{1}{\delta} (\log \frac{1}{\epsilon} + \log \log n))$  time.

The rest of the paper is organised as follows. Section 2 presents problem definitions and related work. In Section 3, we present our first contribution of the paper. Section 4 presents the second contribution. In Section 5, we report our experiment results. Section 6 concludes the paper.

## 2 Background Information

We first state the problem. Then we present the related work.

### 2.1 Problem Statement

We model a set of observations by data streams. Each observation is treated as an element in a data stream and is represented by  $e = (x, t)$  where  $x$  is an object ID and  $t$  is the time-stamp. Note that for two data elements  $e = (x, t)$  and  $e' = (x', t')$ ,  $t > t'$  iff  $e$  comes later than  $e'$ ; that is,  $e$  is *younger* than  $e'$ . Although  $e \neq e'$ ,  $x$  could be the same as  $x'$ ; that is, the same object is observed twice at  $t$  and  $t'$ , respectively.

In a collection  $S$  of data elements, there may be many elements with the same object ID (e.g., objects are moving around and observed multiple times).  $D_S$  denotes the set of distinct object IDs in  $S$ . In this paper, we study the following counting problem.

**Distinct Counting against Sliding Window (DCSW).** For a given  $t$ , let  $S_t$  denote the data elements observed after (inclusive)  $t$  and  $D_{S,t}$  denote the set of distinct object IDs contained in  $S_t$ . We compute  $|D_{S,t}|$  for any  $t$  and denote  $|D_{S,t}|$  by  $n_{S,t}$ .

We investigate the problem of answering DCSW queries with the precision guarantee of  $\epsilon$ -approximation; that is, to enforce  $\frac{|n' - n|}{n} \leq \epsilon$  where  $n'$  is an approximate solution of  $n$ .

**Problem Description.** We investigate the problem of continuously maintaining a sketch (consisting of several sub-sketches) over a data stream  $S$  such that for any given  $t$ , the sketch can be used to return an  $\epsilon$ -approximate answer to  $n_{S,t}$ . The aim is to minimize the maximum memory space required in such a continuous computation, as well as to process high-speed data streams in real time.

### 2.2 Related Work

We briefly overview the techniques in [1, 7, 13]. They are most closely related to our work.

#### 2.2.1 FM Algorithm

Suppose that  $S$  is a collection of elements whose domain is  $\mathcal{D}$ . FM algorithm [7] proceeds as follows.

Let  $B$  be a bitmap of length  $k$  with subindexes  $[0, k-1]$ . Suppose that  $h()$  is a randomly generated hash function  $\mathcal{D} \rightarrow B$ , such that  $\forall x \in \mathcal{D}$ , 1) for each bit,  $h(x)$  has the equal opportunity to have 0 or 1, 2)  $h(x)$  is enforced to have one and only one bit with value 1, and 3)  $h(x)$  assigns the last bit (the bit with subindex  $k-1$ ) with value 1 iff the first  $k-1$  bits (from left) take value 0. To enforce property 2),  $h(x)$  may be interpreted as a serial binary hash functions that start from the first bit and terminate once the current bit is assigned by value 1. It can be immediately shown [2] that on average,  $h()$  runs in time  $O(1)$  (two calls of a binary hash function) per data element and the probability of having the  $i$ th bit with value 1 is  $\frac{1}{2^{i+1}}$ .

A FM sketch on  $S$  is defined as  $FM(S) = \bigvee_{x \in S} h(x)$ , where  $FM(S)$  is a bitmap with length  $k$  and the  $i$ th bit of  $FM(S)$  takes value 1 iff  $\exists x \in S$  such that  $h(x)$  assigns the value 1 to the  $i$ th bit. We define  $FM_{min}(S)$  as follows:

- If  $i$  is the least bit (from left) with value 0,  $FM_{min}(S)$  is defined as  $i$ .
- Otherwise,  $FM_{min}(S)$  is defined as  $\infty$  (in our implementation, we define  $FM_{min}(S)$  as  $k$ ).

To improve the accuracy of FM algorithm, multiple copies (say,  $l$ ) of FM sketches are constructed. Therefore, each data element is hashed into  $l$  FM sketches,  $FM_1(S), FM_2(S), \dots, FM_l(S)$ , respectively. The number  $n_S$  of distinct elements in  $S$  is estimated by:

$$A_S = \frac{1}{\varphi} 2^{\sum_{i=1}^l FM_{i,min}(S)/l}. \quad (1)$$

Here,  $\varphi \stackrel{\text{def}}{=} 2^{E(FM_{1,min}(S))/n_S}$ . Note that as  $E(FM_{1,min}(S))$  cannot be explicitly represented and  $n_S$  is unknown, in our implementation we approximately choose  $\varphi$  as 0.775351 according to the approximate results in [7]. Each  $FM_{i,min}(S)$  related to  $FM_i(S)$  is defined in the same way as  $FM_{min}(S)$  related to  $FM(S)$ . As shown in [7],  $E(FM_{i,min}(S)) = E(FM_{j,min}(S))$  ( $1 \leq i < j \leq l$ ). From the insight in Section 3.2 in [4], Theorem 2 in [7], and the *Central Limit Theorem* (pp 229 in [6]), the following lemma can be immediately verified using the independence assumption.

**Lemma 1** Suppose that  $A_S$  is returned by FM algorithm as shown in (1). Then,  $P(|A_S - n_S| < \epsilon n_S) \geq 1 - \delta$ , for any given  $0 < \delta < 1$  and  $0 < \epsilon < 1$ , if  $n_S > \frac{1}{\epsilon}$ ,  $k = O(\log m + \log \epsilon^{-1} + \log \delta^{-1})$ , and  $l = O(\frac{1}{\epsilon^2} \log \delta^{-1})$ , where  $m = |\mathcal{D}|$ .

#### 2.2.2 BJKST algorithm

In [1], a novel variation of FM algorithm, BJKST algorithm, has been proposed to speed-up the computation, while the accuracy and the space-efficiency can be retained. It proceeds as follows. First, we pick at random a pairwise independent hash function  $h$  to hash  $\mathcal{D}$  to  $[1, m^3]$  where  $\mathcal{D}$  is the domain of data elements  $x$  and  $|\mathcal{D}| = m$ . The following Lemma has been shown as folklore.

**Lemma 2** If  $m \geq \delta^{-1}$  then  $h$  is injective over  $S$  with probability at least  $1 - \delta$ .

Based on this, BJKST algorithm always keeps the  $k$  smallest elements (i.e. with the  $k$  smallest distinct hash values) and uses the following  $A_S$  to estimate  $n_S$

$$A_S = \frac{k \times m^3}{f_{k-\min}}. \quad (2)$$

Here,  $f_{k-\min}$  is the  $k$ th smallest distinct hash value. If there are less than  $k$  distinct values, then  $A_S = \infty$  (in our implementation, we assign  $A_S$  as  $k'$  ( $k' \leq k$ ) if there are only  $k'$  distinct values). To improve the accuracy, BJKST algorithm picks at random  $l$  pairwise independent hash functions  $h_i$  (hashing  $\mathcal{D}$  to  $[1, m^3]$ ), and outputs  $A_{i,S}$  for each  $h_i$  where  $A_{i,S}$  (for  $1 \leq i \leq l$ ) related to  $h_i$  is defined in the same way as  $A_S$  related to  $h$ . BJKST algorithm outputs  $A_S$  as the median of these  $A_{i,S}$  to estimate  $n_S$ . BJKST algorithm keeps only  $k$  elements with the  $k$  smallest distinct hash values. The following Lemma 3 has been proved in [1].

**Lemma 3** Suppose that  $0 < \epsilon, \delta < 1$ . If  $m \geq \delta^{-1}$ ,  $k = O(\frac{1}{\epsilon^2})$ ,  $l = O(\log \delta^{-1})$ , and  $n_S \geq k$ , then  $P(|A_S - n_S| < \epsilon n_S) \geq 1 - \delta$ .



### 2.2.3 Spatio-Temporal Aggregation

Regarding counting distinct spatial objects intersecting a spatial region, Tao *et al.* [13] observe that the space required to exactly count distinct spatial objects intersecting a spatial region is  $\Omega(N)$  where  $N$  is the number of observations. To reduce space, they proposed to use FM algorithm to build space-efficient sketches to approximately conduct the computation.

To support an arbitrary time interval, an FM sketch is constructed against a batch of new observations with the same time stamp. A sketch at time-stamp  $t_2$  is kept if it is different than that generated at time  $t_1$  where  $t_1$  is largest time-stamp but smaller than  $t_2$  (i.e.  $t_1$  and  $t_2$  are consecutive). Consequently,  $O(N)$  FM sketches are required to be maintained with the total space  $O(N \frac{1}{\epsilon} \log \frac{1}{\delta} \log m)$  to guarantee  $\epsilon$ -approximation where  $N$  is the number of observation batches. To support efficient off-line computation, an aRB like [10] index is developed in [13]. Clearly, it is quite space efficient when  $N \ll N$ . Nevertheless, the technique is not applicable to data streams where  $N$  often equals  $O(N)$ .

### 3 FM-based Sketches

Our techniques are based on the following observation. For a dataset  $S$ , if we first select the data elements from  $S$  with time-stamp values not smaller than a given  $t$  (the result is denoted by  $S|_{t+}$ ) and apply FM Algorithm on  $S|_{t+}$ , then the obtained estimation  $A_{S,t}$  of the number  $n_{S,t}$  of distinct objects in  $S|_{t+}$  follows Lemma 1.

Below, we first present a novel, space-efficient data structure (sketch) by using FM algorithm. Then, we present space- and time-efficient algorithms to continuously maintain such sketches to achieve an  $\epsilon$ -approximation.

#### 3.1 The Framework

The following example illustrates the basic idea in our framework based on FM algorithm.

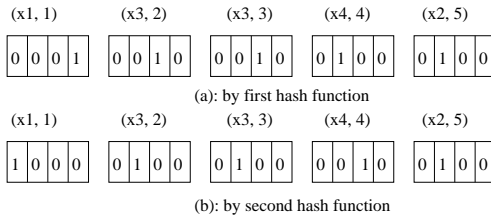


Figure 1: An Example

As shown in Figure 1, there are 5 data elements  $(x, t)$  where the second and the third represent two occurrences of the same object  $x_3$ . Suppose that in FM algorithm  $l = 2$  and  $k = 4$ . Two hash functions  $h_1$  and  $h_2$  are randomly picked to hash each element, respectively. The total 10 bitmaps with length 4 are generated, respectively, by  $h_1$  and  $h_2$ , as depicted in Figure 1(a)-(b).

To effectively keep values information, we map a bitmap into an array by replacing the bit with value 1 by its corresponding time-stamp. Figure 2 shows the corresponding arrays converted from the bitmaps in Figure 1.

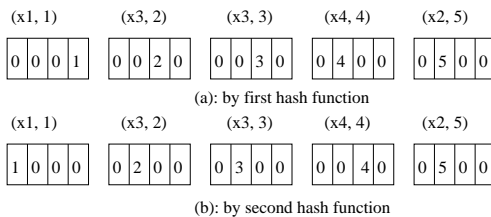


Figure 2: Transformed from Figure 1

For a time  $t$  and a hash function  $j$  ( $j$ th subsketch),

to estimate  $n_{S,t}$  (number of distinct elements arriving no earlier than  $t$ ) by using FM algorithm we first select the arrays generated by  $h_j$  such that their corresponding non-zero values not smaller than  $t$ , then find the left-most common element with value 0 and return its subindex as  $f_{j,t}$ .

**Example 1** Let  $t = 2$  and  $j = 1$ . Regarding Figure 2 (a), the 2nd array, 3rd array, 4th array, and 5th array are selected. Then,  $f_{1,2} = 0$  (i.e. the subindex of the left-most common element, 1st element, in these arrays). Here, the 2nd and 4th arrays are redundant.

Clearly, computing  $f_{j,t}$ , by this way, is equivalent to what have been discussed in the beginning of this section; that is, we do a selection on  $S$  to output  $S|_{t+}$ ; then apply  $h_j$  on  $S|_{t+}$  and use FM Algorithm to get  $FM_{j,min}(S|_{t+}) (= f_{j,t})$ . Moreover, this example also demonstrates that if two arrays have non-zero values allocated in the same position, the one with smaller time-stamp value (older) will never be used in any query (i.e., regarding any  $t$ ); consequently, this redundant array should be removed. Therefore, in the worst case we only keep  $k$  arrays where  $k$  is the length of bitmaps in the hash functions. Furthermore, after removing redundant arrays the remaining arrays generated by  $h_j$  can be merged into one array with non-zero values remain in the same positions, respectively.

**Example 2** Regarding the example in Figure 2(a), 2nd and 4th arrays are redundant and thus, are removed. The merged result is depicted in Figure 3(a). For the example in Figure 2(b), 2nd and 3rd arrays are redundant. The merged result is depicted in Figure 3(b).

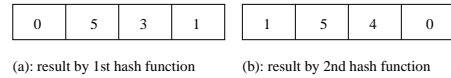


Figure 3: Compressed from Figure 2

Below, we present our continuous sketch construction and maintenance algorithm in Algorithm 1. We maintain  $l$  arrays  $\{s_i : 1 \leq i \leq l\}$  each of which is generated, as described above, by a randomly picked hash function  $h_i$ , and has  $k$  elements with subindexes from 0 to  $k - 1$ . Note that the time-stamp of an element takes a positive value. Thus, each array  $s_i$  can be initialized to  $(0, 0, \dots, 0)$ . For every  $h_i(x)$  ( $1 \leq i \leq l$ ),  $\rho(h_i(x))$  denotes the position (subindex) of the bit, with value 1, in  $h_i(x)$ . Note that  $s_i[\rho]$  is the  $\rho$ -th element in  $s_i$ . Moreover, to ensure  $\epsilon$ -approximations for sliding windows with the number of distinct objects less than  $\frac{1}{\epsilon}$  precise answers are the only possibility; consequently, we always keep the  $L$  most recent “distinct” objects (i.e., with the largest time-stamp values) in  $\mathcal{L}$  in addition to  $\{s_i : 1 \leq i \leq l\}$ , so that counting the number of most recent distinct objects, which is smaller than  $L$ , can be answered exactly. We use  $t_{sml}$  to denote the smallest time-stamp value in  $\mathcal{L}$  and  $x_{sml}$  is the element with the time-stamp. Note that in  $\mathcal{L}$  we keep each element  $(x, t)$ . The following theorem is immediate.

**Theorem 1** Algorithm 1 requires the space of  $L + l \times k$  elements.

To estimate  $n_{S,t}$  for a given  $t$ , our query algorithm proceeds as follows. If  $t > t_{sml}$  then we only query  $\mathcal{L}$ . Otherwise, in the light of earlier discussions we first select the elements in  $s_i$  with positive time-stamps (corresponding to objects in  $S|_{t+}$ ) not smaller than  $t$ ; the result is denoted by  $s_i|_{t+}$ . Then, we return the location of the left-most element in  $s_i$  that is not included in  $s_i|_{t+}$ . If such a left-most element does not exist, we return  $k$  (corresponding to the situation  $\infty$ ).

**Algorithm 1** Space-Efficient Sketches (SE-FM)**Input:**  $l, k, L$ , a stream  $S$  of  $(x, t)$ .**Output:**  $\mathcal{L}$ : the set of  $L$  most recent distinct objects;  
 $\{s_i : 1 \leq i \leq l\}$ : each  $s_i$  is an array with  $k$  elements.**Description:**

- 1: Initialize  $\{s_i : 1 \leq i \leq l\}; \mathcal{L} \leftarrow \emptyset; j \leftarrow 0;$
- 2: Generate  $l$  hash functions  $\{h_i() : 1 \leq i \leq l\};$
- 3: **for** each new  $(x, t)$  **do**
- 4:   **if**  $\exists (x_1, t_1) \in \mathcal{L}$  s.t.  $x_1 = x$  **then**
- 5:     replace  $(x_1, t_1)$  by  $(x, t);$
- 6:   **else**
- 7:     **if**  $j < L$  **then**  $\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, t)\}; j \leftarrow j + 1;$
- 8:     **else** remove  $(x_{sml}, t_{sml})$  and add  $(x, t)$  in  $\mathcal{L};$
- 9:   **for**  $i = 1$  to  $l$  **do**
- 10:      $\rho \leftarrow \rho(h_i(x));$
- 11:      $s_i[\rho] \leftarrow t;$
- 12: **Return**  $\mathcal{L}$  &  $\{s_i : 1 \leq i \leq l\}.$

when we presented FM Algorithm). Let  $\Pi$  denote a subset of elements in an array and  $I(\Pi)$  denote the set of subindexes of the elements in  $\Pi$ . Our query algorithm is presented in Algorithm 2.

**Algorithm 2** Approximating  $n_{S,t}$ **Input:**  $t, \mathcal{L}, \{s_i : 1 \leq i \leq l\}$  generated by Algorithm 1;**Output:**  $A_{S,t};$ **Description:**

- 1: get  $t_{sml}$  from  $\mathcal{L};$
- 2: **if**  $t_{sml} < t$  **then**
- 3:    $A_{S,t} \leftarrow |\mathcal{L}|_{t+};$
- 4: **else**
- 5:   **for**  $i = 1$  to  $l$  **do**
- 6:     **if**  $[0, k-1] - I(s_i|_{t+}) \neq \emptyset$  **then**
- 7:        $f_{i,t} \leftarrow \min\{j : j \in [0, k-1] - I(s_i|_{t+})\};$
- 8:     **else**
- 9:        $f_{i,t} = k;$
- 10:    $A_{S,t} \leftarrow \frac{1}{\varphi} 2^{\sum_{i=1}^l f_{i,t}/l};$
- 11: **Return**  $A_{S,t}.$

Similar to Lemma 1, the following Lemma holds for every pair of  $A_{S,t}$  and  $n_{S,t}$  if  $L = \frac{1}{\epsilon}$ .

**Lemma 4** For a given  $t, \epsilon$ , and  $\delta$ ,  $A_{S,t}$  returned by Algorithm 2 against the output of Algorithm 1 has the property that  $P(|A_{S,t} - n_{S,t}| < \epsilon n_{S,t}) \geq 1 - \delta$  if  $L = \frac{1}{\epsilon}$ ,  $l = O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ , and  $k = O(\log m + \log \delta^{-1} + \log \epsilon^{-1})$ .

**Proof 1** If  $A_{S,t}$  is returned from only counting  $\mathcal{L}$ , it is the exact answer. The lemma is immediate.

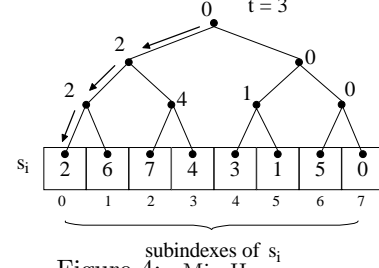
Consider that  $A_{S,t}$  is returned from  $\{s_i : 1 \leq i \leq l\}$ . It can be immediately verified that Algorithm 2, in this case, is equivalent to: 1) doing a select on  $S$  to output  $S|_{t+}$ , and then 2) applying FM algorithm on  $S|_{t+}$ , 3)  $n_{S,t} > \frac{1}{\epsilon}$ . According to Lemma 1, this lemma is also immediate.

Lemma 4 and Theorem 1 together imply that Algorithm 1 and Algorithm 2 guarantee the  $\epsilon$ -approximation with probability (confidence)  $1 - \delta$  if  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log m)$  space is allocated when  $m > \epsilon^{-1}$  and  $m > \delta^{-1}$ .

**3.2 Time Complexity**

Suppose that we treat an element with value 0 in  $s_i$  as having the time-stamp 0. The query algorithm (Algorithm 2), to compute  $f_{i,t}$  for each  $i$  and a given  $t$ , selects the sub-index of the left-most element, from  $s_i$ , with the time-stamp smaller than  $t$ . Below we show it can be conducted in a logarithmic time if a min-heap is maintained.

As illustrated in Figure 4, each  $s_i$  is organized by a *min-heap* [3] built against the time-stamp values on a binary balanced tree. Then, we search the heap according to the order of depth first following the left-most path from the root satisfying the criterion - the search key value is smaller than  $t$ . For instance, for  $t = 3$  the search returns the first element; thus,  $f_{i,t} = 0$  (the subindex of 1st element). It can be immediately verified that such a search can be done in  $O(\log k)$ . Consequently, Algorithm 2 can run in time  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log \log m)$  if  $m > \epsilon^{-1}$  and  $m > \delta^{-1}$  since there are  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$  subsketches and  $k = O(\log m)$ .

Figure 4: Min-Heap on  $s_i$ 

Algorithm 1 runs in time  $O(\log \frac{1}{\epsilon})$  per element to dynamically maintain  $\mathcal{L}$  if we maintain a search tree on  $\mathcal{L}$ . As discussed earlier, each  $h_j()$  ( $1 \leq j \leq l$ ) takes constant time on average to hash a data element. Thus, Algorithm 1 runs in time  $O(\frac{1}{\epsilon^2} \log \delta^{-1} \log \log m)$  on average per data element, given there are  $O(\frac{1}{\epsilon^2} \log \delta^{-1})$  such arrays and maintaining such a min-heap takes  $O(\log \log m)$  time.

**3.3 PCSA-like Algorithm**

While Algorithm 1 is space-efficient and guarantees a probabilistic relative  $\epsilon$ -approximate, each element is hashed into  $\Omega(\frac{1}{\epsilon^2} \log \delta^{-1})$  arrays (subsketches). This potentially makes the algorithm less efficient. Our performance study in Section 6 demonstrates it can only handle a medium speed data stream in real-time.

Consider that in many recent applications, to support on-line computation of high speed data streams is a crucial requirement. In this subsection, we propose a time-efficient algorithms following the framework in the last section. It retains the space requirement but there is no theoretical guarantee of accuracy with confidence  $1 - \delta$  though the expected accuracy is  $\epsilon$ -approximate. Our performance study, nevertheless, indicates the algorithm is practically very space-efficient and highly accurate, and it is able to support on-line computation of high-speed data streams.

The algorithm is an immediate application of PCSA technique [7] to our algorithm, Algorithm 1. The basic idea is to hash each data element randomly to  $\zeta$  arrays (subsketches) among the  $l$  arrays (subsketches). Algorithm 1 may be modified as follows.

- First, we pick at random another  $\zeta$  hash functions:  $\{H_i : 1 \leq i \leq \zeta\}$  besides the  $l$  hash functions in Algorithm 1, where each  $H_i$  hashes the element domain  $\mathcal{D}$  to  $[1, l]$ .
- Then, in Algorithm 1 instead of the iteration (in line 9) from  $i = 1$  to  $l$ , we do the iteration for each  $i \in \{H_1(x), H_2(x), \dots, H_\zeta(x)\}$ . The others in Algorithm 1 remain the same.

We call such a modified Algorithm 1 “Algorithm SE-PCSA”. Suppose that all the parameters are selected as those in Lemma 4. It is immediate Algorithm SE-PCSA runs in time  $O(\zeta \log \log m)$  for each data element.

In the light of PCSA technique, Algorithm 2 is modified accordingly as follows to estimate a  $n_{S,t}$ . We change line 10 in Algorithm 2 to  $A_{S,t} \leftarrow \frac{1}{\zeta \varphi} 2^{\sum_{j=1}^{\zeta} f_{j,t}/l}$ . It can be implemented in the same

way as what we described in Section 3.2 with the same time complexity. These, together with the facts in [7], immediately imply that the expected accuracy of Algorithm SE-PCSA is  $\epsilon$ -approximate. Note that in our implementation, we use a pairwise independent hash function for  $H_i$  and our experiment indicates that when  $\zeta$  approaches 100, its accuracy remains quite stable.

#### 4 K-skyband Technique

While the PCSA-like algorithm speeds up the sketch maintenance algorithm by hashing each element into a small number  $\zeta$  ( $\zeta \leq 100$  in our implementation) of randomly selected FM-based subsketches, it still retains  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$  subsketches to pursue  $\epsilon$ -approximation. Consequently, the query algorithm runs in  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log \log m)$  time; it is not very efficient when  $\epsilon$  is small. This prevents us from processing a large number of queries simultaneously in real-time.

Our second sketch technique is based on BJKST algorithm since BJKST requires only  $O(\log \frac{1}{\delta})$  subsketches to achieve  $\epsilon$ -approximation with  $1 - \delta$  confidence. We will show that applying BJKST algorithm to our problem DCSW leads to the  $k$ -skyband problem. Consequently, we show our sketch technique requires the (expected) space  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log n)$ . Our experiment indicates that in practice, it requires much less space than that required by SE-FM (Algorithm 1) or PCSA-like technique. Moreover, this sketch technique enables us to develop an efficient query algorithm with  $O(\log \frac{1}{\delta} (\log \frac{1}{\epsilon} + \log \log n))$  (expected) time in contrast to  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log \log m)$ .

##### 4.1 Outline

We maintain  $l$  subsketches. Without loss of generality, we assume the domain  $\mathcal{D}$  of object IDs is  $[1, m]$ . For each sketch  $i$ , a randomly generated pairwise hash function  $h_i$  hashes  $[1, m]$  to  $[1, m^3]$  so that each data element  $(x, t_x)$  is hashed to  $s_i$  by the form  $(x, h_i(x), t_x)$  for all  $i$ .

**Query.** Similar to our FM-based algorithm, we may divide a query algorithm logically into two parts. For a given  $t$ , we first select all elements with time-stamps not smaller than  $t$ ; that is, get  $S|_{t+}$ . Then, we apply BJKST algorithm to  $S|_{t+}$ . More specifically, in each  $s_i$  we choose the set  $s_i|_{t+}$  of elements with time-stamps not smaller than  $t$ . Then, we apply the BJKST query method, as described in (2) and the last paragraph of Section 2.2.2 on all  $s_i|_{t+}$  for  $1 \leq i \leq l$ .

To address the accuracy guarantee not covered by Lemma 3 when  $n_{S,t} < k$  (e.g. a sliding window with the number of distinct objects less than  $k$ ), we globally maintain a  $\mathcal{L}$  to store the most recent  $k$  distinct objects, that is, the  $k$  distinct objects with the largest time-stamp values. Then, for each  $t$  we first find out if  $t > t_{sml}$  where  $t_{sml}$  is the smallest time-stamp (oldest object) among the elements in  $\mathcal{L}$ . If  $t > t_{sml}$ , then query  $\mathcal{L}$ ; otherwise query all  $s_i$  as above.

Let the query result obtained above be denoted by  $A_{S,t}$ . We have the following Theorem -  $\epsilon$ -approximation.

**Theorem 2** Suppose that  $0 < \epsilon, \delta < 1$ . If  $m \geq \delta^{-1}$ ,  $k = O(\frac{1}{\epsilon^2})$ ,  $l = O(\log \delta^{-1})$ , then  $P(|A_{S,t} - n_{S,t}| < \epsilon n_S) \geq 1 - \delta$ .

**Proof 2** If  $A_{S,t}$  is obtained from  $\mathcal{L}$ , then it is the exact solution. The Theorem holds.

If  $A_{S,t}$  is obtained from  $s_i$ , then it can be viewed as if we applied BJKST algorithm on  $S|_{t+}$ . Therefore, the theorem holds according to Lemma 3.

**Sketch Maintenance.** As with Algorithm 1, once a new element comes we first examine if an element  $(x, t_x)$  in  $\mathcal{L}$  needs to be replaced by the new element. Meanwhile, we hash the new element  $(x, t_x)$  into  $\{s_i : 1 \leq i \leq l\}$  by the form  $(x, h_i(x), t_x)$ .

Below, we show that in each  $s_i$ , we do not have to keep every element. We only need to keep “ $k$ -skyband”.

##### 4.2 Space Minimization

A hashed data element  $(x, h_i(x), t_x)$  is kept in the sketch of  $s_i$  (for  $1 \leq i \leq l$ ) if  $s_i$  does not “ $k$ -dominate”  $(x, h_i(x), t_x)$ . An  $s_i$   $k$ -dominates  $(x, h_i(x), t_x)$  iff there are  $k$  data elements in  $s_i$  with distinct hash values not greater than  $h_i(x)$  and their time-stamp values not smaller (not older) than  $t_x$ .

The *distinct value based  $k$ -skyband* of  $s_i$  is the set, denoted by  $SK(s_i)$ , of data elements in  $s_i$  that are not  $k$ -dominated by  $s_i$ .<sup>1</sup> In our sketch algorithm, we maintain the distinct value based  $k$ -skyband  $SK(s_i)$ , instead of  $s_i$ , for  $1 \leq i \leq l$ . To simplify the notation, we abbreviate “distinct value based  $k$ -skyband” to “ $k$ -skyband” hereafter in this paper.

**Theorem 3** Each current  $SK(s_i)$  for  $1 \leq i \leq l$  has the following properties.

**P1:** If  $s_i$  currently  $k$ -dominates an element  $e (\in s_i)$ ,  $e$  will never be used by our query algorithm above for any  $t$ .

**P2:** For each element  $e = (x, h_i(x), t_x) \in SK(s_i)$ , either

**P2a:** there is a  $t_0$  such that  $h(x)$  is the  $k$ th smallest among the elements in  $D_{s_i, t_0^+}$  where  $D_{s_i, t_0^+}$  denotes the set of elements in  $s_i$  with distinct hash values and time-stamp values at least  $t_0$ , or

**P2b:**  $h(x)$  is one of the  $k-1$  smallest distinct hash values in  $s_i$ .

**Proof 3** We prove P1 and P2 as follows.

**Proof of P1.** According to the definition, if  $e = (x, h_i(x), t_x)$  is  $k$ -dominated by  $s_i$  then we have the property that there are at least  $k$  elements in  $s_i$  with distinct hash values not greater than  $h_i(x)$  and come after (inclusive)  $t$ . This property will be retained regardless how many new elements come. Consequently, our query algorithm will never choose  $h_i(x)$ . Thus, P1 holds.

**Proof of P2.** If  $(x, h_i(x), t_x)$  does not belong to category P2b, then there are  $\lambda$  ( $\lambda > k-1$ ) elements in  $s_i$  with distinct hash values smaller than  $h_i(x)$ .

Let  $t_0$  is the time-stamp of the element with the  $(k-1)$ th largest time-stamp value among these  $\lambda$  elements. Since  $e$  is in  $SK(s_i)$ , among these  $\lambda$  elements there are only  $\lambda_1$  ( $\lambda_1 \leq k-1$ ) elements with time-stamp values larger than  $t_x$ . Therefore,  $t_0 \leq t_x$ ; that is,  $e$  belongs to category P2a.

Note that P1 in Theorem 3 implies that we only need to maintain  $SK(s_i)$ . Clearly, an element in the category P2a will be used in a DCSW query with time  $t_0$ . Moreover, any element with one of the  $k-1$  smallest distinct hashed values (category P2b) may be used in processing DCSW for the whole stream once future elements have hashed values smaller than the current  $k-1$  smallest values; thus, it needs to be kept. Therefore, Theorem 3 implies that  $SK(s_i)$  is the minimum number of elements we should keep to achieve  $\epsilon$ -approximation. In fact, we can also show that  $SK(s_i)$  (for  $1 \leq i \leq n$ ) has the following (expected) space.

<sup>1</sup>The problem is the same as “ $k$ -Skyband” in [11] if we focus on  $(h(x), t_x)$  only, except we enforce distinct values.

**Theorem 4** In a 2-d set  $s = \{(x_i, y_i) : 1 \leq i \leq n\}$  with  $n$  elements, assume all  $x$  and  $y$  values are unique,  $x$  and  $y$  are independent, each  $x$  follows a same distribution, and each  $y$  also follows a same distribution. Then, the  $k$ -skyband  $SK(s)$  has the expected number of elements  $O(k \ln(\frac{n}{k}))$  where  $x_i$  corresponds to a hashed value and  $y_i$  corresponds to a time-stamp.

**Proof 4** Without loss of generality, we assume that  $y_i > y_j$  if  $i < j$ .

For  $1 \leq i \leq n$ , let the random variable  $X_i = 1$  if  $(x_i, y_i)$  is a  $k$ -skyband element, otherwise,  $X_i = 0$ . The expected number of  $k$ -skyband elements is  $E(\sum_{i=1}^n X_i) = \sum_{i=1}^n P(X_i = 1)$  where  $P$  denotes the probability.

Clearly, the value (0 or 1) of each  $X_i$  (for  $1 \leq i \leq n$ ) depends on  $\{(x_j, y_j) : 1 \leq j \leq i-1\}$  as  $y_j$  is decreasingly ordered and any element  $(x_j, y_j)$  for  $j > i$  does not dominate  $(x_i, y_i)$ .<sup>2</sup> Note that every element  $(x_i, y_i)$  when  $i \leq k$  belongs to  $SK(s)$ ; thus,  $P(X_i = 1) = 1$  when  $i \leq k$ .

For  $i > k$ ,  $(x_i, y_i)$  is a  $k$ -skyband element iff  $x_i$  is one of the  $k$  smallest values in  $\{y_j : 1 \leq j \leq i\}$ . Note that each  $y_j$  has the same probability to fall into the  $k$  smallest values as each  $y_j$  follows the same distribution, and we assume the independence among all  $y_j$  and between  $x$  and  $y$ . Thus,  $P(X_i = 1) = \frac{k}{i}$ .

It can be immediately verified

$$E(\sum_{i=1}^n X_i) = k + \sum_{i=k+1}^n P(X_i = 1) = k \times (1 + H_{1,n} - H_{1,k}).$$

Here,  $H_{1,n} = \ln(n)$ , the Theorem immediately follows.

From Theorems 2 and 4, it follows that to achieve  $\epsilon$ -approximation we need (expected)  $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \log n)$  space with the high probability (confidence)  $1 - \delta$  if all time-stamps are unique, objects and time-stamps are independent, object ID of each element follows the same distribution, and the time-stamp of each element follows the same distribution. Our experiment demonstrates that in practice, this algorithm requires a smaller space than the FM-based techniques in Section 3.

### 4.3 Query Algorithm

Keeping only  $SK(s_i)$  ( $1 \leq i \leq l$ ) not only minimize the information kept but also enables us to develop an efficient query algorithm. Below, we show that querying each  $s_i$  for  $t$  by searching  $SK(s_i)$  takes  $O(\log |SK(s_i)|)$  time only.

For a given  $t$ ,  $s_i$  has two cases: A)  $s_i$  has at least  $k$  elements with distinct hashed values and time-stamp values not smaller than  $t$ ; B) not A). The theorem below is a key.

**Theorem 5** For a given  $t$  and  $i$ , suppose that  $e_1$  is the element in  $SK(s_i)$  with the smallest time-stamp among all elements arriving no earlier than  $t$ , and the time-stamp of  $e_1$  is the  $r_1$ th smallest in  $SK(s_i)$ . Then,  $s_i$  belongs case A) if and only if the element  $e_2$  with the  $(r_1 + k - 1)$ th smallest hashed values in  $SK(s_i)$  has the  $k$ th smallest hashed values among the elements in  $SK(s_i)$  coming no earlier than  $t$ .

**Proof 5** First, we can show that for any element  $e \in SK(s_i)$  its hashed value always falls in the  $(r+k-1)$ th smallest values of all hashed values in  $SK(s_i)$  where  $e$  has the  $r$ th smallest time-stamp in  $SK(s_i)$ . This can

<sup>2</sup>  $(x, y)$  dominates  $(x', y')$  iff  $x \leq x'$  and  $y \geq y'$ .

be immediately shown by mathematic induction from the element with the smallest time-stamp in  $SK(s_i)$  based on the fact that  $e$  is not  $k$ -dominated by  $SK(s_i)$ .

The above fact immediately implies that in  $SK(s_i)$  if we remove from  $SK(s_i)$  all elements ( $r_1 - 1$  in total) with the time-stamp values smaller than  $t_1$  then  $e_2$  has the  $k$ th smallest values among the remaining elements.

Based on Theorem 5 and in the light of our BJKST-based query algorithm, for any  $t$  we only need to find such an  $e_1$  against time-stamps to determine  $r_1$  and then find  $e_2$  against hashed values. If such  $e_2$  does not exist (i.e., less than  $k$  elements in  $SK(s_i)$  with time-stamps not smaller than  $t$ ), then we return  $(n - r_1 + 1)$  according to BJKST query algorithm in Section 2.2.2.

To speed-up the search, we continuously maintain a binary search tree  $eT$  on  $t$  and a binary search tree  $eV$  on  $h(x)$  with the information of the number of elements in each subtree, respectively, for the elements in  $SK(s_i)$  for  $1 \leq i \leq l$ . It is clear with such search trees, the query algorithm above can be done in  $O(\log |SK(s_i)|)$  time. Regarding the expected size of  $SK(s_i)$ , to achieve  $\epsilon$ -approximation with confidence  $1 - \delta$ , our query algorithm runs in (expected) time  $O(\log \frac{1}{\delta} (\log \frac{1}{\epsilon} + \log \log n))$  as there are  $O(\log \frac{1}{\delta})$  subsketches.

Below we describe our query algorithm in Algorithm 3.

### Algorithm 3 Query Algorithm

**Input:** Query time  $t$ ,  $\{SK(s_i), eT_i, eV_i : 1 \leq i \leq l\}$ ;

**Output:**  $A_{s,t}$ ;

**Description:**

```

1: get  $t_{sml}$  from  $\mathcal{L}$ ;
2: if  $t > t_{sml}$  then
3:   return  $|\mathcal{L}|t^+$ ;
4: else
5:   for  $i = 1$  to  $l$  do
6:     compute  $r_1$  regarding  $t$  against  $eT_i$ ;
7:     if  $r_1 + k - 1 \leq |s_i|$  then
8:       compute the  $(r_1 + k - 1)$ th smallest hashed value  $v_i$ 
       against  $eV_i$ ;
9:        $A_i \leftarrow \frac{k \times m^3}{v_i}$ ;
10:    else
11:       $A_i \leftarrow (|s_i| - r_1 + 1)$ ;
12:   return the median of  $\{A_i\}$  as  $A_{s,t}$ ;
```

### 4.4 Sketch Maintenance

We present our techniques to continuously maintain each  $SK(s_i)$  for  $1 \leq i \leq l$ , as well as  $eT_i$  and  $eV_i$ . Clearly, if we have determined which element should be added or removed from  $SK(s_i)$ , the  $eT_i$  and  $eV_i$  can be updated (insertion or deletion) in time  $O(\log |SK(s_i)|)$  per element by using the standard tree rebalancing technique in [3]. While every new element has to be added into  $SK(s_i)$ , below we present an efficient technique, by effectively utilizing  $eV_i$ , to determine whether or not elements in  $SK(s_i)$  should be removed.

An immediate way is to compute the *total dominance count* for each element  $e$ ; that is, count the number of elements in  $SK(s_i)$  that dominate  $e$ . Nevertheless, in our problem setting there may be many elements in each  $SK(s_i)$  since  $k$  has to be  $O(\frac{1}{\epsilon^2})$  to guarantee  $\epsilon$ -approximation. In addition, an element may dominate many other elements. Therefore, simply computing dominance count for each element is too expensive; our experiment in Section 6 confirms this.

In fact, we do not have to count dominance for each element; instead, many times we can record the dominance count for a group of elements.

**Example 3** Let  $k = 2$  and  $l = 1$ . As depicted in Figure 5(a), suppose the elements  $e_1, e_2, \dots, e_8$  have been hashed into  $s_1$  where  $x$ -coordinators are hashed values  $h(e_i)$  and  $y$ -coordinators are time-stamps (the larger,

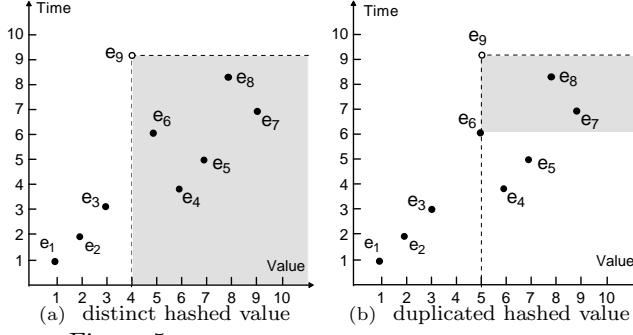


Figure 5: Example for two kind of dominate

the younger). It can be verified that before  $e_9$  arrives, these 8 elements form 2-skyband  $SK(s_1)$ . Once  $e_9$  is in, according to the definition it dominates the elements from  $e_4$  to  $e_8$ . Instead of updating dominance count of each element from  $e_4$  to  $e_8$ , it is possible to group them into several groups and then adding dominance counts on each group to avoid visit each individual element.

Below, we first present an augmentation of  $eV_i$  to speed-up our computation of continuously maintaining  $SK(s_i)$  for  $1 \leq i \leq l$ .

#### 4.4.1 Dominance Aggregation Search Tree

We augment an  $eV_i$  ( $1 \leq i \leq l$ ) tree to a dominance aggregation search tree (dAs-tree). As depicted in Figure 6, a balanced binary search tree is maintained on hashed values over the 8 elements from  $e_1$  to  $e_8$ . Since the number of elements in a subtree is irrelevant to this part and they can be maintained in a similar way as search keys, we omit them in our discussions here.

To simplify the presentation, we enforce the constraint that all elements are at the leaf-levels, and each intermediate node has two children; that is, an AVL-like tree but all elements allocated at leaf nodes. At each node, we keep 5 search values  $h$ ,  $\alpha$ ,  $\beta$ ,  $t_{min}$ , and  $t_{max}$ . Here,  $h$  stores the search key value built on all hashed values,  $t_{min}$  stores the minimum time-stamp, and  $t_{max}$  stores the maximum time-stamp in the subtree. In addition, at each node  $j$ ,  $\alpha_j$  denotes the captured dominance count of this node,<sup>3</sup>  $\beta_j$  is defined as follows.

$$\beta_j = \max\left\{\sum_{e \in p} \alpha_e : \forall p \in \mathcal{P}_j\right\} \quad (3)$$

Here,  $\mathcal{P}_j$  is the set of all paths from node (exclusive)  $j$  to a leaf, and  $p$  is such a path. At leaf-node, we do not need to count  $\beta$  (thus it is omitted in our implementation), and  $t_{min} = t_{max} = t$  where  $t$  is the time-stamp of the element kept there. It can be immediately verified that if  $E_L$  and  $E_R$  are the two children of node  $j$ , then

$$\beta_j = \max\{(\alpha_{E_L} + \beta_{E_L}), (\alpha_{E_R} + \beta_{E_R})\} \quad (4)$$

Note that the dominance relationship of “ $e$  dominating  $e'$ ” is captured either at  $e'$  or at an ancestor of  $e'$ .

In Figure 6, we assume that the dAs-tree is built by the bulk-loading technique [12] once  $e_8$  comes and all dominance counts have been done on each node. Note that  $e_6$  dominates both  $e_4$  and  $e_5$  as  $e_6$  comes later with smaller hashed values. In this case, we could record either  $\alpha = 1$  at node  $E_5$  or  $\alpha = 1$  at both  $e_4$  and  $e_5$ . Later option is used in Figure 6.

A dAs-tree is *balanced* if it satisfies the criteria for an AVL-tree. In the next section, we will show how a dAs can be rebalanced by applying standard rebalancing technique in [3].

<sup>3</sup>A dominance count at a node  $e$  is the number of other elements captured that dominate all elements in the subtree with root  $e$ .

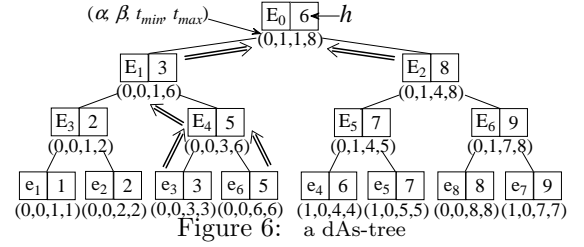


Figure 6: a dAs-tree

#### 4.4.2 Algorithm

Our algorithm to continuously maintaining  $SK(s_i)$  is outlined in Algorithm 4. Below we describe the 3 steps in Algorithm 4 in details.

##### Algorithm 4 Continuously Maintaining $k$ -Skyband

**Input:**  $k, l$  hash functions  $\{h_j : 1 \leq j \leq l\}$ ,  
dAs-trees  $eV_j$  for  $1 \leq j \leq l$ ;

**Output:**  $SK(s_j)$  for  $1 \leq j \leq l$ ;

**Description:**

- 1: **for** a new  $(x, t_x)$  **do**
- 2: **for**  $i = 1$  **to**  $l$  **do**
- 3: **STEP 1.** Update dominance counts in  $eV_i$  against  $(x, h_i(x), t_x)$ ;
- 4: **STEP 2.** (Possibly) delete the nodes not in  $SK(s_i)$  from  $eV_i$  and  $eT_i$ ;
- 5: **STEP 3.** Insert  $(x, h(x), t_x)$  into  $eV_i$  and  $eT_i$ ;

**Step 1: Update Dominance Counts.** There are two cases once  $(x, h_i(x), t)$  is obtained.

Case 1. There is no element in the current  $SK(s_i)$  with the same hashed value as  $(x, h_i(x), t)$ .

Case 2. There is an element in the current  $SK(s_i)$  with the same hashed value as  $(x, h_i(x), t)$ .

*Case 1.* Regarding the example in Figure 5(a), no element dominates  $e_9$  as  $e_9$  is the most recent element; nevertheless,  $e_9$  dominates the elements from  $e_4$  to  $e_8$ . In this example, we need to increase the  $\alpha$  at  $e_6$  by 1. In addition, the dominance counts of  $e_4$ ,  $e_5$ ,  $e_7$ , and  $e_8$  can be “globally” updated at  $E_2$  by increasing its  $\alpha$  by 1 without having to visit any of its descendants to save computation costs. We can immediately verify that at each node  $i$ ,

**Max-Count.**  $(\beta_i + \alpha_i + \sum_{j \in \pi_i} \alpha_j)$  is the maximum total dominance count among elements in its subtree, where  $\pi_i$  is the set of ancestors of node  $i$ .

This, together with (3), require that in our algorithm, any update of the dominant count  $\alpha$  at a node  $e$  has to be propagated to its root by updating all  $\beta$  values, by using (4), on the path. We describe our algorithm below in Algorithm 5.

##### Algorithm 5 UpdatedAs ( $eV_i, (x, h_i(x), t_x)$ )

**Input:**  $eV_i$  and  $(x, h_i(x), t_x)$ ;

**Output:** Updated  $eV_i$ ;

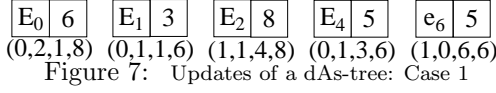
**Description:**

- 1: get the root  $E$  of  $eV_i$ ;
- 2: **if**  $E$  is a leaf **then**
- 3: **if**  $h(x) < E.h$  **then**  $E.\alpha := E.\alpha + 1$
- 4: **else**
- 5: **if**  $h(x) < E.h$  **then**
- 6:  $E2.\alpha := E2.\alpha + 1$ ; ( $E2$  is the right-child)
- 7: UpdatedAs (dAs. $E1, (x, h_i(x), t_x)$ ); ( $E1$  is the left-child)
- 8: **else**
- 9: UpdatedAs (dAs. $E2, (x, h_i(x), t_x)$ );
- 10:  $E.\beta := \max\{E1.\alpha + E1.\beta, E2.\alpha + E2.\beta\}$ ;

In Algorithm 5,  $E.h$  denotes the search key value of the tree at node  $E$ ,  $E.\alpha$  and  $E.\beta$  denote the  $\alpha$  and  $\beta$  values, respectively at  $E$ , dAs. $E1$  denotes the subtree with root  $E1$ . It can be immediately verified that the algorithm visit at most two nodes at each level; thus

the algorithm runs in  $O(\log |SK(s_i)|) (= O(\log |eV_i|))$  time per element.

**Example 4** Regarding the example in Figure 5(a), once  $e_9$  arrives, the updates to dominance counts of the tree in Figure 6 follow the arrows as illustrated. In Figure 7, we illustrate the result that only contains the nodes with some changes.



**Case 2.** Regarding the example in Figure 5(b), the hashed value  $h(x)$  of  $e_9$  is the same as that of  $e_6$ . Thus  $e_6$  has to be removed. In this case, if we update the existing dAs-tree in Figure 6 in the same way as Case 1. We over-count the dominance count of  $e_4$  and  $e_5$  as they have already counted by  $e_6$ . To resolve this issue, we only update the dominance count of elements with time-stamp values greater than 6 - the time-stamp of  $e_6$ .

The algorithm has the same traversal paradigm as that in Case 1 (Algorithm 5) except we need to add the constraint - the time-stamps greater than  $t'$  where  $t'$  is the time-stamp of the element with the same hashed value as a new element. We can modify Algorithm 5 as follows.

- For the situation at line 3, we add the constraint  $t' \leq E.t_{min}$ .
- For the situation at line 5, we also add the condition  $t' \leq E2.t_{min}$ . Then, we do line 7 if another condition  $t' \leq E1.t_{max}$ .
- We change the whole “else part” at line 8 and line 9 to “UpdatedAs (dAs.E1,  $(x, h_i(x), t)$ ) if  $h(x) < E.h$  and  $E1.t_{min} < t' \leq E1.t_{max}$ , UpdatedAs (dAs.E1,  $(x, h_i(x), t)$ ) if  $t' \leq E2.t_{max}$ ”.

Note that after adding these constraints, the algorithm no longer guarantees logarithmic time complexity and runs in  $O(TR)$  where  $TR$  is the size of the tree spanning the nodes visited. Although the algorithm may require linear time for one new element in the worst case, our experiment demonstrates that it is very efficient in practice.

**Example 5** Regarding the example in Figure 5(b), once  $e_9$  arrives, the updates of dominance counts of the tree in Figure 6 are illustrated in Figure 8 where only the nodes with some changes are shown.

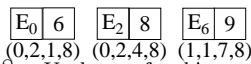


Figure 8: Updates of a dAs-tree: Case 2

**Step 2: Removal Element.** Removing an element from  $eT_i$  for each  $i$  can be done in  $O(\log |eT_i|)$ . Below we mainly focus on removing an element from each  $eV_i$ .

An element needs to be removed if either Case A: its hashed value is the same as that of the new element, or Case B: its total dominance count reaches  $k$  after the new element comes.

We can immediately verify that in both cases, we do not need to discount dominance counts at other elements in  $SK(s_i)$  due to the removal of a new element. This is because for Case A, when we update the dominance counts of other elements for the new element, we already discount the dominance counts caused by  $e$ . For Case B, any element dominated by  $e$  has been already removed in earlier rounds; this is because the total dominance count of  $e$  must be  $k-1$  (for its to reach  $k$ ) before a new element arrives.

Nevertheless, once an element  $e$  removes in both cases we may need to fix the  $\beta$  value of at its parent if the dominance count of  $e$  was larger than its sibling; consequently, it may be propagated to the root. Clearly, this takes  $O(\log |SK(s_i)|)$  time. Moreover, once  $e$  removes from  $eV_i$ , we also remove its parent and connect its sibling to the grand parent to enforce the constraint that every internal node has

two children nodes (noting all elements are kept at leaf level); meanwhile we also pass the  $\alpha$  value at the original parent of  $e$  to  $e$ 's sibling. Next, we may need to re-balance  $eV_i$  from the leaf-level. The AVL-tree balancing technique is to iteratively re-balance the tree from the leaf-part where the node is deleted; it is based on the following 4 cases as depicted in Figure 9 where in each case  $a, b, c, d$  are the subtrees. In each case, while re-balancing the tree, we also need to recalculate  $\alpha$  and  $\beta$ . Specifically, we pass the  $\alpha$  values of  $x, y$ , and  $z$  to their decedents  $a, b, c$ , and  $d$  respectively. Then after rebalancing the tree, we recalculate  $\beta$  values at  $x, y, z$  from  $a, b$ , and  $c$  in a bottom-up fashion, while  $\alpha$  value at  $x, y$ , and  $z$  are assigned to zero. Clearly, this takes constant time.

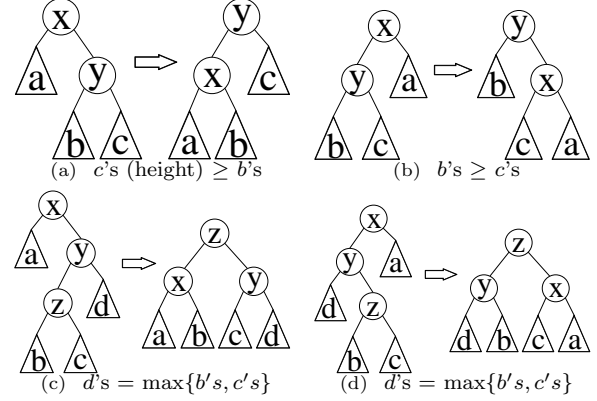


Figure 9: Rebalance

**Example 6** Regarding Case (d) in Figure 9, we give the  $\alpha$  value at  $x$  to the roots of trees  $a, b, c$ , and  $d$ , respectively. Pass the  $\alpha$  value at  $y$  to the roots of  $b, c$ , and  $d$ , and pass the  $\alpha$  to the roots of  $b$  and  $c$ .

After rebalancing, we calculate the  $\beta$  value at  $x$ , the  $\beta$  value at  $y$ , and the  $\beta$  at  $z$  iteratively by using (4).

Regarding the example in Figures 5 and 6, after removing the elements  $e_4, e_5$ , and  $e_7$ , the dAs-tree has been updated to the one as depicted in Figure 10 by the rebalancing technique.

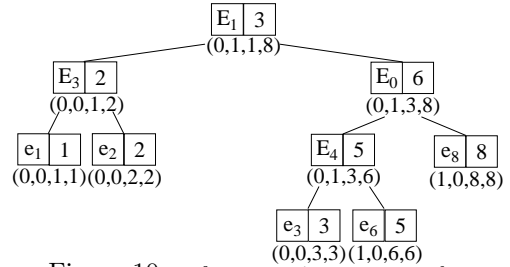
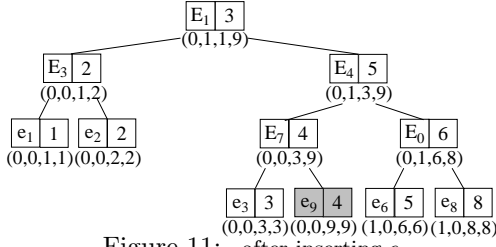


Figure 10: after removing  $e_4, e_5$ , and  $e_7$

In our implementation, we combine the update of dominating counts due to the removal of  $e$  with the rebalancing by one-pass bottom-up. The total time-complexity is  $O(\log |SK(s_i)|)$ . Finally, it should be clear that if the total dominance count at an element (leaf) is  $k$ , then at the root  $\alpha + \beta = k$  according to Max-Count property in the last section. Then, iteratively using the Max-Count property from the root we can reach an a leaf with the total dominance count  $k$ ; this also runs in time  $O(\log |SK(s_i)|)$ .

**Step 3: Insert an element.** Clearly, inserting a new element into each  $eT_i$  can be done in  $O(\log |SK(s_i)|)$ . We focus on the corresponding update of each  $eV_i$ .

Recall that a new element  $e$  is not dominated by any existing elements. Therefore, after inserting  $e$  into  $eV_i$ , the  $\alpha$  values at the ancestors of  $e$  should be 0. On the other hand, the original  $\alpha$  values contributed to computing the maximum dominance count of the elements rooted at those ancestors (i.e. Max-Count property in Section 4.4.2); thus we iteratively update the  $\beta$  values to enforce the Max-Count property on the path  $p$  from the root to  $e$  while changing

Figure 11: after inserting  $e_9$ 

$\alpha$  to 0. We also need to pass the  $\alpha$  of each node in  $p$  to its child which is not on  $p$ . Then, we use the above rebalancing technique to balancing  $eV_i$ , if necessarily. Clearly, the complexity per element is  $O(\log |SK(s_i)|)$ .

Regarding the example in Figure 10, after inserting 9 the dAs-tree becomes the one in Figure 11.

## 5 Performance evaluation

We present the evaluation results of a comprehensive performance study. As mentioned earlier, the techniques in [8, 13] are the only 2 existing techniques which may be immediately applied to counting distinct elements against sliding windows. Nevertheless, the sketch technique in [8] requires a pre-fixed sample space  $\Omega(\frac{1}{\epsilon^2} \log \frac{1}{\delta} \sqrt{m})$ , and the technique requires space  $O(N \frac{1}{\epsilon^2} \log \frac{1}{\delta} \log m)$  in the worst case, to achieve  $\epsilon$ -approximation with  $(1 - \delta)$  confidence. In fact, our experiment shows that for the datasets used below in our performance study, an application of the algorithm in [13] into our problem always requires tens of times more than the dataset size, while the technique in [8] requires at least tens of times more space than our technique. Therefore, we only present the performance evaluation of our techniques presented in this paper. We summarize them below.

**SE-FM** Algorithm 1: the space-efficient sketch construction algorithm in Section 3.1.

**SE-PCSA** The sketch construction technique in Section 3.3.

**k-SKB** Algorithm k-Skyband in Section 4: sketch construction techniques.

We evaluate their space and time efficiency, as well as accuracy in terms of the relative errors; that is,  $\frac{|A_{s,t} - n_{s,t}|}{n_{s,t}}$ . The corresponding query algorithms are also implemented.

In our experiments, two synthetic datasets are generated, *Random* and *Zipf*. In a *Random* dataset, time-stamps of data elements are randomly generated following a uniform distribution, while time-stamps in a *Zipf* dataset follow a Zipf distribution. We assume that data elements arrive according to their time-stamps. We use *duplication ratio*,  $\alpha = \frac{N-n}{N}$ , to control the total number of duplicated objects, where  $n$  is the number of distinct objects and  $N$  is the total number of data elements. For each synthetic dataset, we first generate  $n$  distinct objects, then each object pairs one of the  $N$  time-stamps randomly according to a uniform model. Each of the remaining  $(N - n)$  time-stamps randomly pair one of the  $n$  distinct objects to a uniform model.

The following real dataset *WCH* (World Cup 98's HTTP request data) is used in our performance study. It is downloaded from the Internet Traffic Achieve [9] and consists of 20 million records of requests made to the 1998 World Cup Web site on June 10, 1998. Each record contains time-stamp, clientID, URLID, serverID, and package size (PSIZE). In the dataset, we treat  $\langle \text{clientID}, \text{URLID}, \text{serverID}, \text{PSIZE} \rangle$  as an object. In the dataset, we found there are totally more than 1.97M duplicated data objects and the maximum duplication number of an object is 566.

All algorithms are implemented by C++ and the experiments have been carried out on a PC with Intel P4 2.8GHz CPU and 1G memory under the operation system - Debian Linux. Table 1 below lists the parameters that potentially have an impact on our performance study. In our experiments, all parameters use default values unless otherwise specified.

Notation	Definition (Default Values)
$d$	Dataset Model (Random)
$N$ (syn. data)	Dataset Size (10M)
$\alpha$ (syn. data)	Duplication Ratio (0.2)
$\zeta$ (SE-PCSA)	Number of times to hash an item (100)
$\epsilon$	Guaranteed Precision (0.02)
$1 - \delta$	Confidence (0.95)

Table 1: System Parameters

We adopt the same constant factor 2 for SE-FM, SE-PCSA, and k-SKB. In SE-FM and SE-PCSA, we choose  $k = 32$ ; we also choose  $L = \frac{1}{\epsilon}$ . In k-SKB, we choose  $L = k$ ,  $l = \log \frac{1}{\delta}$ .

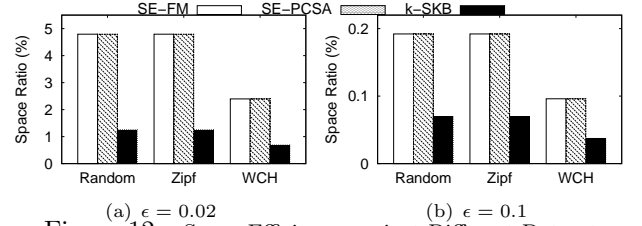
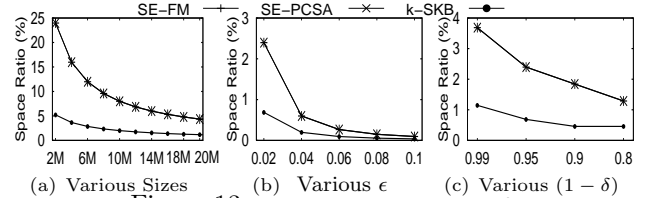


Figure 12: Space Efficiency against Different Dataset

### 5.1 Space Efficiency

We record the maximal size (i.e. the maximal number of elements) of sketch, by each algorithm, during the continuous processing of a dataset. The ratio of such sketch size to the total number of elements processed is called *space ratio*. We study possible impacts from dataset models, dataset sizes,  $\epsilon$  and  $(1 - \delta)$ . Note that the space requirement in SE-FM and SE-PCSA are the same and fixed for given  $m$ ,  $\epsilon$ , and  $\delta$ , while the space required in k-SKB is “opportunistic”.

Figure 13: Impact of Sizes,  $\epsilon$ , and  $\delta$ 

In the first experiment, Figure 12 demonstrates that k-SKB requires the smallest space. The second experiment evaluates the possible impacts from data sizes,  $\epsilon$ , and  $\delta$ . The evaluation results against the real dataset (*WCH*) are presented in Figure 13 where the experiments regarding Figures 13(b) and 13(c) are against the whole dataset. Again, they demonstrate that k-SKB requires the smallest space.

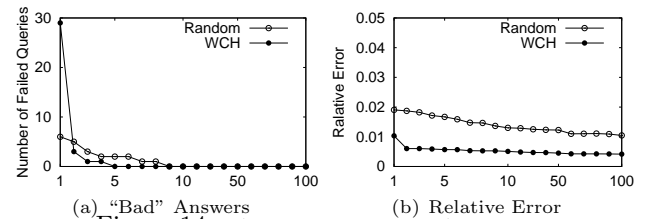


Figure 14: Accuracy of SE-PCSA Variants

### 5.2 Evaluating Accuracy

The results of the first experiment, against the dataset *Random* and *WCH*, are reported in Figure 14. We study an impact of different values of  $\zeta$  (i.e., the number of subsketches an element will be hashed in

SE-PCSA).<sup>4</sup> As demonstrated by Figure 14(a), when  $\zeta = 100$  the number of query results exceeding the relative error guarantee is 0, and an improvements of relative errors becomes less significant after  $\zeta \geq 100$ .

The second experiment is conducted against the 3 different datasets and is reported in Figure 15. It shows that SE-FM provides the highest accuracy, while k-SKB is the second. The numbers (0 or 1) above those “bar figures” are the numbers of answers exceeding their corresponding probabilistic (with confidence at least 0.95) relative error guarantees.

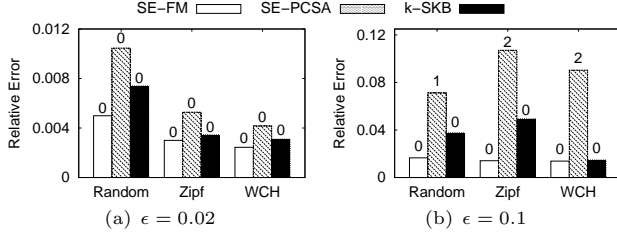


Figure 15: Accuracy against Different Dataset

The third experiment evaluates possible impacts from data sizes,  $\epsilon$ , and  $(1 - \delta)$ . The experiment is conducted against real dataset WCH and is reported in Figure 16. It shows that SE-FM always provides the highest accuracy and k-SKB is the second accurate. We also report that all answers obtained against the sketches by SE-FM or k-SKB satisfy the corresponding probabilistic error guarantees while SE-PCSA leads to 8 answers exceeding a designated relative error guarantee  $\epsilon$  for the setting -  $\epsilon = 0.02$  and  $(1 - \delta) = 0.8$ .

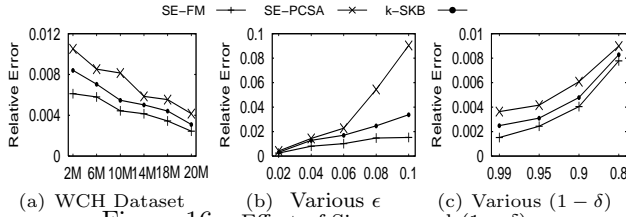


Figure 16: Effect of Sizes,  $\epsilon$ , and  $(1 - \delta)$

### 5.3 Time Efficiency

The cost of processing one data element may be too small to be recorded accurately (especially for SE-PCSA and k-SKB), we record the average time for processing every batch of 1K elements as *the delay of one element*. In addition, we also record the maximum value of such delay per data element time as *the maximal delay of each element*.

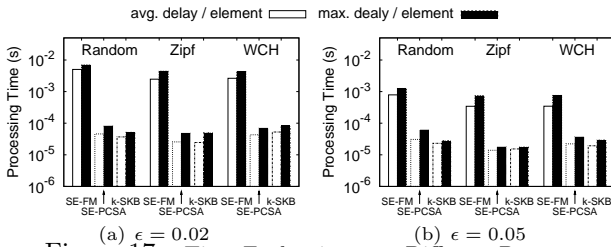


Figure 17: Time Evaluation over Different Datasets

The first experiment is conducted against the 3 datasets Random, Zipf, WCH. The experiment results are reported in Figure 17. They indicate that SE-FM can only process a medium speed data stream online - 200-400 elements per second when  $\epsilon = 0.02$  and about 2500 elements per second when  $\epsilon = 0.05$ . However, both k-SKB and SE-PCSA can process high speed data streams. They can process at least 20,000 data elements per second even with  $\epsilon = 0.02$ . We also tested the naive technique to maintain each  $SK(s_i)$  regarding BJKST-based sketches; that is, we update the total dominance count of each data element once a new element comes (we first find the left-most ele-

ment with hashed value smaller than that of the new element and then do a linear scan from the element). Our experiment shows that the naive algorithm can only process about 100 elements per seconds; thus it can support very slow data streams only.

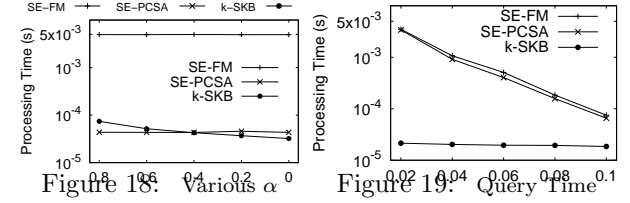


Figure 18: Processing Time (s) vs. Query Time

In the second experiment, Figure 18 shows that that our techniques are not very sensitive to different duplication ratios.

Finally, we evaluate the 3 query processing algorithms against the real dataset WCH. 1K queries are randomly generated as before. We vary  $\epsilon$  from 0.02 to 0.1 and other parameters adopt default values. The average response time of the 1K queries for each algorithm, is reported in Figure 19, respectively. As expected, SE-FM and SE-PCSA have the similar performance as they use the same sketch structure for query; both of them require much more time than k-SKB does, especially when  $\epsilon$  is small.

## 6 Conclusions

In this paper, we investigated the problem of approximately counting distinct elements against sliding windows (DCSW). Novel space and time efficient techniques are developed for continuously maintaining sketches so that a DCSW can be processed with the guarantee of  $\epsilon$ -approximation. This is the first work providing the space and time efficient data stream techniques to approximately counting distinct objects over sliding windows. The space required by our techniques is near optimal. Besides proven accuracy and space guarantees, our algorithms are also efficient enough to support on-line computation of very high speed data streams with an element arrival rate up to 20K/second.

**Acknowledgement.** The work was supported by ARC Grant (DP0987557, DP0881035 and DP0666428) and Google Research Award.

## References

- [1] Z. Bar-Yossef, T. S. Jayram, R. Kumar, D. Sivakumar, and L. Trevisan. Counting distinct elements in a data stream. In *RANDOM'02*.
- [2] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *ICDE'04*.
- [3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. 2001.
- [4] G. Cormode, S. Muthukrishnan, and W. Zhuang. What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *ICDE'06*.
- [5] M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. In *SODA'03*.
- [6] W. Feller. *An Introduction to Probability Theory and Its Applications*. John Wiley & Sons, Inc., 1966.
- [7] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985.
- [8] P. B. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and event reports. In *VLDB*, 2001.
- [9] Internet Traffic Archive. <http://ita.ee.lbl.gov>.
- [10] D. Papadias, Y. Tao, R. Kalnis, and J. Zhang. Indexing spatio-temporal data warehouses. In *ICDE*, 2002.
- [11] D. Papadias, Y. Tao, and F. Seeger. Progressive skyline computation in database systems. *TODS*, 30(1):41–82, 2005.
- [12] R. Ramakrishnan and J. Gehrke. *Database Management Systems, Second Edition*. 2000.
- [13] Y. Tao, G. Kollios, J. Considine, F. Li, and D. Papadias. Spatio-temporal aggregation using sketches. In *ICDE 2004*.

<sup>4</sup>In SE-PCSA, different values of  $\zeta$  will not make any difference in space requirement if the other parameters are the same.



# Privacy-Aware Access Control in XML Databases

Anders H. Landberg

J. Wenny Rahayu

Eric Pardede

Department of Computer Science & Computer Engineering  
La Trobe University, Melbourne, Australia

Email: a.landberg@latrobe.edu.au, w.rahayu@latrobe.edu.au, e.pardede@latrobe.edu.au

## Abstract

With the growing use of XML for data transfer and data storage across the web, securing XML documents has become an important issue. The XML privacy and data access control issues are especially significant in XML data repositories because they typically store large collections of highly sensitive business data, health information, etc. Protecting privacy by only restricting access to individual nodes in the XML document tree is not sufficient, as combinations of nodes and aggregations thereof can lead to disclosure of sensitive information. Moreover, a mechanism is required to cope with such combined data privacy levels, as they must be validated on query-time. Extending from XML access control models, this paper proposes a privacy-aware access control model for XML with composite security levels, which adds a further level of fine-granularity to existing approaches. In order to enforce these composite security levels, we then introduce a methodology based on path-triggers. Finally, we evaluate the performance of our new approach using three different implementation techniques.

## 1 Introduction

Privacy-aware access control is an important issue when storing and publishing sensitive information, such as financial and health data. We define privacy-aware access control as an access control model, which aims at securing data from access that may lead to privacy violation. So far, most existing work in the area have been focusing on security access level only. The most influential works have been proposed by Damiani [5][6][7] and Bertino [2], and many other approaches have been inspired by their work [4][12][9][13][8]. These works are based on static access rule definitions, and are not targeting privacy issues in connection with access control. The fundamental concept of these approaches is to define constraints on XML elements, attributes, and links, and thus enabling only authorised users or user groups with access to those data. These constraints are specified as XPath expressions, and are associated with levels of access. However, while static access control can secure a database from unauthorised access by specifying access rules, it does not guarantee it from privacy violation, which is caused by accessing combinations of nodes, and occurs at query time.

Example: Given an XML document instance (see Figure 1) and a schema definition that describes the document, we can apply access constraints to each and every node. Treatments and personal information should have a higher level of access than the ward. Personal information should have a higher access level than treatments, because a composition of its child nodes 'ZIP', 'Age', and 'Gender', reveal more detailed information about a patient and can easier lead to disclosure of the data.

When restricting access to data, it is also important to consider how it can be queried and how the access control mechanism functions. Further, it must be ensured that after the application of a privacy-aware access control model, users such as analysts or researchers are still able to retrieve useful information from the data. This is an issue because by restricting access to particular nodes, the query result may be useless.

While the existing solution is acceptable for single node access, it fails when restricting access to composition nodes. The reason for this is because these access control policies are defined on design-time. However, combined node access restrictions cannot be predetermined, because they depend on the query. Thus, these composition node privacy constraints must be validated on query-time.

### 1.1 Defects of static(single) privacy constraints

In a publicly accessible hospital XML data repository, where the specific identifiers (such as name) of patients have been removed, a combination of two or more identifier attributes may lead to the disclosure of a particular patient. For example (see Figure 1), an adversary knows that Mr. President had treatments on the 1st of June 2007, but he does not know which treatments were conducted. The adversary has access to those parts of the document with a privacy level of 3 or less (see Table 1). After querying the hospital's public domain statistics on 'Treatments' and 'Date', he finds that only one male patient received treatment on this date. The treatment being dialysis, the adversary now identifies Mr. President as the patient that had dialysis on the 1st of June 2007.

One way to overcome this problem using existing access control models, is to increase the level of access for nodes that will lead to disclosure when combined, or to completely restrict access to one of the nodes. In our example, these nodes are 'Treatments' and 'Date'. In this way it can be ensured that an adversary will fail to re-identify Mr. President, because one of the two nodes is not available for querying. However, this has created a new problem. Let us assume that we previously increased the access level of the 'Date' node, so that it was no longer accessible to the adversary. As a result of this, any other data anal-

Table 1: Access authorization definitions

Node	Level of access
/PatientRecords/PatientRecord/Date/	1
/PatientRecords/PatientRecord/Treatments/	2
/PatientRecords/PatientRecord/Ward/	2
/PatientRecords/PatientRecord/Personal/	3
/PatientRecords/PatientRecord/PreviousRecords/	1

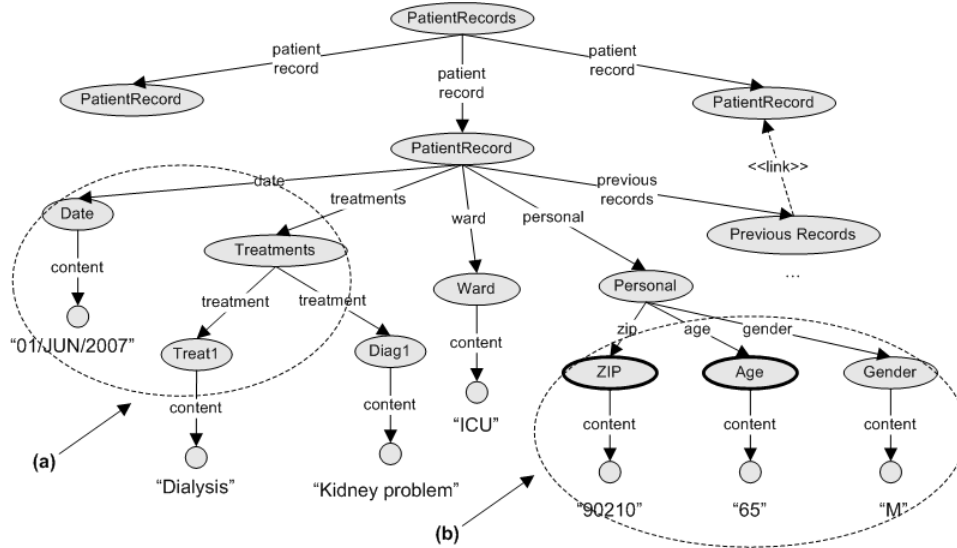


Figure 1: XML Document Instance: Patient Record

ysis queries issued by any user will not have access to the date node either, hence making the queries nearly useless (see Table 2). The same dilemma arises if we choose to restrict access to the 'Treatments' node, or some of its child-nodes.

Table 2: Inaccessible data. Access levels too high.

Node	Access Level
/../PatientRecord/Date/	5
/../PatientRecord/Treatments/	2
/../PatientRecord/Ward/	2
/../PatientRecord/Personal/	3
/../PatientRecord/PreviousRecords/	1

Table 3: Unnecessarily restricted data.

Node	Access Level
/../PatientRecord/	7
/../PatientRecord/Date/	(7)
/../PatientRecord/Treatments/	(7)
/../PatientRecord/Ward/	(7)
/../PatientRecord/Personal/	(7)

Another work-around solution to overcome this problem using existing approaches, is to identify the parent node of the sensitive nodes 'Date' and 'Treatments', and increase its level of access. An example of this scenario is given in Table 3, where the 'PatientRecord' node is given a very large access level, hence restricting access to any of its child nodes that lead to disclosure.

While this solution achieves sufficient privacy protection by restricting access to 'Date' and 'Treatments' nodes at the same time, it brings the side ef-

fect of also restricting access to these nodes individually and to all other nodes below the 'PatientRecord' node. In other words, a simple query on the 'Ward' node is not possible any longer, although this query cannot lead to privacy disclosure. Hence, this solution suffers unnecessarily restricted data and limited use.

This issue can not be avoided using existing access control models, as it is subject to the combination of nodes that is queried on run-time. Such constraints can not be pre-defined by approaches found in existing literature because they must be checked and evaluated when the query is executed. The reason behind this is that we cannot foresee which queries the user will issue, nor do we know the users' intentions.

Referring back to the access levels that we defined in Table 1, we can also see that some users (with access level less than 3) can not access the personal data such as 'Age' and 'Gender', and thus, making data analysis almost useless (see (b) in Figure 1). As far as existing works are concerned, we can only attempt to remedy the problem by specifying individual access levels to each child node of the personal node. However, this does not solve the problem of restricting composition node access.

## 1.2 Contributions

To overcome the defects of previous approaches, we propose a novel privacy-aware access control model for XML that captures query-time access control for combined access to nodes. The goal of security level composition is to group nodes in the XML data (see Table 4), which when accessed in combination, may lead to disclosure of individual patient records, and further to the re-identification of individual patients. While the individual access levels for the nodes within a combined security level can remain unchanged, a higher level of access will be required when attempt-

ing to access the nodes together. This means that the information in the data can be protected much stronger, yet granting less restrictive access to individual nodes, which is crucial when conducting data analysis by querying.

It is a well-known problem in the area of data privacy that a combination of attributes can lead to disclosure of sensitive information [15]. Determining the combinations of data that lead to privacy disclosure is a research field in itself and goes beyond the scope of this paper. We do not offer a new solution to arrange the data in a privacy preserving way, but propose concepts and methods to enforce these principles.

By introducing the notion of *composite security levels CSL*, our model provides a tool to define privacy-aware access constraints over sets of sensitive elements on a schema definition, or directly within the XML data itself, where a security level can be attached to a set of aggregated values or an instance of XML data. Secondly, we propose a new mechanism to implement the combined privacy-aware access control model with the use of XML triggers. We show that combined privacy-aware access constraints can be maintained in a cost-efficient manner by using this new privacy preserving access control mechanism.

The rest of the paper is organised as follows. Section 2 summarises related work in the area of access control for XML, section 3 formalises our methodology, section 4 describes how our model is enforced with the use of XML triggers, and section 5 presents the analysis of our approach. Finally, section 6 concludes the paper.

## 2 Related Work

In the area of access control for XML, an early work is represented by Samarati et al. [14] that proposes an access control model for HTML documents. This approach is clearly limited by the semantic deficits of HTML as opposed to XML and as such, the model by Samarati et al. is limited in many ways when defining access policies as well as partitioning a document into meaningful segments for an access control approach.

Damiani et al. [6] propose a method to define user groups and access authorisations in XML documents. User groups can be granted different types of access (such as read, write) to certain parts of the document by defining access authorisations. These access authorisations are described based on subjects and objects. Subjects are accessing entities such as users, and objects are path expressions that identify parts in an XML document.

Damiani's paper offers a detailed approach to define data access control in XML documents. However, it is based on the existence of a schema (DTD), and cannot be applied to (schema-less) XML document instances themselves. The access policies are restrictive (either "+" or "-" for access or no access), and there is no differentiation between different levels of confidentiality of the data. By contrast, our model allows for both data-level and schema-level definition of security levels. Also, our concept of combined security levels is a novel approach in this area.

Bertino et al. [2][3] propose an XML data access model by introducing various levels of protection granularity, such as document/DTD, set of documents, (sub)elements, attributes, and links. By applying these protection levels to policies, a fine-grained and detailed access control is now possible. The authors also focus on different levels of well-formedness for XML documents, and make suggestions on which policies to apply in which case.

Our paper can be differentiated from [2] and [3] in the following areas. First, their access control poli-

cies are defined 'outside' the document instances. Although the authors claim to define access policies for document instances, this approach is ultimately still schema-based. This means that if these definitions are lost or corrupted, then the data will be unsecured and vulnerable to misuse. Our model in contrast can be applied to a schema, as well as defined in the data itself, yet maintains *propagation* (inheritance) of security levels throughout the document structure. Second, there is no mentioning about the existence of combined security levels or similar phenomena.

Kuper et al. [10] use views over XML data to restrict access. It is important to point out the significant difference between this approach and ours. By creating (static) views of the XML data, Kuper et al. hard-code the security model, instead of (dynamically) validating it on query-time. This means that after changes in the data, the views possibly need to be updated for the security model to remain valid. On the other side, our model is immune to such changes, and any security levels and composite security levels still apply even after data modification.

A large number of works in the area of XML access control and privacy have been proposed that are inspired by the works of Damiani and Bertino [4][12][9][13][8]. The major rationale behind all approaches, however, is to bind access or privacy policies with particular nodes and/or attributes, which we will refer to as single, or static privacy levels. The reason for this is because the definition of access control and privacy protection policies and levels takes place prior to the querying of the data, hence static, and cannot be dynamically modified on query-time.

The major difference between previous works and ours is that we focus our privacy model on privacy control validation of composition nodes that is performed at run-time, rather than a static privacy definition scheme. Our model can be implemented in the XML data, and also for schema.

## 3 Proposed Method

This section formalises the proposed security model by focusing on the process of expanding and labelling nodes in an XML document tree. It explains how the hierarchical structure influences the security constraints throughout the document tree, and provides a methodology on how the security constraints can be implemented. We use Figure 2 to illustrate the new concepts and methodology.

### 3.1 Definitions and Rules

**Definition 1.** A *security level SL* is defined as  $SL(entity) = value$ , where  $entity \in \{user, system, node\}$ , and *value* specifies the security level where  $1 \leq value \leq max$ .

**Property 1.** Unless otherwise defined, the security level *SL* along a *path*  $P = N_1/N_2/.../N_m$  must be incremental or equal, such that  $SL(N_i) \leq SL(N_{i+1})$  where  $1 \leq i \leq m-1$ .

For the following definitions and examples, we will use entity instances *E1*, *E2*, and *User1* with the properties  $E1 \in \{user, system\}$ ,  $E2 \in \{node\}$ ,  $User1 \in \{user, system\}$ ,  $SL(E1) = 6$ ,  $SL(E2) = 4$ , and  $SL(User1) = 3$ .

**Example 1.** Given an XML schema as shown in Figure 2,  $SL(User1) = 3$  defines a security level of *value* = 3 for entity *User1*.  $SL(PatientRecords) = 1$  defines security level of *value* = 1 for entity *PatientRecords*, which in this case is the *PatientRecords*

Table 4: Privacy protected. Data accessible.

Node	Single SL	Composite SL
/PatientRecords/PatientRecord/Date/	1	5
/PatientRecords/PatientRecord/Treatments/	2	
/PatientRecords/PatientRecord/Ward/	2	
/PatientRecords/PatientRecord/Personal/	3	
/PatientRecords/PatientRecord/Personal/ZIP/	(3)	7
/PatientRecords/PatientRecord/Personal/Age/	(3)	
/PatientRecords/PatientRecord/Personal/Gender/	(3)	
/PatientRecords/PatientRecord/PreviousRecords/	1	

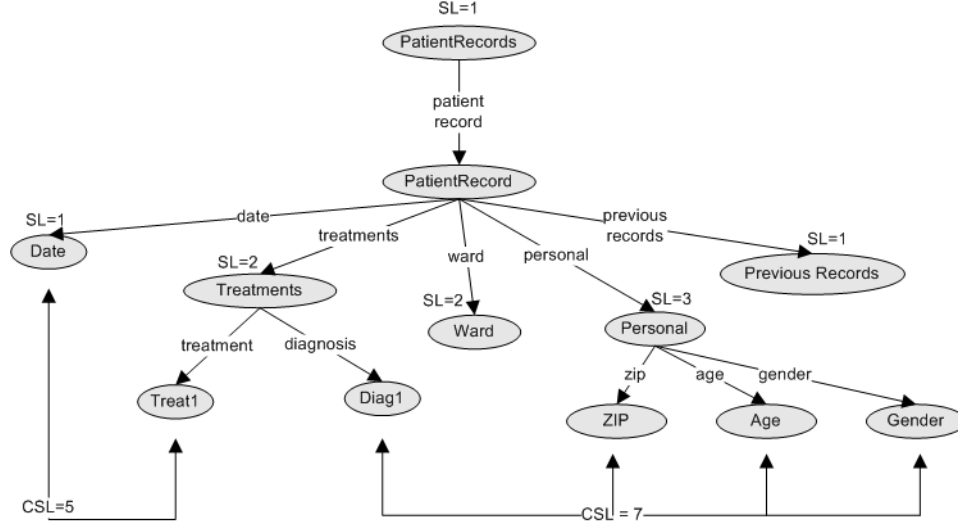


Figure 2: Privacy model applied on XML schema

node.

**Rule 1.1** If a node does not specify  $SL$ , then its  $SL$  is inherited from its nearest ancestor that specifies  $SL$ .

**Rule 1.2** Security level grouping: if  $SL$  is equal for all child nodes of a parent node, then this  $SL$  applies to the parent node.

**Definition 2.** Two entities  $E1$  and  $E2$  are defined as matching, when  $SL(E1) \geq SL(E2)$ , where  $E1 \in \{user, system\}$ , and  $E2 \in \{node\}$ .

**Example 2.** Given an XML schema as shown in Figure 2, and given  $SL(User1) = 3$ , and given  $SL(Personal)=3$ ,  $SL(User1)$  and  $SL(Personal)$  are said to be matching because  $SL(User1) \geq SL(Personal)$ , and  $User1 \in \{user, system\}$ , and  $Personal \in \{node\}$ .

**Definition 3.** A composite security level  $CSL$  is defined as a tuple  $(CN, SL)$ , where  $CN$  is a set of composite nodes and a  $SL$  is a security level.  $CSL = \{CN, SL(entity) | \forall N \in CN : SL(N) \leq SL(entity)\}$ .

**Example 3.** The  $SL$  of a  $CSL$  applies to all nodes in  $CN$  if all nodes in  $CN$  are accessed together.

**Definition 4.** Given a  $CSL(CN, SL(entity))$ , and given an entity  $E1$ ,  $E1$  and  $CSL(CN, SL)$  are defined as matching, when  $\forall N \in CN : SL(E1) \geq SL(N)$ , and  $SL(E1) \geq SL(entity)$ , where  $E1 \in \{user, system\}$ .

**Example 4.** Given an XML schema as shown in Figure 2, and given a  $CSL(CN, SL(entity))$

where  $CN = \{ 'ZIP', 'Age', 'Gender', 'Diag1' \}$ , and  $SL(entity)=7$ , and given  $E1$  where  $SL(E1)=7$ ,  $E1$  is said to be matching with  $CSL(CN, SL)$  because  $SL(E1) \geq SL(entity)$  and  $\forall N \in CN : SL(E1) \geq SL(N)$ .

The ancestor nodes of nodes that participate in a  $CSL$  are not affected by an increased level of privacy. This means that the  $SL$  of any ancestor nodes relative to the  $CSL$  participants remains unchanged if a  $CSL$  applies during validation. As for the descendant nodes of  $CSL$  participants, the following rule applies.

**Rule 2.** If a node  $N$  participates in a  $CSL$ , and if this  $CSL$  applies during a data access operation, then all descendants of this node  $N$  inherit the security level of the  $CSL$ . Exception: If a descendant node  $D$  of node  $N$  has a security level that is greater or equal to the security level of the  $CSL$ , then the security level of node  $D$  remains unchanged, and is treated according to Definition 2.

The above definitions and rules have described our privacy model. In the following section, Algorithm 1 represents the access control mechanism. It is used to detect security levels in the XML data, and to match these against the accessing entity's security level.

**Algorithm.** The algorithm first validates basic  $SL$ 's (security levels), and in the same process, identifies and validates composite security levels. The  $CSL$  validation is divided into three parts: (i) identification of  $CSL$ id's ( $CSL$  tags), (ii) identification of participating nodes for these  $CSL$  id's, and (iii) the actual matching of the  $CSL$ 's security level (if the  $CSL$  applies) against the accessing entity's security level.

Function `query_node_instances(D, path_expr)` re-

**Algorithm 1:** Matching Algorithm

---

**Data:**  
 $D$  : XML document  
selected\_nodes: Set of selected node types as path expressions  
 $SL(\text{accessing\_entity})$ : accessing entity's security level  
**Result:**  $b$ : returns true if access is granted, false otherwise

**begin**  
  **foreach**  $path\_expr$  **in**  $SN$  **do**  
     $node\_list \leftarrow query\_node\_instances(D, path\_expr)$   
    **foreach**  $node$  **in**  $node\_list$  **do**  
      **if**  $node.SL > SL(\text{accessing\_entity})$  **then**  
        | return false  
      **else**  
        | continue  
      **if**  $has\_attribute(node, "CSLid")$  **and** **not**  $node.CSLid \in csl\_list$  **then**  
         $part\_nodes \leftarrow query\_dct\_node\_typ(D, all\_paths, "CSLid")$   
        **if**  $selected\_nodes \subseteq part\_nodes$  **and**  $node.CSL > SL(\text{accessing\_entity})$  **then**  
          | return false  
        **else**  
          | continue  
      **else**  
        | add( $csl\_list$ ,  $node.CSLid$ )  
    return true  
**end**

---

turns a set of node instances from document  $D$  as specified by path expression  $path\_expr$ . Function  $has\_attribute(node, "CSLid")$  returns true if  $node$  has attribute "CSLid", false otherwise. Function  $query\_dct\_node\_typ(D, all\_paths, "CSLid")$  returns a set of all distinct node types from document  $D$  that have attribute "CSLid". Function  $add(csl\_list, node.CSLid)$  adds attribute value  $node.CSLid$  to list  $csl\_list$ .

In regards to Rule 2, this algorithm is applied recursively to all descendant nodes of a given context node, which is being validated during data access.

#### 4 XML Triggers for Privacy-Aware Access Control

Single and composite security level validation as proposed in the matching algorithm, can efficiently and effectively be realised using XML triggers. We have introduced the notion of XML triggers as a mechanism to maintain XML data consistency after a sequence of updates [11]. In this section, we adopt a triggering mechanism to activate a certain access policy each time a query that may violate privacy rules is executed. We believe that this is an effective and unique way of dynamically firing an access control in the database layer.

##### 4.1 Single security level validation trigger (SSLVT)

A single security level is either hard coded into the document as an attribute 'SL' (document scope), or the security level is defined outside the document with an XPath expression. In the latter case, the security level applies to the particular element in all docu-

ments that are associated with that schema (schema scope).

To create a mechanism for validation of this single security level, we first create a new path-trigger, and apply it to the node (expressed by  $tp$ ) that is to be validated upon access. It requires the trigger's event to be 'access', so whenever there is an access to this node, then the trigger is fired. The event 'access' is not available in [11], therefore we must modify the trigger's event options accordingly.

The target path  $tp$  of a SSLVT is the XPath expression that specifies the node to be protected by the single security level, and has the form  $\{<XPath\ expr>\}$ .

The next step is to adjust the trigger's action-body appropriately. The attribute that stores the security level must now be compared to the accessing entity's security level. The latter must be either hard-coded into the trigger, or retrieved/passed dynamically to the trigger on run-time. Function

Trigger definition:  $SSLVT := \{e, c, a\}$   
 $e := \{\text{access } tp\}$   
 $c := \{SL(\text{entity}) < \$con/@SL\}$  ; for document scope  
 $c := \{SL(\text{entity}) < node\_SL(\$con)\}$  ; for schema scope  
 $a := \{\text{RETURN } \$con/*[@SL \leq SL(\text{entity})]\}$

$node.SL(path\_expr)$  takes as input an XPath expression and returns the security level for it.  $SL(\text{entity})$  specifies the security level of the accessing entity.

Although we only consider read-operations (read-only queries) for our privacy model, it can be clearly seen that the path-trigger mechanism is capable of supporting other operations such as 'insert', 'delete', and 'update' as well. Compared with other approaches [6][2], these operations would be similar to 'write', 'remove', and 'replace' respectively.

**Example.** As an example trigger implementation, we first declare a security level for target node  $tp = ./PatientRecord/Ward$ , and set its  $SL = 1$ . This implies that when we declare the trigger, the event will be  $e = \text{access } tp$ . We must explicitly specify which implementation type we use, so that the correct condition is chosen. In this case we chose schema type. Thus, the condition of the trigger will be  $c := \{SL(\text{entity}) < node\_SL(\$con)\}$ , where  $\$con$  is the XPath expression specifying the context node that is being evaluated by the trigger.

##### 4.2 Composite security levels validation trigger (CSLVT)

If two or more elements participate in a composite security level, then these elements specify two additional attributes, namely the id of the CSL, and the security level of the CSL. These values can either be directly hard-coded into XML data (document scope), or specified in an XML schema definition (schema scope). Alternatively, a CSL can be defined as a set of XPath expressions that specify the participating nodes in the CSL, and a security level (a number) that applies for the CSL.

The strategy for implementing a path-trigger that validates a composite security level is similar to the above described one. The differences are that to validate a CSL, we must first identify the closest common ancestor node (this will be the context node for the trigger [11]), and then implement the trigger's action-body in such a way that all participating nodes are retrieved, and their security level for the CSL is matched against the accessing entity's security level.

The target path  $tp$  of a CSLVT is the path that specifies the closest common ancestor node, relative

to all the participating nodes in  $CN$ , and has the form  $\{<XPath\ expr>\}$ .

The reason why we need to find the closest common ancestor node of all participating nodes, is because from this point in the XML document structure, the number of node traversals to each of the participating nodes, is minimised. Also, the path-trigger will automatically traverse all context paths relative to the context node, and in this way allow direct access to possible participants of CSLs.

```

Trigger definition: CSLVT := {e, c, a}
e := {access tp}
c := true
a := { IF $path IN S(CN) THEN mark($path) END IF;
IF getMarked(CSL id) EQUALS S(CN) THEN
IF SL(entity) < SL(CSL id) THEN
THROW EXCEPTION
END IF;
END IF; }

```

Function `mark(path_expr)` marks a path as belonging to the CSL that is to be validated. Function `getMarked(CSL id)` returns the set of marked nodes for the CSL with *CSL id*.

The most time consuming task in this process is to determine whether a CSL applies or not, e.g. whether all or a subset of the retrieved nodes by a query are all participating in one and the same CSL. To do this, we must first retrieve the set of nodes that participate in the CSL with the respective id that is to be validated. Then, this set of nodes must be compared against the set of participating nodes that are actually retrieved during the query. If the sets match, then the CSL applies, and the security level of CSL and accessing entity can be matched against each other.

**Example.** Referring to Figure 2 and Table 5, this example shall demonstrate a practical application of the CSLTV mechanism. Table 5 lists a number of sample queries that are issued by accessing entities. Each of the entities have an associated security level that is used to determine whether they have access to the XML contents specified by the query.

The 'Public' user profile can access nodes that have an SL below their own (query 1), but when attempting to access nodes on which a CSL applies (query 2), then access will be denied. In this case the selection in the query is made based upon the @SL attribute.

## 5 Analysis

For a more realistic and fair comparison of our method, we conducted the tests in three different modes. These modes are (i) 'Cold-run', (ii) 'View', and (iii) 'Trigger' respectively. In 'Cold-run' mode, security levels and composite security levels are validated on query-time, and are implemented as pre-conditions of the respective queries. In 'View' mode, only composite security levels are validated on run time, single security levels are implemented using views of the XML data. In 'Trigger' mode, XML triggers are used to implement the privacy model. It is obvious that a system without access control will perform better, as access control mechanisms add performance overhead.

### 5.1 Experimental Environment

The tests described in this section were conducted using the above described implementation using XML triggers on a machine equipped with a 2.0GHz Intel

Pentium M 760 processor and 1GB DDR2 memory, running Windows XP professional as operating system. We used the Sigmod-Record in XML format<sup>1</sup> to build the XML repository, with the main document holding a total of 23047 nodes. The XPath queries were conducted in such a way that the CSLs were activated and executed.

We define a *computational cost unit* as the cost of traversing a node in the XML document tree.

### 5.2 Performance Results

The results of our tests in respect to composite security levels can be summarised as follows. Both the cold-run (CR) mode and the view (VW) mode showed a linear increase by 5 cost units per additional CSL that had to be validated. The trigger (TR) mode performed at a linear increase by 3 cost units, and therefore 60% better than the former traditional modes.

As we expected, the computational cost of validating an incremental number of CSLs increases linearly. Maintaining a privacy preservation model will always incur an overhead, but the results show us that the mechanism we use helps to significantly reduce the cost. We can clearly see that the trigger mode outperforms both other modes, since pre-processing of single security levels is performed when the trigger is created, and context paths can be pre-cached by the path-trigger to save computation time on query-time.

Traditional approaches rely on the 'Cold-Run' mode, where all access validation is performed at query-time. Some more recent approaches that use views [10][1] rely on what we call 'View' mode, where a set of access control and privacy constraints are 'hard-coded' into the database or repository as a view. While the latter of these two modes performs slightly better than the cold-run, it still suffers the deficit of being incapable of supporting the composite security levels to a satisfactory degree.

The tests in path-trigger mode achieve much better results, because the trigger creates a dynamic compilation of the participating node paths when it is created. This has the effect that query-time assembly of participating nodes is reduced, and hence computation time can be kept to a minimum. Further computational cost is saved when composite security levels overlap, which will be discussed in the next section. This scenario of overlapping nodes for CSLs is common, because generally a set of quasi-identifying nodes in combination are the reason for privacy disclosure of sensitive nodes. Therefore, CSLs will most likely apply to these sets of quasi-identifiers with one or more additional sensitive nodes.

Table 6: CSL Overlap, n=10

k	max CSL	ovlp	ovlp/CSL (%)
2	45	9	20%
3	120	36	30%
4	210	84	40%
5	252	126	50%
6	210	126	60%
7	120	84	70%
8	45	36	80%
9	10	9	90%
10	1	1	100%

### 5.3 Scalability

The maximum number of composite security levels with a cardinality  $k$  that can be applied to an XML

<sup>1</sup> Available for download at <http://www.sigmod.org/record/xml>

Table 5: Sample XPath queries

Query	Accessing Entity	Access
/PatientRecords/PatientRecord/*[@SL=1]	Public (SL=4)	yes
/PatientRecords/PatientRecord/*[@SL<=3]	Public (SL=4)	no
/PatientRecords/PatientRecord/*[position()<3]	Researcher (SL=6)	yes
/PatientRecords/PatientRecord/*[position()>1]	Researcher (SL=6)	no
/PatientRecords/PatientRecord/	Admin (SL=10)	yes
/PatientRecords/PatientRecord/	Web-Service (SL=4)	no

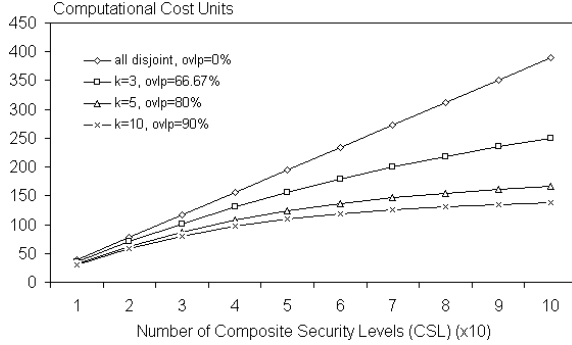


Figure 3: Validating CSLs

structure containing  $n$  leaf nodes is specified by the binomial coefficient  $C_n^k = \binom{n}{k} = \frac{n!}{k!(n-k)!}$ , which is also known as the *choose function*. We apply this function to our method to measure how many CSLs will possibly be generated based on a given XML structure.

Table 6 presents the maximum number of CSLs that can be generated with a given cardinality  $k$ , the number of nodes that overlap for the CSLs, and the percentage of overlapping nodes per CSL. From these results we derive a cost model that represents the percentage of overlap per CSL:  $CSL_{ovlp} = \frac{100}{n}$  with parameter  $n$  that denotes tree cardinality. We calculate the number of overlapping nodes with the formula  $n_{ovlp} = C_{n-1}^{k-1} = \binom{n-1}{k-1} = \frac{(n-1)!}{(k-1)!(n-k)!}$ .

The performance gain is dependent on the cardinality of the grain of the XML document, to which the CSLs are being applied. As an example, if we have a document structure as illustrated in Figure 2, then the grain of the document is a 'PatientRecord' node, and the cardinality  $n$  thereof is 7, given that there is only one 'Treat1' and 'Diag1' node below the 'Treatments' node. The performance gain that can be achieved in this document yields  $ovlp = \frac{100}{7} = 14.3\%$ .

The maximum CSL overlap percentage  $MAX(CSL_{ovlp})$  is dependent on the cardinality of the CSLs, and is calculated using the following formula.  $CSL_{ovlp} = 100 * (1 - \frac{1}{k})$ . An increasing possible overlap of CSLs occurs with increasing variable  $k$  and constant  $n$ . For example, given two CSLs with  $k=2$  each, then the overlap can be at most 50%, and given three CSLs with  $k=3$  each, then  $MAX(CSL_{ovlp}) = 66.67\%$ .

A low number of CSLs indicates a likelihood of the CSLs to be disjoint. With increasing number of CSLs, the likelihood of overlapping among these CSLs increases, and hence brings performance gain (see Figure 3). The first (upper, 0% overlap) graph show performance for the worst-case when there is no overlap (all CSLs disjoint), in which case the performance is a linear graph with a constant increase in computational cost for each additional CSL.

The next three graphs represent performance for CSL cardinalities  $k=3$ ,  $k=5$ , and  $k=10$  respectively.

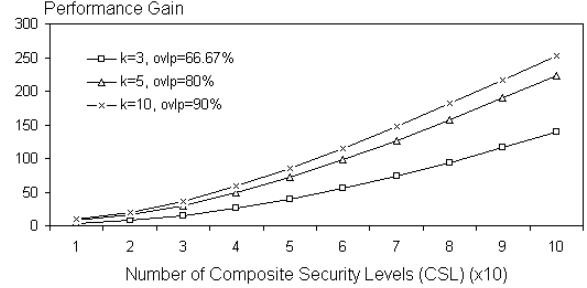


Figure 4: Performance gain

It can be seen that with increasing overlap percentage, the performance stabilises and will become constant once there is a 100% overlap of CSLs.

Figure 4 shows that with moderate CSL cardinalities, a substantial performance gain can be achieved.

### Summary

The performance evaluation revealed that the new proposed concept of composite security levels does not add any substantial overhead to the access control mechanism. This is particularly visible in the comparison of cold-run and view modes against trigger mode. Whereas the first modes only slightly vary in cost, the trigger mode clearly outperforms both other modes. So we can say that with increasing number of CSLs of variable depths, participating nodes, and levels of distribution, the performance remains stable. This means that an implementation of our method does not add much more cost than already existing access constraint models, but provides a range of additional new functionalities.

Moreover, the scalability analysis showed that there exists an upper limit of CSLs that can be applied to XML documents, and that there is a potential performance gain of up to 50% for the validation of overlapping CSLs. Our derived cost model can be used to forecast performance gain, and consequently the expected computation time can be calculated.

## 6 Conclusion and Future Work

This paper has proposed (i) instance/schema level data access control for XML documents, (ii) composite security levels, and (iii) levels of access, offering a more rigorous concept for XML access control. The definitions and methodologies that are introduced in this paper were used to incorporate privacy-aware data access control into an existing XML data repository using XML triggers. Hence, XML data access control has been extended, so that XML document nodes can be assigned individual security levels, and combinations of nodes can also be given an additional (higher) level of security. A case study and analysis were used to demonstrate and quantify performance of the proposed concepts. Hereby, the new access control method was compared for three different test modes. Our scalability analysis showed that CSLs can be implemented in a cost-effective manner.

There are several avenues for future work. First, we want to show how the CSL approach can be integrated into query processing algorithms of (native) XML databases. In this way we wish to prove the applicability of the CSL mechanism. Second, we plan to conduct extensive experiments and case studies using more complex XML data in order to further prove scalability and robustness.

The conclusion of this paper is, that the proposed concepts of data access control have proven to be a beneficial extension to existing access models and XML, and that these new concepts also can be implemented and are scalable.

## References

- [1] P. Ayyagari, P. Mitra, D. Lee, P. Liu, and W.-C. Lee. Incremental adaptation of xpath access control views. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 105–116, 2007.
- [2] E. Bertino, S. Castano, E. Ferrari, and M. Mesiti. Specifying and enforcing access control policies for xml document sources. *World Wide Web*, 3(3):139–151, 2000.
- [3] E. Bertino and E. Ferrari. Secure and selective dissemination of xml documents. *ACM Trans. Inf. Syst. Secur.*, 5(3):290–331, 2002.
- [4] J. Crampton. Applying hierarchical and role-based access control to xml documents. In *SWS '04*, pages 37–46. ACM, 2004.
- [5] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. Securing xml documents. In C. Zaniolo, P. C. Lockemann, M. H. Scholl, and T. Grust, editors, *EDBT*, volume 1777 of *Lecture Notes in Computer Science*, pages 121–135. Springer, 2000.
- [6] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. A fine-grained access control system for xml documents. *ACM Trans. Inf. Syst. Secur.*, 5(2):169–202, 2002.
- [7] E. Damiani, M. Fansi, and A. G. an Stefania Marrara. A general approach to securely querying xml. In *Computer Standards & Interfaces*, pages 379–389, 2008.
- [8] W. Fan, C.-Y. Chan, and M. Garofalakis. Secure xml querying with security views. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 587–598. ACM, 2004.
- [9] S. K. Goel, C. Clifton, and A. Rosenthal. Derived access control specification for xml. In *XMLSEC '03: Proceedings of the 2003 ACM workshop on XML security*, pages 1–14. ACM, 2003.
- [10] G. Kuper, F. Massacci, and N. Rassadko. Generalized xml security views. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 77–84, New York, NY, USA, 2005. ACM Press.
- [11] A. H. Landberg, J. W. Rahayu, and E. Pardede. Extending xml triggers with path-granularity. In *WISE*, pages 410–422, 2007.
- [12] B. Luo, D. Lee, W.-C. Lee, and P. Liu. Qfilter: fine-grained run-time xml access control via nfa-based query rewriting. In *CIKM '04*, pages 543–552. ACM, 2004.
- [13] P. Röder, O. Tafreschi, and C. Eckert. History-based access control for xml documents. In *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*, pages 386–388. ACM, 2007.
- [14] P. Samarati, E. Bertino, and S. Jajodia. An authorization model for a distributed hypertext system. *IEEE Trans. Knowl. Data Eng.*, 8(4):555–562, 1996.
- [15] X. Xiao and Y. Tao. Anatomy: Simple and effective privacy preservation. In *VLDB*, pages 139–150, 2006.



# Systematic Clustering Method for $l$ -diversity Model

Md Enamul Kabir<sup>1</sup>, Hua Wang<sup>1</sup>, Elisa Bertino<sup>2</sup> & Yunxiang Chi<sup>3</sup>

<sup>1</sup>Department of Mathematics and Computing  
University of Southern Queensland,  
Toowoomba, Queensland 4350, Australia,  
Email: {kabir, wang}@usq.edu.au

<sup>2</sup>Department of Computer Science and CERIAS  
Purdue University, West Lafayette, Indiana, USA  
Email: bertino@cs.purdue.edu

<sup>3</sup>Toowoomba Pearl Company,  
Toowoomba, Queensland 4350, Australia,  
Email: toowoombapearls@yahoo.com.au

## Abstract

Nowadays privacy becomes a major concern and many research efforts have been dedicated to the development of privacy protecting technology. Anonymization techniques provide an efficient approach to protect data privacy. We recently proposed a systematic clustering<sup>1</sup> method based on  $k$ -anonymization technique that minimizes the information loss and at the same time assures data quality. In this paper, we extended our previous work on the systematic clustering method to  $l$ -diversity model that assumes that every group of indistinguishable records contains at least  $l$  distinct sensitive attributes values. The proposed technique adopts to group similar data together with  $l$ -diverse sensitive values and then anonymizes each group individually. The structure of systematic clustering problem for  $l$ -diversity model is defined, investigated through paradigm and is implemented in two steps, namely clustering step for  $k$ -anonymization and  $l$ -diverse step. Finally, two algorithms of the proposed problem in two steps are developed and shown that the time complexity is in  $O(\frac{n^2}{k})$  in the first step, where  $n$  is the total number of records containing individuals concerning their privacy and  $k$  is the anonymity parameter for  $k$ -anonymization.

**Keywords:** Privacy,  $k$ -anonymity,  $l$ -diversity, Systematic clustering.

## 1 Introduction

In recent years, the phenomenal advance technological developments in information technology have lead to an increase in the capability to store and record personal data about customers and individuals (Byun et al. 2006). Data mining is a common methodology to retrieve and discover useful hidden knowledge and information from the personal data. This has lead to concerns that the personal data may be breached and misused. Therefore it is necessary to protect personal data through some privacy preserving techniques before conducting data mining.

One of the most important concept for privacy is anonymity. Anonymity refers to a state where one's identity is completely hidden, and anonymity is oftentimes used as a synonym for privacy (Byun et al. 2007). Anonymous data can protect individuals in two ways, firstly to protect identity privacy for example it is not possible to learn about to whom a data record is related and secondly, attribute privacy for example not possible to know about particular property of individuals. In any databases specially where health records are collected by hospitals or government organizations, anonymity has a significant role to protect privacy as the information is linked to individuals could be highly sensitive. In commercial databases where organizations would like to disclose individual's data to third parties (e.g. external organizations), anonymity could be used to protect privacy of individuals as in such cases individual's privacy may not be respected. Thus within organizations individual's data should be restricted to access and anonymous by removing all information that can directly link data items to individuals via generalization or suppression before disclosing so that privacy is not breached. Such a process is referred to as data anonymization.

A contemporary approach dealing with the data privacy relies on the  $k$ -anonymity. The  $k$ -anonymity model proposed by Samarati (2001) and Sweeney (2002) is a simple and practical privacy-preserving approach to protect data from individual identification. The  $k$ -anonymity model works by ensuring that each record of a table is identical to at least  $(k - 1)$  other records with respect to a set of privacy-related features, called *quasi-identifiers*, that could be potentially used to identify individuals by linking these attributes to external data sets (Lin & Wei 2008). Therefore, privacy related information can't be revealed from the  $k$ -anonymity protected table during a data mining process. For example, consider the patient diagnosis records in a hospital in Table 1, where the attributes *ZipCode*, *Gender*, *Age* and *Education* are regarded as quasi-identifiers and *Disease* is a sensitive attribute. A diagnosis classifier can predict patients illness history based on attributes of *ZipCode*, *Gender*, *Age* and *Education* using these data. If the hospital simply publishes the table to other organizations for classifier development, the organizations might extract patients' disease history by joining this table with other tables (Chiu & Tsai 2007). On the contrary, Table 2 is a 3-anonymization version where data values of Table 1 in attributes *ZipCode*, *Gender*, *Age* and *Education* have been generalized as common values and the number of

Copyright ©2010, Australian Computer Society, Inc. This paper appeared at the Twenty-First Australasian Database Conference (ADC2010), Brisbane, Australia, January 2010. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 104, Heng Tao Shen and Athman Bouguettaya, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

<sup>1</sup>Clustering partitions record into clusters such that records within a cluster are similar to each other, while records in different clusters are most distinct from one another.

Table 1: Patients records in a hospital

	<i>ZipCode</i>	<i>Gender</i>	<i>Age</i>	<i>Education</i>	<i>Disease</i>	<i>Expense</i>
1	4350	Male	24	9th	Flue	2000
2	4351	Male	25	10th	Cancer	3500
3	4352	Male	26	9th	HIV+	6500
4	4350	Male	35	9th	Diabetes	2000
5	4350	Female	40	10th	Diabetes	3200
6	4350	Female	38	11th	Diabetes	2800
7	4352	male	41	9th	Flue	2700
8	4352	Female	42	10th	Heart disease	4800
9	4352	male	43	10th	Cancer	5200

Table 2: 3-Anonymization table

	<i>ZipCode</i>	<i>Gender</i>	<i>Age</i>	<i>Education</i>	<i>Disease</i>	<i>Expense</i>
1	435*	Person	21-30	Primary	Flue	2000
2	435*	Person	21-30	Primary	Cancer	3500
3	435*	Person	21-30	Primary	HIV+	6500
4	435*	Person	31-40	Secondary	Diabetes	2000
5	435*	Person	31-40	Secondary	Diabetes	3200
6	435*	Person	31-40	Secondary	Diabetes	2800
7	435*	Person	41-50	Primary	Flue	2700
8	435*	Person	41-50	Primary	Heart disease	4800
9	435*	Person	41-50	Primary	Cancer	5200

records in its two equivalence classes are both equal to three. It should be noted that the value of  $k$  in  $k$ -anonymity model is specified by users according to the purpose of their applications. By enforcing the  $k$ -anonymity requirement, it is guaranteed that even though an adversary knows that a  $k$ -anonymous table contains the record of a particular individual and also knows some of the quasi-identifier attribute values of the individual, he/she cannot determine which record in the table corresponds to the individual with a probability greater than  $\frac{1}{k}$  (Byun et al. 2007). This indicates that the larger the values of  $k$ , the adversary has less chance of determining personal identifiable information and the data is more protected. On the other hand, if the  $k$ -values are too large it incurs more information loss. So, the  $k$ -value of the  $k$ -anonymization problem should not be too small or too large.

Usually, there are two methods to accomplish in  $k$ -anonymizing a dataset. The first one is suppression which involves not releasing entire tuple or a value at all to the third party, just like deleting them. The other one is generalization which involves replacing the value or tuple with less specific but semantically consistent value. For example, suppose there exists following five ages of individuals 51, 52, 53, 53, 55. We can generalize attribute *Age* to a age groups 50-55. On the other hand, we can also generalize them to other set 5\*. However, we can suppress the age values by \*. Intuitively, generalization is better than suppression because of extracting some information. Undoubtedly, anonymization is accompanied with information loss. In order to be useful in practice, the dataset should keep as much informative as possible. Hence, it is necessary to consider deeply the tradeoff between privacy and information loss. To minimize the information loss due to  $k$ -anonymization, all records are partitioned into several groups such that each group contains at least  $k$  similar records with respect to the quasi-identifiers and then the records in each group are generalized or suppressed such that the values at each quasi-identifier are the same. Such similar groups are known as clusters. In the context

of data mining, clustering is a useful technique that partitions records into clusters such that records within a cluster are similar to each other, while records in different clusters are most distinct from one another (Lin & Wei 2008). So  $k$ -anonymity model can be addressed from the viewpoint of clustering and recently Kabir et al. (2009) proposed systematic clustering method for  $k$ -anonymization. The experimental results showed that the proposed method outperforms over the recent clustering based  $k$ -anonymization techniques. However  $k$ -anonymity model may reveal sensitive information under the two attacks, namely homogeneity attack and background knowledge attack (Machanavajjhala et al. 2006). For example, Jak and Ron are two antagonistic neighbors. Jak knows that Ron goes to hospital recently and tries to find out the disease Ron suffers. Jak finds the 3-anonymous table as in Table 2. He knows that Ron is 39 years's old and lives in the suburb with postcode 4350. Ron must be record 4, 5 or 6. All three patients are suffering from diabetes. Jak knows for sure that Ron suffers from diabetes. Thus homogeneous values in the sensitive attribute of a  $k$ -anonymous group escape private information. Similarly  $k$ -anonymity does not protect individuals from a background knowledge attack. To overcome this problem, Machanavajjhala et al. (2006) presented an  $l$ -diversity model to enhance the  $k$ -anonymity model. The  $l$ -diversity model assumes that a private dataset contains some sensitive attribute(s) which cannot be modified. Such a sensitive attribute is then considered disclosed when the association between a sensitive attribute value and a particular individual can be inferred with a significant probability. In order to prevent such inferences, the  $l$ -diversity model requires that every group of indistinguishable records contains at least  $l$  distinct sensitive attributes values; thereby the risk of attribute disclosure is kept under  $\frac{1}{l}$ . For example, records 4, 5 and 6 in Table 2 form a 3-diverse group. The records contain three values with equal frequencies of 33.33%, and no value is dominant. If we assume that  $l = 2$ , then although Table 2 is 3-anonymized but it is not a 2-diverse table as in the second equivalence class the num-

ber of sensitive attribute value is only one (Diabetes).

As discussed, a key difficulty of data anonymization comes from the fact that data quality and privacy are conflicting goals. Although it is possible to enhance data privacy by hiding more data values, it decreases data quality. On the contrary, disclosing more data values increases data quality but decreases data privacy. Thus it is necessary to devise new enhanced  $k$ -anonymization approaches (for example  $l$ -diversity) that best address both the quality and the privacy of the data. In the previous paper (Kabir et al. 2009), we developed systematic clustering method for  $k$ -anonymization. However, as  $l$ -diversity is more primitive and protected model than  $k$ -anonymization, it is necessary to extend the systematic clustering algorithm in  $l$ -diversity model. This extension of systematic clustering method to  $l$ -diversity model is presented in this paper. It has done in two steps. In the first steps it develops some clusters that satisfy the  $k$ -anonymity requirements, called clustering step for  $k$ -anonymization. According to this step, first exclude the number of records containing individuals who don't bother about the disclosure of personal identification information. Sort all records by their quasi-identifiers and partitions all records into  $\lceil \frac{n}{k} \rceil$  groups. Randomly select a record  $r$  from first group to form the first cluster and the first records of the subsequent clusters will form in a systematic way. Then adjusts the records in each group in a systematic way such that each group contains at least  $k$  records. Finally distribute the records of individuals who don't bother about the disclosure to their closest clusters or these records constitute another cluster/clusters depending on the number of such records and the  $k$ -value. Note that the process of including of such records cause no information loss. In the second step, it develops clusters that satisfy the  $l$ -diverse requirement on the sensitive attributes, called  $l$ -diverse step. According to this step, first remove clusters in the first step that does not satisfy  $l$ -diversity requirement. Then add the records containing in these clusters to other clusters that already satisfy  $l$ -diversity requirement where cause least information loss. Note that inclusion of new records to other clusters does not violate  $l$ -diversity requirement. There are many clustering based  $k$ -anonymization techniques (Byun et al. 2007, Loukides & Shao 2007, Chiu & Tsai 2007, Lin & Wei 2008, Gonzalez 1985) are available but to the best of our knowledge there is no such approaches exist for  $l$ -diversity model in the literature. Based on the leakages, this work is devoted a systematic clustering method for  $l$ -diversity model.

The reminder of this paper is organized as follows. We present some concepts relating to information loss and a brief overview of the clustering based approaches for  $k$ -anonymization in Section 2. In Section 3 we present proposed systematic clustering method for  $l$ -diversity model that consists in two steps. Important properties of the proposed algorithm are discussed in Section 4. We compare our proposed algorithms with the most recent clustering based algorithm in Section 5. Finally, concluding remarks are included in Section 6.

## 2 Preliminaries Relating to Anonymization

The  $k$ -anonymity model has drawn a considerable interest in the research community for the last few years and a number of algorithms have been proposed (Ciriani et al. 2008, Bayardo 2005, Fung et al. 2005, LeFevre et al. 2005, 2006, Sweeney 2002, Sun et al.

2008). However, these way out suffer from high information loss mainly due to reliance on pre-defined generalization hierarchies (Bayardo 2005, Fung et al. 2005, LeFevre et al. 2005, Sweeney 2002) or total order (Ciriani et al. 2008, LeFevre et al. 2006) imposed on each attribute domain. Some existing work on  $k$ -anonymization has attempted to capture usefulness by measuring the number of total suppressions (Meyerson 2004), the size of the anonymized group (Bayardo 2005, LeFevre et al. 2006), the height of generalisation hierarchies (Samarati 2001, Byun et al. 2007), or information loss through anonymization (Xu et al. 2006). However, such metrics fail to detain security. In other works Machanavajjhala et al. (2006), Truta & Vinary (2006) attempts have been made to enhance protection by enforcing anonymized groups. The intuition behind this is that if the values of a sensitive attribute of an anonymized group are quite diverse, then it is difficult for an attacker to breach privacy. However, these frequency-based criteria treat numerical attributes as categorical and thus protection is not captured adequately. For instance,  $l$ -diversity proposed by Machanavajjhala et al. (2006) requires a sensitive attribute to have at least  $l$  distinct values in an anonymized group. Please refer to Ciriani et al. (2008) and Machanavajjhala et al. (2006) for a survey of various  $k$  anonymization and  $l$ -diverse approaches.

### 2.1 Information Loss

Anonymization via generalization or suppression usually causes information loss. Now a natural question arise, how much information is lost due to anonymization. Thus the idea of information loss is used to measure the amount of information loss due to  $k$ -anonymization. There are various methods of coniving information loss (Bayardo 2005, Byun et al. 2007, Lin & Wei 2008, Solanas et al. 2008, Iyengar 2002). The measurement of information loss in this article is based on the description given by Byun et al. (2007). Please also refer to Byun et al. (2007) for more details.

Let  $\eta$  denote a set of records with  $r$  numeric quasi-identifiers  $N_1, N_2, \dots, N_r$  and  $s$  categorical quasi-identifiers  $C_1, C_2, \dots, C_s$ . Let  $\mathfrak{S} = \{\Omega_1, \Omega_2, \dots, \Omega_p\}$  be a partitioning of  $\eta$ , such that  $\cup_{i=1}^p \Omega_i = \eta$ ,  $\Omega_i$  and  $\Omega_j$  ( $i \neq j$ ) are pair wise mutually exclusive. To generalize the values of each categorical attribute  $C_i$  ( $i = 1, 2, \dots, s$ ), let  $\tau_{C_i}$  be the taxonomy tree defined for the domain of  $C_i$ .

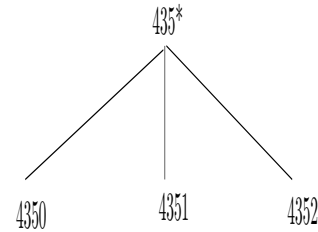


Figure 1: Taxonomy tree of *ZipCode*.

Consider a cluster  $\Omega$  in  $\eta$  which consists of some numerical and categorical attributes. Let  $N_{i_{max}}$ ,  $N_{i_{min}}$  be the maximum and minimum values of the records in  $\Omega$  and  $\eta_{N_{i_{max}}}$ ,  $\eta_{N_{i_{min}}}$  be the maximum and minimum values of the records in  $\eta$  with respect to numeric attribute  $N_i$  ( $i = 1, 2, \dots, r$ ) and  $\cup_{C_i}$  be the union set of values in  $\Omega$  with respect to the categorical attribute  $C_i$  ( $i = 1, 2, \dots, s$ ). Then the amount of information loss due to generalizing  $\Omega$ , denoted by

$IL(\Omega)$  is defined as

$$IL(\Omega) = |\Omega| \cdot \left( \sum_{i=1}^r \frac{N_{i_{max}} - N_{i_{min}}}{\eta_{N_{i_{max}}} - \eta_{N_{i_{min}}}} + \sum_{j=1}^s \frac{H(\Lambda(\cup_{C_j}))}{H(\tau_{C_j})} \right). \quad (1)$$

where  $|\Omega|$  is the number of records in  $\Omega$ ,  $\tau(\cup_{C_j})$  is the subtree rooted at the lowest common ancestor of every value in  $\cup_{C_j}$  and  $H(\tau)$  is the height of taxonomy tree  $\tau$ .

Suppose that the total number of records in  $\eta$  is partitioned into  $p$  clusters, namely  $\mathfrak{S} = \{\Omega_1, \Omega_2, \dots, \Omega_p\}$ . The total information loss of  $\eta$  is the sum of the information loss of each  $\Omega_i (i = 1, 2, \dots, p)$ . So the total information loss will be:

$$\begin{aligned} IL(\eta) &= \sum_{i=1}^p IL(\Omega_i) \\ &= \sum_{i=1}^p |\Omega_i| \cdot \left( \sum_{k=1}^r \frac{N_{ik_{max}} - N_{ik_{min}}}{\eta_{N_{ik_{max}}} - \eta_{N_{ik_{min}}}} \right. \\ &\quad \left. + \sum_{j=1}^s \frac{H(\Lambda(\cup_{C_{ij}}))}{H(\tau_{C_{ij}})} \right) \end{aligned} \quad (2)$$

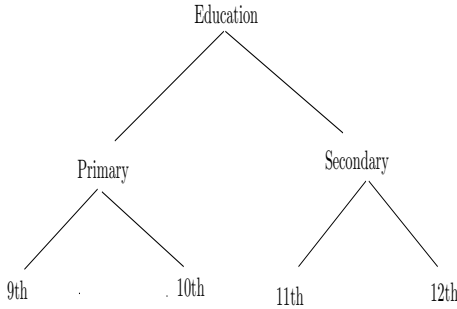


Figure 2: Taxonomy tree of *Education*.

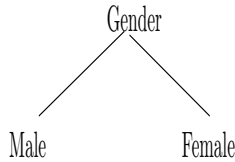


Figure 3: Taxonomy tree of *Gender*.

The main objective of clustering techniques is to construct the clusters in such a way that the total information loss of  $\eta$  will be minimum.

*Example:* Consider patients records in Table 1 and the 3-anonymization table in Table 2. The anonymized table consists of three clusters. The first cluster consists of first three records, the second cluster consists of middle three records and the last cluster consists of last three records. Consider attributes ZipCode, Gender, Age, Education, where Age is a quantitative variable and the others are categorical variable. Also consider the taxonomy tree of ZipCode, Education and Gender in Figure 1, Figure 2 and Figure 3 respectively. In the table the number of clusters are 3 and the size of each cluster is also 3. In the first cluster the maximum and minimum values respectively as 26 and 24, in the second cluster these values are respectively as 40 and 35 and finally in the last cluster these values are respectively as 43 and 41. Also the maximum and minimum values of all records

respectively as 43 and 24. Then the total information Loss of the anonymized table in Table 2 will be

$$\begin{aligned} IL(\eta) &= |3| \left( \frac{26-24}{43-24} + 1 + 1 + \frac{1}{2} \right) + |3| \left( \frac{40-35}{43-24} \right. \\ &\quad \left. + 1 + 1 + \frac{2}{2} \right) + |3| \left( \frac{43-41}{43-24} + 1 + 1 + \frac{1}{2} \right) \\ &\approx 25.44. \end{aligned} \quad (3)$$

## 2.2 Clustering based techniques

Clustering based techniques are now using in anonymization to protect the privacy of sensitive attributes and there are various  $k$ -anonymization clustering techniques in the literature (LeFevre et al. 2006, Byun et al. 2007, Loukides & Shao 2007, Chiu & Tsai 2007, Lin & Wei 2008). Byun et al. (2007) introduced clustering techniques instead of equivalence class on  $k$  anonymization and proposed the greedy  $k$ -member clustering algorithm. This algorithm works by first randomly selecting a record  $r$  as the seed to start building a cluster, and subsequently selecting and adding more records to the cluster such that the added records incur the least information loss within the cluster. Once the number of records in this cluster reaches  $k$ , this algorithm selects a new record that is the furthest from  $r$ , and repeats the same process to build the next cluster. When there are fewer than  $k$  records not assigned to any cluster yet, this algorithm then individually assigns these records to their closest clusters. This algorithm has two drawbacks. First, it is slow. Second, it is sensitive to outliers. To build a new cluster, this algorithm chooses a new record that is the furthest from the first record selected for previous cluster. If the data contains outliers, it is likely that outliers have a great chance of being selected. If a cluster contains outliers, the information loss of this cluster increases. The time complexity of the algorithm is  $O(n^2)$ , where  $n$  is the number of records in the data set to be anonymized. Their experimental results showed that the  $k$ -member algorithm causes significantly less information loss than another  $k$ -anonymization technique called “Mondrian” proposed by LeFevre et al. (2006).

Loukides & Shao (2007) proposed another clustering technique for  $k$ -anonymization. Similar to  $k$ -member this algorithm forms one cluster at a time. But, unlike the  $k$ -member algorithm, this algorithm chooses the seed of each cluster randomly. Also, when building a cluster, this algorithm keeps selecting and adding records to the cluster until the information loss exceeds a user defined threshold. If the number of records of a particular class is less than  $k$ , the entire cluster is deleted. With the help of the user-defined threshold, this algorithm is less sensitive to outliers. The time complexity of the algorithm is  $O(\frac{n^2 \log(n)}{c})$ , where  $c$  is the average number of records in each cluster. However, this algorithm also has two drawbacks. First, it is difficult to decide a proper value for the user-defined threshold. Second, this algorithm might delete many records, which in turn cause a significant information loss. Chiu and Tsai (Chiu & Tsai 2007) proposed another algorithm for  $k$ -anonymization that adapts the weighted feature  $c$ -means clustering. Unlike the previous two algorithms, this algorithm attempts to build all clusters simultaneously by first randomly selecting  $\lfloor \frac{n}{k} \rfloor$  records as seeds. Then this algorithm allocates all records in the data set to their respective

closest cluster and consequently updates feature weights to minimize information loss. This process is continued until the assignment of records to cluster stops changing. If some clusters contain fewer than  $k$  records, merge those clusters with other large clusters to satisfy the  $k$ -anonymity requirement. One of the main drawback of this algorithm is that it can only be used for quantitative quasi-identifier. The time complexity of this algorithm is  $O(\frac{t^2}{k})$ , where  $t$  is the number of iterations needed for the assignment of records to clusters to converge.

To reduce the information loss and execution time recently Lin & Wei (2008), proposed an efficient one-pass  $k$ -mean clustering problem that runs in  $O(\frac{n^2}{k})$ . They showed that their algorithm performs better than the proposed algorithm of Byun et al. (2007) with respect to both execution time and information loss. Like Chiu & Tsai (2007)'s algorithm, this algorithm forms all clusters at a time. According to their methods first sort all records by their quasi-identifiers, determine approximate number of clusters, by  $p = \frac{n}{k}$ , where  $k$  is the cluster size. Then randomly select  $p$  records as seeds to build  $p$  clusters. For each record  $r$  the algorithm finds the cluster that is closet to  $r$ , assign  $r$  to that cluster and subsequently updates the center point. Finally, if some clusters contain more than  $k$  records remove excess records from those clusters that are dissimilar to most of the records and then add these records to other similar clusters (whose size less than  $k$ ). Although this method has less execution time there is still chance of being affected by extreme values. Again if this algorithm first selects  $p$  records that come from same equivalent class then the total information loss will be higher. All of these clustering techniques are based on  $k$ -anonymization techniques. However there is no such approach is available in the literature for  $l$ -diversity.

Very recently Kabir et al. (2009) proposed systematic clustering method for  $k$ -anonymization that run in  $O(\frac{n^2}{k})$ , where  $n$  is the total number of records containing individuals concerning their privacy. It has shown by experiment that the method attains a reasonable dominance with respect to both information loss and execution time over the recent clustering algorithm. The proposed systematic clustering method differs from previously proposed clustering based  $k$ -anonymization methods in four different ways. First, it endeavour to make all clusters simultaneously. On the contrary, the methods proposed by Byun et al. (2007) and Loukides & Shao (2007) build one cluster at a time. Second, it takes less time than the previous two methods as only the first record randomly selects and the subsequently records from in a systematic way. Third, since first record of each clusters contains non identical value, this method easily captures if there are any extreme values and lastly the total information loss will be reduced as in the final step the process of incurring no information loss. Based on the performance of this algorithm, in this paper it is implemented in  $l$ -diversity model.

### 3 Clustering for $l$ -diversity

As discussed before, clustering escorts to better data quality of the disclosed dataset as it partitions a set of records into groups such that records in the same group are more similar to each other than records to other groups. If the records in a particular group are more similar, the group leads to a minimal generalization and thus incurs less information loss. In this

respect, the problem of  $k$ -anonymization can also be considered as a clustering problem, where each equivalent class is a cluster and the size of each cluster is at least  $k$ . But the requirement for  $l$ -diversity model to satisfy at least  $l$  distinct sensitive attribute values in each equivalent class. So the optimal solution of clustering problem is to construct a set of clusters such that it satisfies both  $k$ -anonymity and  $l$ -diversity requirement and the total information loss will be as minimum as possible. In this section, we formally define and present our systematic clustering algorithm that minimizes the information loss and respects the  $k$ -anonymity and  $l$ -diversity requirement.

#### 3.1 Systematic clustering problem

There are various clustering problems in the literature. Among them, the  $k$ -center clustering problem proposed by Gonzalez (1985) aims to find  $k$  clusters from a given dataset such that the maximum inter-cluster distance (or radius) is minimized. Thus the optimum solution is to constitute  $p$  clusters  $\{\Omega_1, \Omega_2, \dots, \Omega_p\}$  in such a way that minimizes the cost metric

$$MAX_{i=1, \dots, p} MAX_{j,k=1, \dots, |\Omega_i|} D(r_{i,j}, r_{i,k}), \quad (4)$$

where  $r_{i,j}$  represents a data point in cluster  $\Omega_i$  and  $D(x, y)$  is a distance between two data points,  $x$  and  $y$ .

In the  $k$ -anonymity problem the restriction is that the number of records in each equivalence class should be at least  $k$  and in the  $l$ -diversity model the restriction is that the number of sensitive attribute values in each equivalence class must be at least  $l$  distinct values but there is no such restriction about the number of clusters in both cases. So a clustering problem is to form in such a way that each cluster contains at least  $k$  similar records,  $l$  distinct sensitive records and the sum of information losses of all clusters is as small as possible. For applying systematic method to  $l$ -diversity model of selecting records we need to follow two steps. The first one is the clustering step for  $k$ -anonymization and the second one is the  $l$ -diverse step. Suppose that we would like to apply systematic clustering method to  $l$ -diversity model for Table 1. Then in the clustering step for  $k$ -anonymization first sort all records in the whole data set with respect to quasi-identifiers. There are 9 records in Table 1 and suppose that dataset already sorted according to the quasi identifier attributes *ZipCode*, *Gender*, *Age* and *Education*. If the anonymized table follows 3-anonymity requirements, then the number of clusters should be  $\frac{9}{3} = 3$ . Select a record (say, 2th record) from the first 3 records to form the first cluster. Then select  $(2 + 3)th = 5th$  and  $(2 + 2 \times 3)th = 8th$  records in a systematic way to form the second and third cluster respectively. Now again select another record from the first 3 records (say, 3rd not 2th as it already selected) and calculate the information loss with all of the three clusters using the equation (1). The information losses are respectively as 5.10, 6.47 and 6.68, if this record includes in the first, second and third cluster. So, 3rd record will be included in the first cluster as it causes least information loss. Similarly select  $(3 + 3)th = 6th$  and  $(3 + 2 \times 3)th = 9th$  record in a systematic way and include them in the second and third cluster respectively. Finally select 1st,  $(1 + 3)th = 4th$  and  $(1 + 2 \times 3)th = 7th$  record and include these records to the first, second and third cluster respectively as they will then cause least information loss. If the total number of records is not exactly divisible by

the  $k$ -anonymity parameter, then rest records will be included to the similar clusters where information loss is minimum and this process continues until the number of records in a particular cluster is  $k$  to satisfy  $k$ -anonymity requirement. Thus in the clustering step for  $k$ -anonymization as set of clusters are built that satisfy the  $k$ -anonymity requirement. In the  $l$ -diverse step, the clusters will be formed in such a way that the number of distinct sensitive attribute values in each cluster is at least  $l$ . Note that if in the clustering step the table already satisfies  $l$ -diversity requirement, next step is not required. Suppose that  $l = 3$ , in this particular example, so the clusters that are obtained in the first step does not satisfy  $l$  diversity requirement as the second cluster consists only one distinct sensitive attribute value. So in the  $l$ -diverse step remove this cluster and the records containing in this cluster to other similar clusters that causes least information loss. All of the three records in this cluster will be included in the third cluster as these records will then incur less information loss. Thus we get a table in Table 3 that satisfy both the 3-anonymity and 3-diversity requirements. The process of building the table by using systematic method protects individuals private information as well as sensitive attributes.

**Definition 1 (Systematic clustering problem for  $l$ -diversity)** *The systematic clustering problem is to find a set of clusters from a given set of  $n$  records such that each cluster contains at least  $k$  ( $k \leq n$ ) records (where the records select in a systematic way and include in a cluster that cause least information loss), the number of distinct sensitive attribute values is at least  $l$  ( $l \geq 2$ ) and that the sum of all intra-cluster distances is as minimum as possible. More specifically, if  $\eta$  be a set of  $n$  records and  $k$  &  $l$  are the specified anonymization and diversity parameter, the optimal solution of the systematic clustering problem is a set of clusters  $\mathfrak{S} = \{\Omega_1, \Omega_2, \dots\}$  such that:*

1.  $\Omega_i \cap \Omega_j = \Phi$ , for all  $i \neq j = 1, 2, \dots$ ,
2.  $\cup_{i=1, \dots} \Omega_i = \eta$ ,
3. for all  $\Omega_i \in \mathfrak{S}$ ,  $|\Omega_i| \geq k$  &  $l \geq 2$ , and
4. the total information loss obtained by using equation (2) is minimized.

In Definition 1, a set of clusters are constructed in such a way that the clusters are mutually exclusive, the sum of records of all clusters is equal to the total number of records, the size of each cluster is at least  $k$  and the number of distinct sensitive attribute values is at least  $l$  that satisfies both the criteria of  $k$ -anonymization and  $l$ -diversity. The problem tries to minimize the sum of all intra-cluster distances, where an intra-cluster distance of a cluster is defined as the maximum distance between any two records in the cluster. In the following subsection we formally design a systematic clustering algorithm for  $l$ -diversity.

### 3.2 Systematic clustering algorithm

Based on the information loss in Subsection (2.1) and the definition of systematic clustering problem, we are now ready to discuss a systematic clustering algorithm for  $l$ -diversity model. As discussed, the whole procedure consists of the two steps, namely clustering step for  $k$ -anonymization and  $l$ -diverse step.

*Clustering step for  $k$ -anonymization*

Table 4: Clustering step for  $k$ -anonymization algorithm

<p>Input: a set <math>\eta</math> of <math>n</math> records containing individuals concerning their privacy, where <math>\eta_1, \eta_2, \dots, \eta_n \in \eta</math>; the value <math>k</math> for <math>k</math>-anonymity</p> <p>Output: a partitioning <math>\mathfrak{S} = \{\Omega_1, \Omega_2, \dots, \Omega_p\}</math> of <math>\tau</math></p> <ol style="list-style-type: none"> <li>1. Sort all records in <math>\eta</math> by their quasi-identifiers;</li> <li>2. Let <math>p := \text{int}(\frac{n}{k})</math>;</li> <li>3. Get randomly <math>k</math> distinct records <math>r_1, r_2, \dots, r_k</math> from first 1 to <math>k</math>;</li> <li>4. Let <math>p_{ij}</math> is the <math>j</math>th element in the <math>i</math>th cluster;</li> <li>5. For <math>i = 1</math> to <math>p</math>;</li> <li>6. Let <math>p_{i1} := \eta_{[r_1 + k(i-1)]}</math>;</li> <li>7. Next <math>i</math>;</li> <li>8. For <math>j := 2</math> to <math>k</math>;</li> <li>9. For <math>i := 1</math> to <math>p</math>;</li> <li>10. Let <math>IL_i := \text{InfoLoss}(\eta_{[r_j + k(i-1)]})</math>;</li> <li>11. Let <math>X := \text{Find cluster number with lowest } IL_i</math>;</li> <li>12. where cluster size <math>\leq k</math>;</li> <li>13. Add <math>\eta_{[r_j + p(i-1)]}</math> to <math>p_x</math>;</li> <li>14. Next <math>i</math>;</li> <li>15. Next <math>j</math>;</li> <li>16. Let <math>e := (n - pk)</math>;</li> <li>17. Find extra element <math>E_1, E_2, \dots, E_e \in E</math>;</li> <li>18. For <math>k := 1</math> to <math>e</math>;</li> <li>19. For <math>m := 1</math> to <math>p</math>;</li> <li>20. Let <math>IL_m := \text{InfoLoss}(E_k)</math> in cluster <math>m</math>;</li> <li>21. Next <math>m</math>;</li> <li>22. Let <math>X := \text{Find cluster number with lowest } IL</math>;</li> <li>23. Add <math>E_k</math> to <math>p_x</math>;</li> <li>24. Next <math>k</math>;</li> </ol>
--

The aim of this step is to develop a set of clusters from a given set of  $n$  records that satisfy the  $k$ -anonymity requirement. The general idea if this step is as follows:

Note that for collecting medical data from patients it may be expected that some patients are not concerned about the privacy of their medical records and the other attributes. We would like to explore this opportunities because unnecessary anonymization may produce more information loss. Let  $q$  be the probability that a particular patient is not concerned about the privacy of medical records. Then out of  $n$  patients we can expect that on an average  $nq$  patients are not concerned about their privacy. According to this step first exclude the records of individuals who are not concerned about the privacy. Then sort all records by their quasi-identifiers and identify the equivalence class and the number of clusters by,  $p = \frac{(n-nq)}{k}$ , where  $k$  is anonymity parameter for  $k$ -anonymization and round this as integer. Randomly select a record  $r_i$  from first  $k$  records as seeds to form the first cluster. If there are  $p$  clusters to be formed then select the  $(r_i + k)$ th,  $(r_i + 2k)$ th, ...,  $\{r_i + (p-1)k\}$ th records in a systematic way to form 2nd, 3rd, ...,  $p$ th cluster respectively. Select another record  $r_j (j \neq i)$  from the first  $k$  records and add this record to the cluster which causes least information loss. Similar in a systematic way select  $(r_j + k)$ th,  $(r_j + 2k)$ th, ...,  $\{r_j + (p-1)k\}$ th records and add these records to their respective clusters that cause least information loss. If any cluster size is exactly  $k$ , stop adding records to that cluster and continue the same process until all records of first  $k$  records finish. If  $(n - nq)$  is not exactly divisible by  $k$  and still there are some records left, add this records to their closest clusters that incur least information loss. Finally distribute the  $nq$  records to their closest clusters or these  $nq$  records constitute separate cluster/clusters depending on their size. Note that these  $nq$  records do not incur any information loss. Since only the first record randomly selects and the subsequent records from in a systematic way, it has less execution time. Again usually the first record of each cluster

Table 3: 3-diversity table

	ZipCode	Gender	Age	Education	Disease	Expense
1	435*	Person	21-30	Primary	Flue	2000
2	435*	Person	21-30	Primary	Cancer	3500
3	435*	Person	21-30	Primary	HIV+	6500
4	435*	Person	31-50	Educated	Diabetes	2000
5	435*	Person	31-50	Educated	Diabetes	3200
6	435*	Person	31-50	Educated	Diabetes	2800
7	435*	Person	31-50	Educated	Flue	2700
8	435*	Person	31-50	Educated	Heart disease	4800
9	435*	Person	31-50	Educated	Cancer	5200

contains non identical value, so this algorithm easily captures if there are any extreme values. Moreover, this algorithm is adding some records that contain no information loss, so it is a natural expectation that the total information loss will be reduced. The clustering step for  $k$ -anonymization algorithm is shown in Table 4. In the algorithm it is assumed that all  $n$  individuals are concerned about their privacy. Thus in this step, we get some clusters that satisfy the  $k$ -anonymity requirement and the total information loss of all of these clusters will be minimum.

#### $l$ -diverse step

As discussed in the previous section, we have some clusters that satisfy  $k$ -anonymity requirement but may or may not satisfy  $l$ -diversity requirement. Note that  $l$ -diverse step is invoked only if in the first step, some of the clusters in the  $k$ -anonymization table does not satisfy  $l$ -diversity requirement. If for a certain  $l$ -value, all clusters in the anonymized table satisfy the  $l$ -diversity requirement, the  $l$ -diverse step of the table is not required. According to this step, remove the clusters that do not satisfy  $l$ -diversity requirements and add the records containing in these clusters to other clusters that cause least information loss. As the existing clusters already satisfy  $k$ -anonymity and  $l$ -diversity requirement, inclusion of new records don't violate these requirement. The algorithm of the  $l$ -diverse step is illustrated Table 5.

Table 5:  $l$ -diverse algorithm

Input: a partitioning  $\mathfrak{S}_1 = \{\Omega_1, \Omega_2, \dots, \Omega_{p_1}\}$  of  $\tau$  that satisfy  $k$ -anonymity requirement, a partitioning  $\mathfrak{S}_1^* = \{\Omega_1^*, \Omega_2^*, \dots, \Omega_{p_2}^*\}$  of  $\tau$  that satisfy both the  $k$ -anonymity and the  $l$ -diversity requirement, a set of sensitive attributes  $S_i (i = 1, 2, 3, \dots)$ , and the value of  $l$  for  $l$ -diversity.  
Output: a partitioning  $\mathfrak{S} = \{\Omega_1, \Omega_2, \dots\}$  of  $\tau$  that satisfy the  $l$ -diversity requirement.

1. Let  $\Upsilon = \{r_1, r_2, \dots\} = \{\text{all records of } \mathfrak{S}_1\}$
2. Let  $r_j$  is the  $j$ th record of  $\Upsilon$ ;
3. For  $j = 1, 2, \dots$ ;
4. For  $i = 1, 2, \dots, p_2$ ;
5. Let  $IL_i^* := \text{InfoLoss}(\Omega_i^*), i = 1, 2, \dots, p_2$ ;
6. Find the cluster  $\Omega_i^*$  with lowest  $IL_i^*$ ;
7. Add  $r_j$  to  $\Omega_i^*$ ;
8. Next  $j$ ;

**Definition 2 (Systematic clustering decision problem for  $l$ -diversity)** In a given data set of  $n$  records, there is a clustering scheme  $\mathfrak{S} = \{\Omega_1, \Omega_2, \dots\}$  such that

1.  $|\Omega_i| \geq k, 1 < k \leq n$ : the size of each cluster is greater than or equal to a positive integer  $k$ ,
2.  $l \geq 2$ , the number of distinct sensitive attribute values in each cluster is at least 2, and

3.  $\sum_{i=1} IL(\Omega_i) < c, c > 0$ : the total information loss of the clustering scheme is less than a positive integer  $c$ .

where each cluster  $\Omega_i (i = 1, 2, \dots)$  contains the records that are more similar to each other with respect to  $k$  and  $l$  such that require minimum generalization and thus causes least information loss. In the following subsection we are going to discuss some properties of the proposed systematic clustering algorithm.

#### 4 Analysis of the new algorithm

As discussed before, the proposed algorithm is designed in such a way that finds a solution of  $l$ -diversity model. In the first step, this algorithm stops adding records in a particular cluster if the cluster size exactly  $k$ . Again it always keep in mind in adding records that incur less information loss. Moreover, the records are selected in a systematic way that makes the algorithm faster. With respect to this, this algorithm has the following desirable properties.

**Theorem 1.** Let  $n$  be the total number of input records and  $k$  be the specified  $k$  anonymity parameter. The time complexity of the clustering step for  $k$ -anonymization is in  $O(\frac{n^2}{k})$ .

*Proof.* In the clustering step for  $k$ -anonymization, after sorting the records with respect to the quasi-identifiers the algorithm determine the numbers of clusters by  $p = \frac{n}{k}$ . Then it selects the records as seeds in a systematic way to form all  $p$  clusters simultaneously. Thus for each tuple in the dataset, the algorithm needs to assign it to one of the  $p$  clusters, which has a complexity of  $O(p)$ . As a result, the assignment of all tuples to the clusters has a time complexity of

$$\begin{aligned} T &= O(\text{Number of tuples} * \text{Number of clusters}) \\ &= O(n * p) = O(n * \frac{n}{k}) = O(\frac{n^2}{k}). \end{aligned} \quad (5)$$

Therefore, the total execution time is in  $O(\frac{n^2}{k})$ .

As discussed, in the first step the algorithm develops clusters that satisfy the  $k$ -anonymity requirement. The second step is required only if in the first step some clusters does not satisfy  $l$ -diversity requirement. The time complexity in the second step thus depends on number of such clusters and the reassignment of records to other clusters. So the time complexity in the second step is not straight forward.

**Theorem 2.** *Let  $n$  be the total number of input records and  $q$  be the probability that a particular individual doesn't bother about the disclosure. Then the algorithms in fact work out the information loss of  $(n - nq)$  individuals instead of all  $n$  individuals.*

*Proof.* If  $q$  be the probability that a particular individual doesn't bother about the disclosure. Then out of  $n$  individuals,  $nq$  individuals are not bothered about the disclosure. Assume that these  $nq$  records are in one separate cluster that causes no information loss. Also let  $IL(\eta)$  and  $IL(\eta_{all})$  are the total information loss due to  $l$ -diversity model and any other clustering algorithm respectively. According to the systematic clustering algorithm, the total information loss will be:

$$\begin{aligned} IL(\eta) &= IL(n) \\ &= IL(nq) + IL(n - nq) \\ &= 0 + IL(n - nq) = IL(n - nq). \end{aligned} \quad (6)$$

So in the systematic clustering algorithm for  $l$ -diversity model, it actually calculate the information loss of  $(n - nq)$  records instead of calculating the information loss of all  $n$  records.

**Theorem 3.** *Let  $n$  be the total number of input records and  $k$  be the anonymity parameter in  $k$ -anonymization. Then according to the algorithm in first step, the cluster size of any cluster is at least  $k$  but no more than  $(2k - 1)$ .*

*Proof.* Let  $n$  be the total number of input records. According to clustering step for  $k$ -anonymization, first select the initial seeds of all clusters in a systematic way and subsequently selecting adding more records to the clusters such that the added records incur least information loss. Again this algorithm stops adding record to a particular cluster if the number of records is exactly  $k$ . So in the worst case, if there are  $(k - 1)$  records left and if these all records include in a cluster that already contains  $k$  records, the total number of records in that cluster will be  $(k + k - 1) = (2k - 1)$ . Therefore the maximum size of a cluster will be  $(2k - 1)$ .

## 5 Comparison

As discussed before, to the best of our knowledge no clustering methods exist in the literature for  $l$ -diversity model. Most of the clustering based approaches (LeFevre et al. 2006, Byun et al. 2007, Loukides & Shao 2007, Chiu & Tsai 2007, Lin & Wei 2008, Kabir et al. 2009) are based on  $k$ -anonymization techniques. However, the closest works related to this paper is the systematic clustering method for  $k$ -anonymization proposed by Kabir et al. (2009) and the one pass  $k$ -means algorithm (OKA) proposed by Lin & Wei (2008). In this section we compare our proposed algorithms with these two algorithms.

Kabir et al. (2009) developed an algorithm that selects records in a systematic way to form the clusters and endeavors to make all clusters simultaneously. Experimental results showed that systematic clustering method is the best fit for  $k$ -anonymization with respect to both information loss and execution time over the recent clustering algorithms. The first step of the clustering technique proposed in this paper is exactly the same as Kabir et al. (2009), however in the second step we developed an algorithm for  $l$ -diversity model that has significantly improve the

work of Kabir et al. (2009). According to the algorithm of Kabir et al. (2009) clusters are formed that satisfy the  $k$ -anonymity requirement. By contrast, by using the algorithms proposed in this paper clusters are formed that satisfy both the  $k$ -anonymity and  $l$ -diversity requirement that their work did not provide.

Lin & Wei (2008) proposed a two-stage algorithm, called one pass  $k$ -means algorithm (OKA). During the first stage, the algorithm clusters data using the  $k$ -means algorithm, but only for the first iteration. During the second stage, records are moved from clusters with more than  $k$  records (called the shrinking clusters) to clusters with fewer than  $k$  records (called the growing clusters) to ensure that each cluster eventually contains no fewer than  $k$  records. The algorithm is designed for  $k$ -anonymization but the authors did not implement the algorithm for  $l$ -diversity model. By contrast, the proposed algorithms presented in this paper are intended for  $l$ -diversity model.

## 6 Conclusion and future works

In this paper, we propose algorithms for  $l$ -diversity model as an enhanced of  $k$ -anonymity model. The proposed technique uses the idea of clustering and is implemented in two steps, namely clustering step for  $k$ -anonymization and  $l$ -diverse step. The basic concepts of the proposed algorithms are discussed and investigated through example and properties. The time complexity of the developed algorithm is in  $O(\frac{n^2}{k})$ , where  $n$  is the total number of records containing individuals concerning their privacy in the first step. The effect of the proposed approach can be useful for protecting private information of individuals as  $l$ -diversity model is one of the most popular approaches for privacy preserving techniques.

The proposed approach in this paper is implemented through paradigm. Our future work is to conduct an experimental study to show the efficiency and the effectiveness of the proposed algorithms. Again recently there are many disparities of the  $k$ -anonymity model have been proposed in the literature to further protect the private information, e.g.,  $t$ -closeness (Li 2007),  $(\alpha, k)$ -anonymity (Wong et al. 2006). Our further work is also to extend the algorithms to these models.

## References

- Samrati, P. (2001), Protecting respondent's privacy in microdata release, in 'TKDE'.
- Sweeney, L. (2002), 'K-anonymity: a model for protecting privacy', *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems* **10**(5), 557–570.
- Ciriani, V., di Vimercati, S.D.C., Foresti, S. & Samarti, P. (2008),  $k$ -anonymous data mining: A survey, in 'Privacy-preserving data mining: Models and algorithms', C.C. Aggarwal and P.S. Yu, Eds. Boston: Kluwer Academic Publishers, 103–134.
- Bayardo, R.J. & Agrawal, R. (2005), Data privacy through optimal  $k$ -anonymization, in 'ICDE'.
- Fung, B.C.M., Wang, K. & Yu, P.S. (2005), Top-down specialization for information and privacy preservation, in 'ICDE'.
- LeFevre, K., DeWitt, D. & Ramakrishnan, R. (2005), Incogniti: Efficient full-domain  $k$ -anonymity, in



- 'ACM International Conference on Management of Data'.
- LeFevre, K., DeWitt, D. & Ramakrishnan, R. (2006), Mondrian multidimensional  $k$ -anonymity, in 'ICDE'.
- Sweeney, L. (2002), 'Achieving  $k$ -anonymity privacy protection using generalization and suppression', *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems* **10**(5), 571–588, 2002.
- Byun, J.W. & Bertino, E. (2006), 'Micro-views, or on how to protect privacy while enhancing data usability: concepts and challenges', *SIGMOD* **35**(1), 9–13, 2006.
- Byun, J.W., Sohn, Y., Bertino, E. & Li, N. (2006), Secure anonymization for incremental datasets, in '3rd VLDB Workshop on Secure Data Management'.
- Byun, J.W., Kamra, A., Bertino, E. & Li, N. (2006), Efficient  $k$ -anonymization using clustering techniques, in 'International Conference on Database Systems for Advanced Applications'.
- Loukides, G. & Shao, J. (2007), Capturing data usefulness and privacy protection in  $k$ -anonymisation, in 'Proceedings of the 2007 ACM symposium on Applied Computing'.
- Chiu, C.C. & Tsai, C.Y. (2007), A  $k$ -anonymity clustering method for effective data privacy preservation, in 'Third International Conference on Advanced Data Mining and Applications'.
- Lin, J.L. & Wei, M.C. (2008), An efficient clustering method for  $k$ -anonymization, in 'Proceedings of the 2008 international workshop on Privacy and anonymity in information society'.
- Meyerson, A. & Williams, R. (2004), On the complexity of optimal  $k$ -anonymity, in 'PODS'.
- Xu, J., Wang, W., Pei, J., Wang, X., Shi, B. & Fu, A.W.C. (2006), Utility-based anonymization using local recording, in 'KDD'.
- Machanavajjhala, A., Gehrke, J., Kifer, D. & Venkatasubramanian, M. (2006),  $l$ -diversity: Privacy beyond  $k$ -anonymity, in 'ICDE'.
- Truta, T. & Vinary, B. (2006), Privacy protection:  $p$ -sensitive  $k$ -anonymity property, in 'International Workshop on Privacy Data Management'.
- Sun, X., Li, M., Wang, H. & Plank, A. (2008), An efficient hash-based algorithm for minimal  $k$ -anonymity, in 'ACSC'.
- Sun, X., Wang, H. & Li, J. (2008), Priority Driven  $K$ -Anonymisation for Privacy Protection, in 'AusDM'.
- Hettich, C.B.S. & Merz, C. (1998), UCI repository of machine learning databases.
- Gonzalez, T.Z. (2002), 'Clustering to minimize the maximum intercluster distance', *Theoretical Computer Science* **38**, 293–306, 1985.
- Li, N. & Li, T. (2007),  $t$ -closeness: Privacy beyond  $k$ -anonymity and  $l$ -diversity, in 'ICDE'.
- Wong, R.C.W., Li, J., Fu, A.W.C. & Wang, K. (2006),  $(\alpha, k)$ -anonymity: an enhanced  $k$ -anonymity model for privacy preserving data publishing, in 'Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining'.
- Solanas, A., Sebe, F. & Domingo-Ferrer, J. (2006), Micro-aggregation-based heuristics for  $p$ -sensitive  $k$ -anonymity: One step beyond, in 'International Workshop on Privacy and Anonymity in the Information Society'.
- Iyengar, V.S. (2002), Transforming data to satisfy privacy constraints, in 'SIGKDD'.
- Kabir, M.E., Wang, H. & Bertino, E. (2009), Efficient Systematic Clustering Method for  $k$ -Anonymization, in 'Working Paper Series', No SC-MC-0905, Faculty of Sciences, University of Southern Queensland, Australia.
- Li, J., Wang, H., Jin, H. & Yong, J. (2008), 'Current Developments of  $k$ -Anonymous Data Releasing', *Electronic Journal of Health Informatics* **3**(1), e6, 2008.



# Tuning QoD in Stream Processing Engines

Mohamed A. Sharaf<sup>1</sup> Panos K. Chrysanthis<sup>2</sup> Alexandros Labrinidis<sup>2</sup>

<sup>1</sup> ECE Department, University of Toronto

<sup>2</sup> CS Department, University of Pittsburgh

msharaf@eecg.toronto.edu {panos, labrinid}@cs.pitt.edu

## Abstract

Quality of Service (QoS) and Quality of Data (QoD) are the two major dimensions for evaluating any query processing system. In the context of data stream management systems (DSMSs), multi-query scheduling has been exploited to improve QoS. In this paper, we are proposing to exploit query scheduling to improve QoD in DSMSs. Specifically, we are presenting a new policy for scheduling multiple continuous queries with the objective of maximizing the freshness of the output data streams and hence the QoD of such outputs. The proposed Freshness-Aware Scheduling of Multiple Continuous Queries (FAS-MCQ) policy decides the execution order of continuous queries based on each query's properties (i.e., cost and selectivity) as well the properties of the input update streams (i.e., variability of updates). Our experimental results have shown that FAS-MCQ can improve QoD by up to 50% compared to existing scheduling policies used in DSMSs. Finally, we propose and evaluate a parametrized version of our FAS-MCQ scheduler that is able to balance the trade-off between freshness and response time according to the application's requirements.

**Keywords:** Quality of Service (QoS), Quality of Data (QoD), Data Freshness, Data Stream Management Systems, Continuous Queries, Operator Scheduling.

## 1 Introduction

*Data streams processing* is an emerging research area that is driven by the growing need for *monitoring applications*. A monitoring application continuously processes streams of data for interesting, significant, or anomalous events, as defined by the users. Monitoring applications have been used in important business and scientific information systems, for example, monitoring network performance, real-time detection of disease outbreaks, tracking the stock market, performing environmental monitoring via sensor networks, providing personalized and customized Web pages.

For example, consider the University of Pittsburgh's Realtime Outbreak of Disease Surveillance

System (<http://rods.health.pitt.edu>). Such a system receives data from different sources (e.g., hospitals, clinics, pharmacies, etc.) and integrates it together in order to detect correlations or abnormal events. In the event of detecting a disease outbreak, CDC and health departments are notified to start mobilizing their resources.

Efficient employment of monitoring applications needs advanced data processing techniques that can support the continuous processing of rapid unbounded data streams. Such techniques go beyond the capabilities of traditional *store-then-query* Data Base Management Systems. This need has led to a new data processing paradigm and created a new generation of data processing systems, called *Data Stream Management Systems (DSMS)* that support the execution of *continuous queries* on data streams (Terry et al. 1992).

Aurora (Carney et al. 2002), STREAM (Motwani et al. 2003), TelegraphCQ (Chandrasekaran et al. 2003), Tribeca (Sullivan 1996), Gigascope (Cranor et al. 2003), Niagara (Chen et al. 2000) and Nile (Hammad et al. 2004) are examples of current prototype DSMSs. In such systems, each monitoring application registers a set of *continuous queries* (CQs), where a CQ is continuously executed with the arrival of new relevant data (Figure 1). In the Real-time Outbreak of Disease System (RODS) example, health officials register queries for tracking specific indicators of disease outbreaks by monitoring multiple input data streams (e.g., prescription data from pharmacies). The arrival of new updates on the input data streams triggers the execution of the registered CQs. The output of such a frequent execution of a continuous query is what we call an *output data stream* (see Figure 1).

As the amount of updates on the input data streams increases and the number of registered queries becomes large, advanced query processing techniques are needed in order to efficiently synchronize the results of the continuous queries with the available updates. Efficient *scheduling* of updates is one such query processing technique which successfully improves the *Quality of Data (QoD)* provided by interactive systems.

QoD can be measured in different dimensions such as accuracy, completeness, freshness etc. (Yeganeh et al. 2009). In this paper, we focus on improving QoD in a DSMS in terms of *freshness* as we assume complete processing of stream data where the DSMS operates under a reasonable load without the need for employing load shedding or approximation techniques. As such, the DSMS provides complete and accurate results that, however, might be stale which makes freshness the main QoD dimension to consider for improvement.

Freshness is especially important, when we are interested in an accurate view of the current physical

---

The first author is supported in part by the Ontario Ministry of Research and Innovation Postdoctoral Fellowship. This work is also partially supported by NSF under project AQSIOs (IIS-0534531) and Career award (IIS-0746696).

world, be it an outbreak of a disease (as in the RODS system) or the detection of traffic patterns and congestions in an urban setting during a physical disaster. Such accurate views must reflect *all positive event "signals"* (i.e., updates) that satisfy the registered CQs.

Freshness, as well as scheduling policies for improving freshness, has been studied in contexts such as replicated databases (Cho & Garcia-Molina 2000, 2003), Web databases (Labrinidis & Roussopoulos 2001, Qu et al. 2006, Qu & Labrinidis 2007), and distributed caches (Olston & Widom 2002). To the best of our knowledge, our work is the first to study the problem of freshness in the context of data streams. In this respect, our work can be regarded as complementary to the current work on the processing of continuous queries, which considers mainly *Quality of Service (QoS)* metrics like response time and throughput such as (Carney et al. 2003, Chandrasekaran et al. 2003, Babcock et al. 2003, Sutherland et al. 2005, Bai & Zaniolo 2008, Sharaf et al. 2008).

The contributions of this paper are as follows:

1. We propose a policy for *Freshness-Aware Scheduling of Multiple Continuous Queries (FAS-MCQ)*. The proposed policy, FAS-MCQ, has the following salient features:
  - It exploits the variability of the processing costs of different continuous queries registered at the DSMS.
  - It utilizes the divergence in the arrival patterns and frequencies of updates streamed from different remote data sources.
  - It considers the impact of *selectivity* on the freshness of the output data stream. Reverting back to our RODS/event detection example, our proposed policy will favor queries that lead to positive signals instead of "blindly" processing queries that lead to negative signals.
2. Beyond the basic FAS-MCQ policy, we have also explored a *weighted version* of our FAS-MCQ scheduling policy that supports applications in which queries have different priorities. These priorities could reflect *criticality*, and hence their importance with respect to QoD captured by freshness, or *popularity*, and thus be used to optimize the overall user satisfaction.
3. We study the trade-off between scheduling CQs with the goal of improving QoD (using FAS-MCQ) as opposed to scheduling to improve QoS, which is provided by Rate-based policies such as the ones proposed in (Sharaf et al. 2008, 2006, Urhan & Franklin 2001)
4. Finally, we propose a parametrized version of our FAS-MCQ scheduler that is able to balance the trade-off between QoD and QoS according to the application's requirements.

In order to evaluate our proposed scheduling policies, we have implemented a simulator of such DSMS scheduler and ran extensive experiments. As our experimental results have shown, FAS-MCQ can improve QoD by up to 55% compared to existing scheduling policies used in DSMSs. FAS-MCQ achieves this improvement by deciding the execution order of continuous queries based on individual query properties (i.e., cost and selectivity) as well as properties of the update streams (i.e., variability of updates).

The rest of this paper is organized as follows. Section 2 provides the system model. In Section 3, we

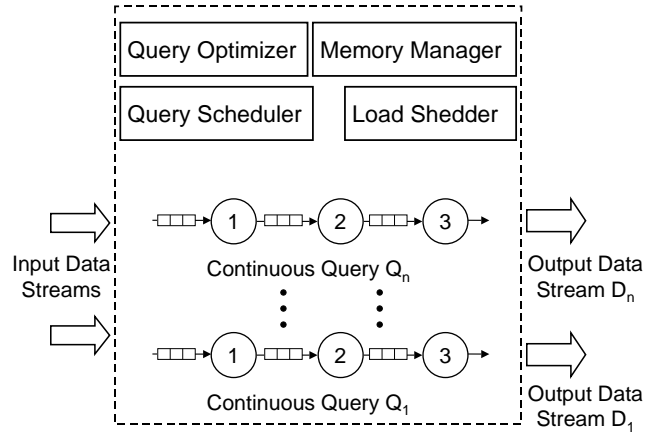


Figure 1: A DSMS hosting multiple continuous queries

define our freshness-based QoD metrics. Our proposed policies for improving freshness are presented in Section 4. In Section 5, we study the trade-off between QoD and QoS. Section 6 describes our simulation testbed, whereas Section 7 discusses our experiments and results. Section 8 surveys related work. We conclude in Section 9.

## 2 System Model

We assume a data stream management system (DSMS) where users register multiple continuous queries over multiple input data streams (as shown in Figure 1). In this section, we discuss the details of data stream processing in a DSMS, whereas in Section 3, we define our freshness-based QoD metrics on data streams.

In a DSMS, each input data stream consists of updates at remote data sources that are either continuously pushed to the DSMS or frequently pulled from the data sources. For example, sensor networks readings are continuously pushed to the DSMS, whereas updates to Web databases are frequently pulled using Web crawlers.

Each update  $u_i$  is associated with a *timestamp*  $t_i$ . This timestamp is either assigned by the data source or by the DSMS. In the former case, the timestamp reflects the time when the update took place, whereas in the latter case, it represents the arrival time of the update at the DSMS.

In this work, we assume single-stream queries where each query is defined over a single data stream. However, data streams can be shared by multiple queries, in which case each query will operate on its own copy of the data stream. Queries can also be shared among multiple users, in which case the results will be shared among them.

A single-stream query plan can be conceptualized as a data flow diagram (Carney et al. 2002, Babcock et al. 2003), i.e., as a sequence of nodes and edges, where the nodes are operators that process data and the edges represent the flow of data from one operator to another (as in Figure 1). A query  $Q$  starts at a *leaf* node and ends at a *root* node ( $O_r$ ). An edge from operator  $O_1$  to operator  $O_2$  means that the output of operator  $O_1$  is an input to operator  $O_2$ . Additionally, each operator has its own input queue where data is buffered for processing.

As a new update arrives at a query  $Q$ , it passes through the sequence of operators that compose  $Q$ . An update is processed until it either produces an output or until it is discarded by some predicate in

the query. An update produces an output only when it satisfies all the predicates in the query.

In a query, each operator  $O_x$  is associated with two values:

- *processing time or cost* ( $c_x$ ), and
- *selectivity or productivity* ( $s_x$ ).

As in traditional database systems, an operator with selectivity  $s_x$  in a DSMS produces  $s_x$  tuples after processing one tuple for  $c_x$  time units.  $s_x$  is typically less than or equal to 1 for operators like filters. Selectivity expresses the behavior or power of a filter. Additionally, for a query  $Q_i$ , we define three parameters

1. *maximum cost* ( $C_i$ ),
2. *total selectivity or total productivity* ( $S_i$ ), and
3. *average cost* ( $C_i^{avg}$ ).

For a query  $Q_i$  which is composed of a stream of operators  $\langle O_1, O_2, O_3, \dots, O_r \rangle$ , the maximum cost  $C_i$ , the total selectivity  $S_i$  and the average cost  $C_i^{avg}$  are defined as follows:

$$C_i = c_1 + c_2 + \dots + c_r$$

$$S_i = s_1 \times s_2 \times \dots \times s_r$$

$$C_i^{avg} = c_1 + c_2 s_1 + c_3 s_2 s_1 + \dots + c_r s_{r-1} \dots s_1$$

The total selectivity measures the probability that a new update will satisfy all the query predicates, while the average cost measures the expected time for processing a new update until it produces an output or until it is discarded. The average cost is computed as follows. An update starts going through the chain of operators with  $O_1$ , which has a cost of  $c_1$ . With a “probability” of  $s_1$  (equal to the selectivity of operator  $O_1$ ) the update will not be filtered out, and, as such, continue on to the next operator,  $O_2$ , which has a cost of  $c_2$ . Moving along, with a “probability” of  $s_2$  the update will not be filtered out, and, as such, continue on to the next operator,  $O_3$ , which has a cost of  $c_3$ . Up until now, on average, the cost will be  $C_i^{avg} = c_1 + c_2 s_1 + c_3 s_2 s_1$ . This is generalized in the formula for  $C_i^{avg}$  above as in (Urhan & Franklin 2001). The maximum cost is a special case of the average cost when the selectivity of each operator in the query is 1.

### 3 Freshness of Data Streams

In this section, we describe our proposed metric for measuring the quality of output data streams. Our metric is based on the *freshness* of data and is similar to the ones previously used in (Cho & Garcia-Molina 2000, Labrinidis & Roussopoulos 2001, Olston & Widom 2002, Cho & Garcia-Molina 2003, Labrinidis & Roussopoulos 2004). However, it is adapted to consider the nature of continuous queries and input/output data streams.

#### 3.1 Average Freshness for Single Streams

In a DSMS, the output of each continuous query  $Q$  is a data stream  $D$ . The arrival of new updates at the input queue of  $Q$  might lead to appending a new tuple to  $D$ . Specifically, let us assume that at time  $t$  the length of  $D$  is  $|D_t|$  and there is a single update at the input queue, also with timestamp  $t$ . Further, assume that  $Q$  finishes processing that update at time  $t'$ . At this time we distinguish two cases:

- If the tuple satisfies all the query’s predicates, then  $|D_{t'}| = |D| + 1$ . In this case, the output data stream  $D$  is considered **stale** during the interval  $[t, t']$  as the new update occurred at time  $t$  and it took until time  $t'$  to append the update to the output data stream.
- If the tuple does not satisfy all the predicates, then  $|D_{t'}| = |D|$ . In this case, the output data stream  $D$  is considered **fresh** during the interval  $[t, t']$  because the arrival of a new update has been discarded by  $Q$ . Obviously, if there is no pending update at the input queue of  $D$ , then  $D$  would also be considered fresh.

Equivalently, if we view a tuple that matches all the predicates of a query as a *positive “signal”*, then the current definition of freshness measures the amount of time that passes before the signal becomes “visible” to the end users.

Formally, to define freshness, we consider each output data stream  $D$  as an object and  $F(D, t)$  is the freshness of object  $D$  at time  $t$  which is defined as follows:

$$F(D, t) = \begin{cases} 1 & \text{if } \forall u \in I_t, \sigma(u) \text{ is false} \\ 0 & \text{if } \exists u \in I_t, \sigma(u) \text{ is true} \end{cases} \quad (1)$$

where  $I_t$  is the set of input queues in  $Q$  at time  $t$  and  $\sigma(u)$  is the result of applying  $Q$ ’s predicates on update  $u$ .

To measure the freshness of a data stream  $D$  over an entire discrete observation period from time  $T_x$  to time  $T_y$ , we have that:

$$F(D) = \frac{1}{T_y - T_x} \sum_{t=T_x}^{T_y} F(D, t) \quad (2)$$

Figure 2 shows an example of measuring the freshness of a data stream. Specifically, the figure shows two output data streams; (1) the *ideal* stream, which shows the times instants when updates became available at the DSMS; and (2) the *actual* stream, which shows the time instants when updates became available to the user. The delay between the time an update is available at the system until the time it is propagated to the user is composed of two intervals: (a) the interval where the continuous query is waiting to be scheduled for execution; and (b) the interval where the continuous query is processing the update. The sum of these two intervals represents the overall interval when the output data stream deviates from the ideal one. That is, when the output data stream is stale compared to the physical world.

In the example illustrated in Figure 2, the output data stream is stale for the intervals  $t_1$ ,  $t_2$  and  $t_3$ . Hence, the staleness of the data stream is computed as:  $(t_1 + t_2 + t_3)/(T_y - T_x)$ , equivalently, the freshness of the data stream is computed as:  $((T_y - T_x) - (t_1 + t_2 + t_3))/(T_y - T_x)$ .

#### 3.2 Average Freshness for Multiple Streams

Having measured the average freshness for single streams, we proceed to compute the average freshness over all the  $M$  data streams maintained by the DSMS. If the freshness for each stream,  $D_i$ , is given by  $F(D_i)$  using Equation 2, then the average freshness over all data streams will be:

$$F = \frac{1}{M} \sum_{i=1}^M F(D_i) \quad (3)$$

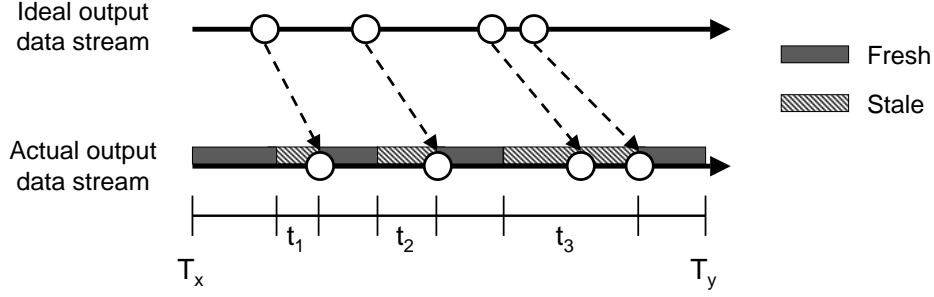


Figure 2: An example on measuring the freshness of a data stream

#### 4 Freshness-Aware Scheduling of Multiple Continuous Queries

In this section we describe our proposed policy for *Freshness-Aware Scheduling of Multiple Continuous Queries (FAS-MCQ)*.

Current work on scheduling the execution of multiple continuous queries focuses mainly on QoS metrics such as response time and throughput. Examples of such work appeared in (Chen et al. 2000, Shanmugasundaram et al. 2002, Carney et al. 2003, Chandrasekaran et al. 2003, Babcock et al. 2003, Sutherland et al. 2005, Sharaf et al. 2006, Bai & Zaniolo 2008, Sharaf et al. 2008). The scheduling policies proposed in that body of work mainly exploited the variability in the *output rate* of different CQs to improve the provided QoS. Meanwhile, previous work on synchronizing database updates exploited the variability in the *amount (frequency)* of updates to improve the provided QoD (Cho & Garcia-Molina 2000, Olston & Widom 2002, Cho & Garcia-Molina 2003).

In contrast to the work mentioned above, our proposal, *FAS-MCQ*, exploits both the variability in output rate as well as the amount of updates to improve the QoD, i.e., freshness, of output data streams.

In our previous work (Sharaf et al. 2005), we provide preliminary work on improving the freshness of data streams. In the next sections, we provide detailed analysis of our approach as well as extensions for handling multi-class continuous queries (Section 4.4) and for efficient implementation (Section 4.5). Additionally, in Section 5, we address the problem of balancing the trade-off between scheduling with the goal of improving QoD vs. QoS.

##### 4.1 Scheduling without Selectivity

In order to explain the intuition underlying our FAS-MCQ scheduler, let us assume two queries  $Q_1$  and  $Q_2$ , with output data streams  $D_1$  and  $D_2$ . Each query is composed of a set of operators, each operator has a certain cost, and the selectivity of each operator is one. Hence, we can calculate for each query  $Q_i$  its maximum cost  $C_i$  as shown in Section 2. Moreover, assume that there are  $N_1$  and  $N_2$  pending updates for queries  $Q_1$  and  $Q_2$  respectively. Finally, assume that the current wait time for the update at the head of  $Q_1$ 's queue is  $W_1$ , similarly, the current wait time for the update at the head of  $Q_2$ 's queue is  $W_2$ .

In order to determine which of the two queries should be scheduled first for execution, we compare two policies  $X$  and  $Y$ :

- Under policy  $X$ , query  $Q_1$  is executed before query  $Q_2$ ,
- Under policy  $Y$ , query  $Q_2$  is executed before query  $Q_1$ .

Under policy  $X$ , where query  $Q_1$  is executed before query  $Q_2$ , the total loss in freshness,  $L_X$ , (i.e., the period of time where  $Q_1$  and  $Q_2$  are *stale*) can be computed as follows:

$$L_X = L_{X,1} + L_{X,2} \quad (4)$$

where  $L_{X,1}$  and  $L_{X,2}$  are the staleness periods experienced by  $Q_1$  and  $Q_2$  respectively. Since  $Q_1$  will remain stale until all its pending updates are processed,  $L_{X,1}$  is computed as follows:

$$L_{X,1} = W_1 + (N_1 C_1)$$

where  $W_1$  is the current loss in freshness (i.e., increase in staleness) and  $(N_1 C_1)$  is the time required to apply all the pending updates. Similarly,  $L_{X,2}$  is computed as follows:

$$L_{X,2} = (W_2 + N_1 C_1) + (N_2 C_2)$$

where  $W_2$  is the current loss in freshness plus the extra amount of time  $(N_1 C_1)$  where  $Q_2$  will be waiting for  $Q_1$  to finish execution. By substitution in Equation 4, we get

$$L_X = W_1 + (N_1 C_1) + (W_2 + N_1 C_1) + (N_2 C_2) \quad (5)$$

Similarly, under policy  $Y$ , where  $Q_2$  is scheduled before  $Q_1$ , we have that the total loss in freshness,  $L_Y$  will be:

$$L_Y = (W_1 + N_2 C_2) + (N_1 C_1) + W_2 + (N_2 C_2) \quad (6)$$

In order for  $L_X$  to be less than  $L_Y$ , the following inequality must be satisfied:

$$N_1 C_1 < N_2 C_2 \quad (7)$$

The left-hand side of Inequality 7 shows the total increase in staleness incurred by  $Q_2$  when  $Q_1$  is executed first. Similarly, the right-hand side shows the total increase in staleness incurred by  $Q_1$  when  $Q_2$  is executed first. Hence, the inequality implies that between the two queries, we start with the one that has the lower  $N_i C_i$  value. Similarly, in the general case, where there are more than 2 queries ready for execution, we start with the one that has the lowest  $N_i C_i$  value since it will have the minimum negative impact on the freshness of the other queries in the system. Finally, it is worth mentioning that minimizing the negative impact on the overall freshness was the same general criterion that we used in our prior work on scheduling updates over materialized WebViews (Labrinidis & Roussopoulos 2001).

## 4.2 Scheduling with Selectivity

Assume the same setting as in the previous section, with the only difference being that the total productivity of each query  $Q_i$  is  $S_i \in [0, 1]$ , which is computed as in Section 2. The objective when scheduling with selectivity is the same as before: we want to minimize the total staleness. Recall from Inequality 7 that the objective of minimizing the total loss is equivalent to selecting for execution the query that minimizes the loss in freshness incurred by other queries in the system.

In the presence of selectivity, we will apply the same principle above. Towards this, we first need to compute for each output data stream  $D_i$  its *staleness probability* ( $P_i$ ) given the current status of the input data stream. This is equivalent to computing the probability that at least one of the pending updates will satisfy all of  $Q_i$ 's predicates. If  $S_i$  is the total selectivity of  $Q_i$ , then  $(1 - S_i)^{N_i}$  is the probability that all pending updates do not satisfy  $Q_i$ 's predicates, and hence  $P_i = 1 - (1 - S_i)^{N_i}$  is the staleness probability for  $Q_i$ .

If out of two queries  $Q_1$  and  $Q_2$ ,  $Q_2$  is executed before  $Q_1$ , then the expected loss in freshness incurred by  $Q_1$  due only to the impact of processing  $Q_2$  first will be:

$$L_{Q_1} = P_1 N_2 C_2^{avg} \quad (8)$$

where  $N_2 C_2^{avg}$  is the expected time that  $Q_1$  will be waiting for  $Q_2$  to finish execution and  $P_1$  is the probability that  $D_1$  is stale in the first place. For example, in the extreme case of  $S_1 = 0$ , if  $Q_2$  is executed before  $Q_1$ , it will not increase the staleness of  $D_1$  since all the updates will not satisfy  $Q_1$ . However, at  $S_1 = 1$ , if  $Q_2$  is executed before  $Q_1$ , then the staleness of  $D_1$  will increase by  $N_2 C_2^{avg}$  with probability one.

Similarly, if  $Q_1$  is executed before  $Q_2$ , then the expected loss in freshness incurred by  $Q_2$  only due to processing  $Q_1$  first is computed as:

$$L_{Q_2} = P_2 N_1 C_1^{avg} \quad (9)$$

In order for  $L_{Q_2}$  to be less than  $L_{Q_1}$ , then the following inequality must be satisfied:

$$\frac{N_1 C_1^{avg}}{P_1} < \frac{N_2 C_2^{avg}}{P_2} \quad (10)$$

Thus, in our proposed policy, each query  $Q_i$  is assigned a priority value  $V_i$  which is the product of its staleness probability and the inverse of the product of its expected cost and the number of its pending updates. Formally,

$$V_i = \frac{1 - (1 - S_i)^{N_i}}{N_i C_i^{avg}} \quad (11)$$

## 4.3 The FAS-MCQ Policy

Our proposed policy for *Freshness-Aware Scheduling for Multiple Continuous Queries (FAS-MCQ)* uses the priority function of Equation 11 to determine the scheduling order of different queries. Under this priority function *FAS-MCQ* behaves as follows:

1. If all queries have the same number of pending tuples and the same selectivity, then *FAS-MCQ* selects for execution the query with the lowest cost.
2. If all queries have the same cost and the same selectivity, then *FAS-MCQ* selects for execution the query with less pending tuples.

3. If all queries have the same cost and the same number of pending tuples, then *FAS-MCQ* selects for execution the query with high staleness probability.

In case (1), *FAS-MCQ* behaves like the *Shortest Remaining Processing Time* policy. In case (2), *FAS-MCQ* gives lower priority to the query with high frequency of updates. The intuition is that when the frequency of updates is high, it will take a long time to establish the freshness of the output data stream. This will block other queries from executing and will increase the staleness of their output data streams. In case (3), *FAS-MCQ* gives lower priority to queries with low selectivity as there is a low probability that the pending updates will “survive” the filtering of the query operators and thus be appended to the output data stream.

## 4.4 Weighted Freshness

In many monitoring applications, some queries are more important than others. That is especially obvious in emergency systems where a few continuous queries can be more critical than others. For example, under the RODS system that monitors for disease outbreaks, it is crucial to monitor for signs of water-borne diseases in areas affected by Hurricane Katrina (and thus consider the corresponding query more crucial than the rest), whereas in other areas of the world it may be more important to monitor for signs of the avian flu. In cases like these, when the system is loaded, it is necessary to maximize the freshness of these critical queries.

Towards handling multi-class CQs, we modify our proposed *FAS-MCQ* policy to increase the freshness of data streams which have higher levels of importance. Specifically, we assign each continuous query  $Q_i$  a *weight*  $\alpha_i$ . This assigned weight represents the importance of the query and it takes values in the range  $(0.0, 1.0]$  where the weight 1.0 is assigned to the most important query. Hence, the objective of our policy would be to maximize the overall *weighted freshness*. A priority function that allows us to maximize the weighted freshness can be easily deduced from Equations 8 and 9. Recall that Equation 8 measures the expected loss in freshness experienced by  $Q_1$  due to executing  $Q_2$  first, thus, the expected loss in weighted freshness experienced by  $Q_1$  is measured as:

$$WL_{Q_1} = \alpha_1 P_1 N_2 C_2^{avg}$$

Similarly, the expected loss in weighted freshness experienced by  $Q_2$ , when  $Q_1$  is executed first, is measured as:

$$WL_{Q_2} = \alpha_2 P_2 N_1 C_1^{avg}$$

In order for  $WL_{Q_2}$  to be less than  $WL_{Q_1}$ , the following inequality must be satisfied:

$$\frac{N_1 C_1^{avg}}{P_1 \alpha_1} < \frac{N_2 C_2^{avg}}{P_2 \alpha_2}$$

Then, the priority assigned to each query is computed as:

$$V_i = \frac{\alpha_i (1 - (1 - S_i)^{N_i})}{N_i C_i^{avg}} \quad (12)$$

The weights of the queries can be explicitly or implicitly defined, depending on the application. For example, in the case of an application that includes queries that are critical, the critical queries can be explicitly assigned higher weights than the rest of the queries. In applications where explicit criticality/importance information is not given, an implicit

measure of importance can be derived. For example, the popularity of each query (i.e., the number of users that registered that query) can be used as the weight. In such an application, the weighted FAS-MCQ policy will provide high levels of overall user satisfaction in terms of QoD (freshness). Finally, it is worth mentioning that the weight given to a query can be dynamic; for example, it can change depending on the time of day or the day of the week (e.g., for traffic management queries).

#### 4.5 Implementing the FAS-MCQ Scheduler

The FAS-MCQ Scheduler is invoked at every *scheduling point* and uses the current values for  $N_i$  and  $S_i$  to compute the priority of each query  $Q_i$ , according to Equation 11. In our implementation of the FAS-MCQ policy, a *scheduling point* is reached when a query finishes execution. In order to keep the scheduling overhead low when computing priorities, we use a *Calendar Queue* (Brown 1988) for priority management. Calendar queues have been widely used for implementing priority-based scheduling algorithms in high-speed networks as well as in the Aurora DSMS (Carney et al. 2003).

A calendar queue is an  $O(1)$  priority queue, based on the idea of *Bucket Sort*. Specifically, the calendar queue is structured as buckets where each bucket corresponds to a class of priorities. To insert an element in the calendar queue, a hash function is used to map its priority to the corresponding bucket. To retrieve elements from the calendar queue, buckets are traversed in order. A calendar queue allows us to avoid re-computing the priorities of queries that received no new updates between consecutive scheduling points. Additionally, for queries with new updates, the amortized cost of updating the priority is of  $O(1)$ .

### 5 Scheduling for QoD vs. Scheduling for QoS

In this section we discuss the difference in behavior between scheduling with the goal of improving QoD as opposed to scheduling with the goal of improving QoS (i.e., when the objective is to minimize the *average response time*). We also present a parametrized version of our FAS-MCQ scheduler that balances the trade-off between both the QoD and QoS metrics.

#### 5.1 Scheduling for QoS

In traditional DBMSs, the response time of a query is defined as the amount of time from the time the query arrives at the system until the time when the last tuple of the result is produced.

The definition of response time above captures the behavior of DBMSs which respond to the arrival of queries. In contrast, DSMSs respond to the arrival of new data. Hence, in a DSMS, it is more appropriate to define response time from the perspective of data (instead of queries). Therefore, we define *tuple response time* as follows:

**Definition 1** *Tuple response time,  $T_i$ , for tuple  $i$  is  $T_i = D_i - A_i$ , where  $A_i$  is the tuple arrival time and  $D_i$  is the tuple departure time. Accordingly, the average response time for  $N$  tuples is:  $\frac{1}{N} \sum_i^N T_i$ .*

Under this definition, tuples that are filtered out during query processing do not contribute to the overall response time metric (Tian & DeWitt 2003).

The *Rate-based (RB)* policy has been shown to improve the average response time of a single multi-stream query with join operators (Urhan & Franklin 2001). In the basic RB policy, each operator path

within a query is assigned a priority that is equal to its production rate. The path with the highest priority is the one scheduled for execution.

In our previous work (Sharaf et al. 2006, 2008), we generalize the basic Rate-based strategy for scheduling multiple continuous queries with the objective of minimizing the average response time. That is, multiple continuous queries are scheduled for execution based on their output rates. In this paper, we call that extended version of RB as *Rate-based for Multiple Continuous Queries (RB-MCQ)*.

Under RB-MCQ, at each scheduling point we select for execution the CQ with the highest priority (i.e., output rate). Specifically, under *RB-MCQ*, each query  $Q_i$  has a value called the *global output rate* ( $GR_i$ ) which is defined in terms of the parameters of the CQ operators. The output rate of a query  $Q_i$ , composed of the operators  $\langle O_1, O_2, O_3, \dots, O_r \rangle$ , is basically the expected number of tuples produced per time unit due to processing one tuple by the operators along the query all the way to the root  $O_r$ . Formally,

$$GR_i = \frac{S_i}{C_i^{avg}} \quad (13)$$

or, equivalently,

$$GR_i = \frac{1 - (1 - S_i)}{C_i^{avg}} \quad (14)$$

where  $S_i$  and  $C_i^{avg}$  are the CQ's expected selectivity and expected cost as defined in Section 2.

#### 5.2 Balancing the Trade-off between QoD and QoS

In this section, we present our approach for balancing the trade-off between QoS and QoD. In particular, we propose a tunable version of our FAS-MCQ scheduler that balances the trade-off between the provided response time and data freshness in a data stream management system.

Towards tuning the trade-off in the perceived performance, we argue that the distinction between scheduling for QoD and QoS is easily identified by comparing the priority functions used by FAS-MCQ (Equation 11) versus the one used by RB-MCQ (Equation 14). Specifically, the scheduling decision made by FAS-MCQ considers three factors: (1) cost (2), selectivity, and (3) number of pending tuples, whereas RB-MCQ considers only the first two factors.

As a result, FAS-MCQ might favor a query with a relatively expensive cost and very few pending tuples as opposed to RB-MCQ which might favor an inexpensive query with a large number of pending tuples. In such case, RB-MCQ may be appending tuples faster to the output data streams (i.e., providing low response time), however, the appended tuples would be stale most of the time. On the other hand, FAS-MCQ might be relatively slower in appending tuples to the output data streams (i.e., providing high response time) yet would maintain most of those output data streams as fresh as possible.

Given the above observations, we propose a parametrized version of FAS-MCQ that balances the trade-off between the achieved QoD and QoS. We will refer to this policy as FAS-MCQ( $\beta$ ), where  $\beta$  is a parameter that specifies the weight given to the number of pending tuples (i.e.,  $N$ ) when computing the priority of a query.

Formally, under FAS-MCQ( $\beta$ ) each query  $Q_i$  is assigned a priority value  $V_i$  which is computed as follows:

$$V_i = \frac{1 - (1 - S_i)^{N_i^\beta}}{N_i^\beta C_i^{avg}} \quad (15)$$



The parameter  $\beta$  takes values in the range  $[0.0, 1.0]$  and it acts as a *knob* for shaping the system's behavior. For instance, for  $\beta = 0.0$ , FAS-MCQ(0.0) behaves like the RB-MCQ policy described above, whereas for  $\beta = 1.0$ , FAS-MCQ(1.0) reverts to the original FAS-MCQ described in Section 4. For settings where  $0.0 < \beta < 1.0$ , the system achieves the desired balance between QoS and QoS.

## 6 Evaluation Testbed

We have conducted several experiments to compare the performance of our proposed scheduling policy and its sensitivity to different parameters. Specifically, we compared the performance of our proposed *FAS-MCQ* policy to a two-level scheduling scheme from Aurora where Round Robin is used to schedule queries and pipelining is used to process updates within the query. Collectively, we refer to the Aurora scheme in our experiments as *RR*. We also included the *RB-MCQ* policy (Sharaf et al. 2006) described in Section 5 as well as a *FCFS* policy where updates are processed according to their arrival times.

**Queries:** We simulated a DSMS that hosts 250 registered continuous queries. The structure of the query is adapted from (Chen et al. 2002, Madden et al. 2002) where each query consists of three operators: two predicates and one projection.

**Real Data Streams:** We use the *LBL-PKT-4* traces from the *Internet Traffic Archive*<sup>1</sup>. The traces contain an hour's worth of all wide-area traffic between the Lawrence Berkeley Laboratory and the rest of the world. In our experiments, we use the *TCP* and *UDP* packet traces as 2 input data streams to the system where the registered queries are uniformly assigned to any of the 2 data streams.

**Synthetic Data Streams:** In this setting, we generate 10 input data streams each of length 10K tuples. Initially, we generate the updates for each stream according to a Poisson distribution, with its mean inter-arrival time set according to the simulated system utilization (or load). For a utilization of 1.0, the inter-arrival time is equal to the expected time required for executing the queries in the system, whereas for lower utilizations, the mean inter-arrival time is increased proportionally. Moreover, to stress test the system, we generate a back-log of updates where we traverse the Poisson stream and group together every 10 consecutive tuples in a burst setting the arrival time of all tuples that belong to the same burst to be equal to that of the first tuple in the burst. In the default setting, 5 out of the 10 data streams are bursty.

**Selectivities:** In any query, the selectivity of the projection is set to 1, while the two predicates have the same value for selectivity, which is selected using a Zipf distribution from the range  $[0.1, 1.0]$ . The Zipf distribution is defined using a Zipf parameter which determines the degree of skewness. In our setting, the skewness is toward queries with selectivity equal to 1.0 and in the default setting the Zipf parameter is set to 0.0 (i.e., uniform distribution).

**Costs:** All operators that belong to the same query have the same cost, which is uniformly selected from three possible classes of costs. The cost of an operator in class  $i$  is equal to:  $K \times 2^i$  time units, where  $i \in [0-2]$  and  $K$  is the *scaling factor* which is used to scale the costs of operators to meet the desired utilization. For synthesized data,  $K$  is equal to 1. For the network traces, we measure the inter-arrival time of the data trace, then we set  $K$  so that the ratio between the total expected costs of queries

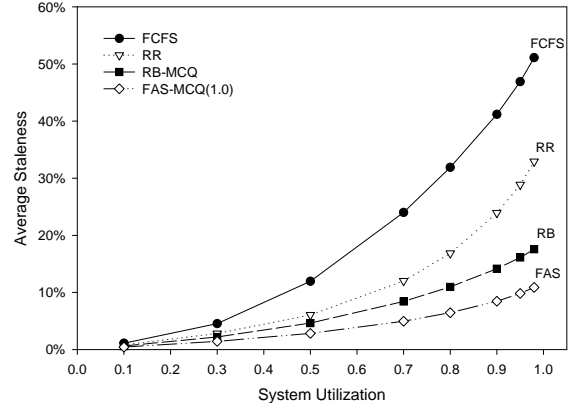


Figure 3: Average staleness vs. system utilization

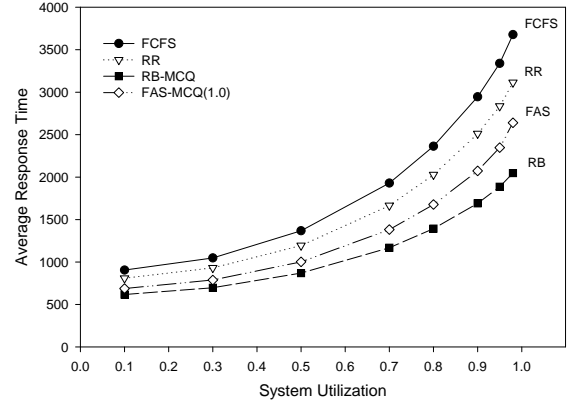


Figure 4: Average response time vs. system utilization

and the inter-arrival time is equal to the simulated utilization. Finally, the cost of each of the calendar queue operations is equal to the cost of the cheapest operator in the system.

Table 1 summarizes our simulation parameters and settings.

## 7 Experiments

In this section, we present a representative sample of the experimental results obtained under the settings described in Section 6.

### 7.1 Impact of Utilization

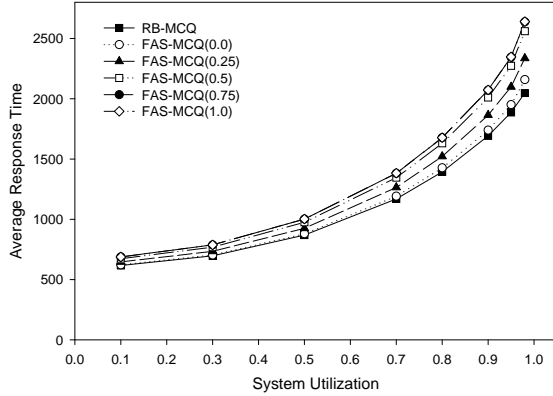
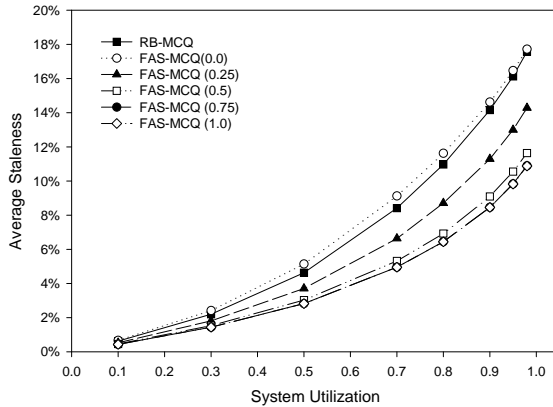
Figure 3 depicts the average total staleness over all output data streams as the utilization of the DSMS increases. In this setting, we use the basic version of FAS-MCQ which is equivalent to setting  $\beta$  to 1.0 in FAS-MCQ( $\beta$ ). The figure shows that, in general, the staleness of the output data streams increases with increasing load. It also shows that the FAS-MCQ policy provides the lowest staleness for all values of utilization with RB-MCQ being the closest contender. Additionally, the relative improvement provided by FAS-MCQ compared to RB-MCQ increases with increasing utilization. For instance, at 0.1 utilization, FAS-MCQ achieves 30% reduction in staleness compared to RB-MCQ, whereas at 0.95 utilization, RB-MCQ provides a 16% staleness while FAS-MCQ reduces the staleness to 10% (i.e., a 40% improvement).

As expected, the reduction in staleness provided by FAS-MCQ comes at the expense of an increase in response time which is illustrated in Figure 4. The figure shows that, like staleness, the response time increases with increasing load. Moreover, it shows

<sup>1</sup><http://ita.ee.lbl.gov/html/contrib/LBL-PKT.html>

Parameter	Value
Policies	FAS-MCQ, RB-MCQ, RR, FCFS
Number of Queries	250
Number of Operators per Query	3
Operators' Costs	1K, 2K, 4K
Operators' Selectivities	0.1–1.0
Utilization	0.1–0.99
Data Streams	real and synthetic
Number of Data Streams	2 (real) and 10 (synthetic)
Number of Bursty Streams	0–10

Table 1: Simulation Parameters

Figure 5: Response time for different  $\beta$ sFigure 6: Staleness for different  $\beta$ s

that RB-MCQ reduces the response time compared to FAS-MCQ. For example, at 0.95 utilization, the response time provided by FAS-MCQ is 23% higher than that of RB-MCQ (at a 40% improvement in staleness compared to RB-MCQ as previously shown in Figure 3). The trade-off between QoD (i.e., freshness) and QoS (i.e., response time) is further illustrated using Figures 5 and 6 as explained next.

## 7.2 Staleness vs. Response Time

Figures 5 and 6 show the average staleness and average response time for the same simulation settings used in the previous experiment. In addition to illustrating the difference in behavior between RB-MCQ and FAS-MCQ, the figures also show the performance of the parametrized FAS-MCQ( $\beta$ ) policy. Figure 5 shows how the response time of FAS-MCQ decreases by decreasing the value of  $\beta$  down to  $\beta = 0.0$ . Notice that at  $\beta = 0.0$ , the response time of FAS-MCQ(0.0) is just slightly higher than RB-MCQ which is due to the scheduling overheads. On the other hand, Fig-

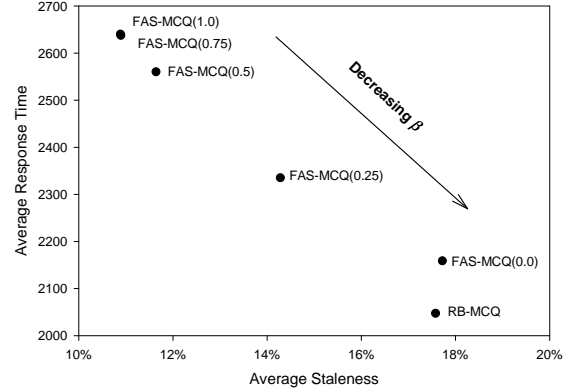


Figure 7: Trade-off between staleness and response time at utilization 0.95

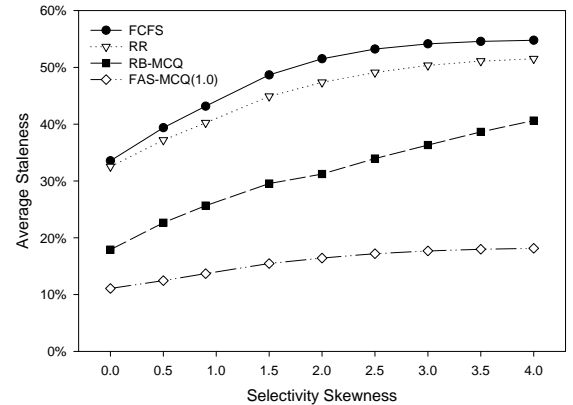


Figure 8: Staleness vs. skewness in selectivity

ure 5 shows the reduction in staleness with increasing values of  $\beta$ .

To better assess the magnitude of the trade-off, we plot the performance of the different policies at utilization 0.95 in Figure 7. For instance, the figure shows that FAS-MCQ(1.0) reduces the staleness by 40% while increasing the response time by 23%, whereas FAS-MCQ(0.25) reduces the staleness by 20% and increases the response time by 14%.

## 7.3 Impact of Selectivity

Figure 8 shows the average staleness for an experiment where all operators have the same cost, utilization is set to 95%, and the skewness of selectivity is variable. Recall that we control the degree of skewness using a Zipf parameter. Specifically, setting the Zipf parameter to 0.0 results in a uniform distribution of selectivity, whereas by the increasing its value the distribution is skewed towards high values. That is, most of the registered CQs are productive.

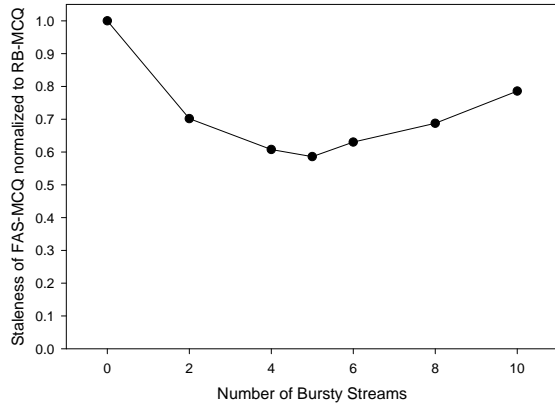


Figure 9: Staleness vs. number of bursty streams (out of 10)

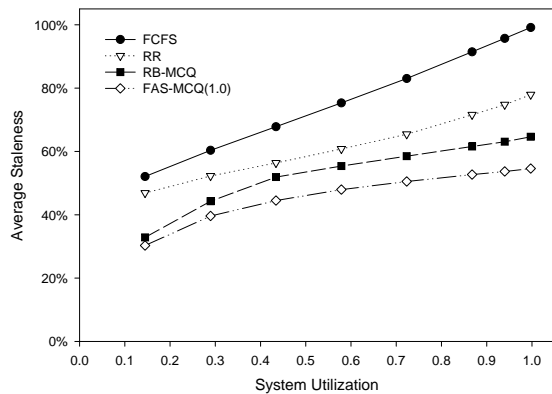


Figure 10: Staleness vs. system utilization (real data traces)

Figure 8 shows that by increasing the skewness, the staleness provided by all policies increases. This is because when most of the queries have high selectivity, then the arrival of new updates will render the output data streams stale most of the time. The figure also shows that the gains provided by FAS-MCQ compared to RB-MCQ increase with increasing the skewness. For instance, at 0.0, FAS-MCQ reduces the staleness by 40% compared to RB-MCQ. This reduction goes up to 55% when the distribution is highly skewed. The reason is that at a highly skewed distribution, all queries will have the same cost and most of them will have the same selectivity, hence, for RB-MCQ most queries will have the same priority. That is in contrast with FAS-MCQ which will utilize the extra information provided by the number of pending tuples to differentiate between queries and to assign higher priorities to queries that feed a stale stream or a stream that could be quickly brought to freshness.

#### 7.4 Impact of Bursts

The setting for this experiment is the same as the default one. The DSMS utilization, however, is set to 95% at all points. Additionally, we plot the average staleness as the number of bursty input streams increases as shown in Figure 9. For instance, at a value of 0, all the arrivals follow a Poisson distribution with no bursts, whereas at 10, all input streams are bursty.

Figure 9 shows the staleness of FAS-MCQ normalized to that of RB-MCQ. Hence, the smaller the value the bigger the reduction. The figure shows that as the number of bursty streams increases, the reduction in staleness provided by FAS-MCQ compared to RB-MCQ increases up until there are 5 bursty streams.

At that point, FAS-MCQ reduces the staleness by 40%. After that point, the performance of the two policies gets closer. The explanation is that at a moderate number of bursty streams (up to 5 streams for this setting), FAS-MCQ has a better chance to find a query with a short queue of pending updates to schedule for execution. As the number of bursty streams increases, the chance of finding such a query decreases, and as such, RB-MCQ is performing reasonably well. For instance, at 10 bursty streams, FAS-MCQ reduces the staleness by only 22% compared to RB-MCQ.

#### 7.5 Real Data

Figure 10 shows the results for our final experiment where we use real network traces. The selectivities and costs of operators are the same as in the first experiment.

In Figure 10, the behavior of the different scheduling algorithms is consistent with the previous experiments, where FAS-MCQ provides the lowest staleness followed by RB-MCQ, then RR and FCFS. Additionally, it shows the relatively high values of staleness exhibited by all policies, which is explained by the fact that the two traces are highly bursty, reflecting an ON/OFF traffic pattern.

#### 8 Related Work

Improving the QoS of multiple continuous queries has been the focus of many research efforts. For example, multi-query optimization has been exploited in (Chen et al. 2000) to improve the system throughput in an Internet environment and in (Madden et al. 2002) for improving the throughput of a data stream management system. Multi-query scheduling has been exploited by Aurora to achieve better response time or to satisfy application-specified QoS requirements (Carney et al. 2003). The work in (Babcock et al. 2003) employs a scheduler for minimizing the memory utilization. Moreover, balancing the trade-off between multiple QoS metrics has been studied in (Sutherland et al. 2005, Bai & Zaniolo 2008).

To the best of our knowledge, no previous work has proposed multi-query scheduling policies for improving the QoD provided by continuous queries. *Load shedding*, however, has been devised as a technique to control the degree of degradation in the provided QoD under overloaded conditions. Several load shedding techniques have been proposed in (Tatbul et al. 2003, Babcock et al. 2004, Tatbul & Zdonik 2006, Tatbul et al. 2007).

Scheduling policies for improving the QoD have been studied in the context of replicated databases and in web databases. For example, the work in (Cho & Garcia-Molina 2000, 2003) provides policies for crawling the Web in order to refresh a local database, where it made the observation that a data item that is updated more often should be synchronized less often. The same observation has been exploited in (Olston & Widom 2002) for refreshing distributed caches and in (Lam & Garcia-Molina 2003) for multi-casting updates. In this paper, we utilize the same observation for improving data stream freshness. Our work, however, makes the scheduling decision based on the current status of the DSMS queues (i.e., the number of pending updates) as opposed to assuming a mathematical model for updates as in (Cho & Garcia-Molina 2000, 2003).

The work in (Labrinidis & Roussopoulos 2001) studies the problem of propagating the updates to derived views. It proposes a scheduling policy for applying the updates that considers the divergence

in the computation costs of different views. Similarly, our proposed *FAS-MCQ* considers the different processing costs of the registered multiple continuous queries. Moreover, *FAS-MCQ* generalizes the work in (Labrinidis & Roussopoulos 2001) by considering updates that are streamed from multiple data sources with different traffic patterns as opposed to a single data source.

Finally, the problem of considering user preferences to manage the trade-off between QoS and QoD has been addressed in the context of web databases. The work in (Qu et al. 2006) provided an admission control framework, whereas the work in (Qu & Labrinidis 2007) introduced a two-level scheduling algorithm to address the problem.

## 9 Conclusions

Motivated by the need to support monitoring applications which involve the processing of update streams by continuous queries, in this paper we studied the different aspects that affect the QoD of these applications. In particular, we focused on the freshness of the output data stream and identified that both the properties of queries, i.e., cost and selectivity, and the properties of the input update streams, i.e., variability of updates, have a significant impact on freshness.

Our major contribution is a new approach to scheduling multiple queries in Data Stream Management Systems. Our approach exploits the properties of the continuous queries as well as the input data streams in order to maximize the freshness of output streams. We proposed a new scheduling policy called *Freshness-Aware Scheduling of Multiple Continuous Queries (FAS-MCQ)* and a weighted version of it that supports applications with multi-class queries. We also introduced a generalized variant of *FAS-MCQ* that balances the trade-off between QoD and QoS, according to application requirements. We have experimentally evaluated our proposed *FAS-MCQ* policy against scheduling policies used in current DSMS prototypes as well as Web servers. Our experiments show that *FAS-MCQ* can reduce staleness by up to 55% compared to the other policies.

## References

- Babcock, B., Babu, S., Datar, M. & Motwani, R. (2003), Chain: Operator scheduling for memory minimization in data stream systems, in 'SIGMOD'.
- Babcock, B., Datar, M. & Motwani, R. (2004), Load shedding for aggregation queries over data streams, in 'ICDE'.
- Bai, Y. & Zaniolo, C. (2008), Minimizing latency and memory in dsms: a unified approach to quasi-optimal scheduling, in 'SSPS'.
- Brown, R. (1988), 'Calendar queues: A fast  $O(1)$  priority queue implementation for the simulation event set problem', *Communications of the ACM* **31**(10), 1220–1227.
- Carney, D., Cetintemel, U., Rasin, A., Zdonik, S., Cherniack, M. & Stonebraker, M. (2003), Operator scheduling in a data stream manager, in 'VLDB'.
- Carney, D. et al. (2002), Monitoring streams: A new class of data management applications, in 'VLDB'.
- Chandrasekaran, S. et al. (2003), TelegraphCQ: Continuous Dataflow Processing for an Uncertain World, in 'CIDR'.
- Chen, J., DeWitt, D. J. & Naughton, J. F. (2002), Design and evaluation of alternative selection placement strategies in optimizing continuous queries, in 'ICDE'.
- Chen, J., DeWitt, D. J., Tian, F. & Wang, Y. (2000), NiagaraCQ: A scalable continuous query system for internet databases, in 'SIGMOD'.
- Cho, J. & Garcia-Molina, H. (2000), Synchronizing a database to improve freshness, in 'SIGMOD'.
- Cho, J. & Garcia-Molina, H. (2003), 'Effective page refresh policies for web crawlers', *ACM Transactions on Database Systems* **28**(4), 390–426.
- Cranor, C., Johnson, T., Spatschek, O. & Shkapenyuk, V. (2003), Gigascope: A stream database for network applications, in 'SIGMOD'.
- Hammad, M. et al. (2004), Nile: A query processing engine for data streams, in 'ICDE'.
- Labrinidis, A. & Roussopoulos, N. (2001), Update propagation strategies for improving the quality of data on the web, in 'VLDB'.
- Labrinidis, A. & Roussopoulos, N. (2004), 'Exploring the trade-off between performance and data freshness in database-driven web servers', *VLDB J.* **13**(3), 240–255.
- Lam, W. & Garcia-Molina, H. (2003), Multicasting a changing repository, in 'ICDE'.
- Madden, S., Shah, M. A., Hellerstein, J. M. & Raman, V. (2002), Continuously adaptive continuous queries over streams, in 'SIGMOD'.
- Motwani, R. et al. (2003), Query processing, resource management, and approximation in a data stream management system, in 'CIDR'.
- Olston, C. & Widom, J. (2002), Best-effort cache synchronization with source cooperation, in 'SIGMOD'.
- Qu, H. & Labrinidis, A. (2007), Preference-aware query and update scheduling in web-databases, in 'ICDE'.
- Qu, H., Labrinidis, A. & Mossé, D. (2006), Unit: User-centric transaction management in web-database systems, in 'ICDE'.
- Shanmugasundaram, J., Tufte, K., DeWitt, D. J., Naughton, J. F. & Maier, D. (2002), Architecting a network query engine for producing partial results, in 'WebDB'.
- Sharaf, M. A., Chrysanthos, P. K., Labrinidis, A. & Pruhs, K. (2006), Efficient scheduling of heterogeneous continuous queries, in 'VLDB'.
- Sharaf, M. A., Chrysanthos, P. K., Labrinidis, A. & Pruhs, K. (2008), 'Algorithms and metrics for processing multiple heterogeneous continuous queries', *ACM TODS* **33**(1).
- Sharaf, M. A., Labrinidis, A., Chrysanthos, P. K. & Pruhs, K. (2005), Freshness-aware scheduling of continuous queries in the dynamic web, in 'WebDB'.
- Sullivan, M. (1996), A stream database manager for network traffic analysis, in 'VLDB'.
- Sutherland, T. M., Zhu, Y., Ding, L. & Rundensteiner, E. A. (2005), An adaptive multi-objective scheduling selection framework for continuous query processing, in 'IDEAS'.
- Tatbul, N., Cetintemel, U. & Zdonik, S. B. (2007), Staying fit: Efficient load shedding techniques for distributed stream processing, in 'VLDB'.
- Tatbul, N., Cetintemel, U., Zdonik, S. B., Cherniack, M. & Stonebraker, M. (2003), Load shedding in a data stream manager, in 'VLDB'.
- Tatbul, N. & Zdonik, S. B. (2006), Window-aware load shedding for aggregation queries over data streams, in 'VLDB'.
- Terry, D. B., Goldberg, D., Nichols, D. & Oki, B. M. (1992), Continuous queries over append-only databases, in 'SIGMOD'.
- Tian, F. & DeWitt, D. J. (2003), Tuple routing strategies for distributed eddies, in 'VLDB'.
- Urhan, T. & Franklin, M. J. (2001), Dynamic pipeline scheduling for improving interactive query performance, in 'VLDB'.
- Yeganeh, N. K., Sadiq, S. W., Deng, K. & Zhou, X. (2009), Data quality aware queries in collaborative information systems, in 'APWeb/WAIM'.

# Building a Dynamic Classifier for Large Text Data Collections

Pavel Kalinov

Bela Stantic

Abdul Sattar

Institute for Integrated and Intelligent Systems

Griffith University,

Brisbane, Australia,

Email: {P.Kalinov, B.Stantic, A.Sattar} @griffith.edu.au

## Abstract

Due to the lack of in-built tools to navigate the web, people have to use external solutions to find information. The most popular of these are search engines and web directories. Search engines allow users to *locate* specific information about a particular topic, whereas web directories facilitate *exploration* over a wider topic. In the recent past, statistical machine learning methods have been successfully exploited in search engines. Web directories remained in their primitive state, which resulted in their decline. Exploration however is a task which answers a different information need of the user and should not be neglected. Web directories should provide a user experience of the same quality as search engines. Their development by machine learning methods however is hindered by the noisy nature of the web, which makes text classifiers unreliable when applied to web data. In this paper we propose Stochastic Prior Distribution Adjustment (SPDA) - a variation of the Multinomial Naïve Bayes (MNB) classifier which makes it more suitable to classify real-world data. By stochastically adjusting class prior distributions we achieve a better overall success rate, but more importantly we also significantly improve error distribution across classes, making the classifier equally reliable for all classes and therefore more usable.

**Keywords:** web directory; categorization; classification; MNB

## 1 Introduction and Motivation

Search engines are currently the main way for users to find information on the web, therefore the main source referring visitors to web sites. There are two sets of problems related to the dominant search engine model: problems for users and problems for the web itself. From the users' point of view, the main issue stems from the *keyword search paradigm*, which relies on the following assumptions:

- **Users know what they are looking for.** While this is often true, it is not the majority of cases (Yoshida et al. 2007). Many people are looking for something they have a vague idea about. This type of search for background information generates more search queries than those for specific information. Since people don't know beforehand what exactly they want, they cannot

define it by using a list of search terms so they issue *navigational queries* ((Broder 2002, Lee et al. 2004)), essentially trying to use the search engine as a web directory.

- **Users know how to find it.** This assumption also doesn't always hold. Even if users know what they want, they may not necessarily be able to formulate it in terms of keywords. It is a paradoxical situation - in order to find a document, you have to know some words it contains; in order to know them, you must have already read the document (or similar ones), i.e. you must have already found it by some other means. This reveals the underlying assumption that the user should have prior knowledge which cannot have come from the search engine. The situation was best formulated by the science fiction writer Robert Sheckley in *Ask a foolish question*: "in order to ask a question you must already know most of the answer". Additionally, there are the *synonymy* and *polysemy* issues (Deerwester et al. 1990), namely - that different people refer to the same concepts in different ways, and that the same word may mean different things to different people (or, even more confusingly, to the same person in different contexts).

Another side of the problem arises from the second aspect mentioned above - that search engines have become the main source of traffic to web sites. This has had an effect "not unlike the Heisenberg Uncertainty Principle [...] The act of Google trying to understand the web caused the web itself to change." (Zawodny 2003). This change has created an economy where a significant part of web content is now published for the benefit of search engines and not human users; content is published for the purposes of *Search Engine Optimization*, which essentially means trying to exploit search engine algorithms by providing content "they like", to the detriment of human users. Search engine spam has thus become a major source of digital garbage (Fetterly et al. 2003, Gyöngyi & Garcia-Molina 2005, Gyöngyi et al. 2006); as a consequence, a large part of development efforts are now targeted at *adversarial information retrieval* (Fetterly 2007).

The proliferation of this class of sites can be directly attributed to weaknesses of the search engine model, and an alternative model should be developed to diminish its impact.

An alternative model should also exist which does not depend on users' prior knowledge and ability to formulate their queries with specific words. Such is the web directory model which allows *discovery* of documents, as opposed to *locating* them by a search engine.

The main drawback of the existing web directory model is its reliance on manual labour for both information gathering and classification. The fact that

web directories have not been combined with web spiders has also created an issue specific to them - a problem with outdated information (sites that were listed once, then not re-checked and updated). This form of *web decay* has significant impact on directories such as Yahoo! (Bar-Yossef et al. 2004) (also confirmed by our experiments).

There are a number of machine learning methods used for text classification, which is the basis needed for automatically building a web directory. An example is the *Multinomial Naïve Bayesian* (MNB) classifier, a simple probabilistic model which classifies data instances into multiple classes using Bayesian statistics and relying on the *independent feature model* (Frank & Bouckaert 2006). However, this method works well only for simple cases, while for unbalanced classes it is not accurate. Another method of text classification is *Support Vector Machines* (SVM); Liu et al. (2005) made a large-scale study using SVM over data from the Yahoo! directory and found performance “far from satisfactory”, due to noisy and sparse data.

Statistical methods in general usually suffer from unbalanced classes and give preference to classes with more instances. This is particularly detrimental for web directories, since they are very unbalanced as content. Furthermore, an unbalanced end result is disproportionately perceived as useless by users, e.g. - if the classifier makes an error in 5% of instances, but they are all in a particular category and as a result this category remains empty, users would declare the algorithm totally unsuccessful and not just 5% unsuccessful.

In this study, we aim to build a viable web directory by improving existing classification methods, which suffer from unbalanced classes. We build on top of the MNB classifier and propose an addition to it by adding a stochastic adjustment of class prior distributions, which enables the modified algorithm to learn from its mistakes. While the classic algorithm and its many variants are static (i.e. - they will return the same values at every pass over the document collection), our method shows an improvement with successive passes over data. Furthermore, while the classic algorithm suffers from unbalanced classes and shows preference for the dominant class and very high error rates in classes with less instances, our method achieves an even distribution of the error rate, making it equally reliable for any class.

The remainder of the paper is organised as follows: in the next section we review the relevant prior work. In section 3 we outline our methodology. Section 4 describes the experiments we conducted. In section 5 we show and comment experimental results. Finally in section 6 we conclude the paper and outline the course for future work.

## 2 Relevant Work

There has been extensive research on the use of classification methods for web documents: (Lang 1995, Chakrabarti, Dom, Agrawal & Raghavan 1998, Xue et al. 2008); some systems have later been employed in real-world web directories (Attardi et al. 1999), but none persisted or are in use by major directories at present.

### 2.1 Data Processing

Text classification is usually based on the *bag of words* model (Baeza-Yates & Ribeiro-Neto 1999) where a document is considered a collection of non-dependent words (or *n*-grams) and is analyzed based on statistics such as their occurrence in documents of different

classes. Zipf’s law (according to which a large number of words appear in a text only a few times, while a few words occur orders of magnitude more often (Zipf 1949)) is ignored, which for a number of reasons does not render the algorithms ineffective.

Before starting a learning process, any algorithm needs to first obtain the data. Document text is usually parsed by a *tokenizer*, which splits it into words and normalizes them according to some rules (for example forcing lower case only, so that *Word* becomes the same as *word*). Some algorithms use these tokens directly, while others construct *n-grams* - a series of *n* tokens. This preserves some information about word order in texts, but requires much more storage and computation. Another alternative is to build Markov models; in them word order is not just preserved, but is in fact more important than the actual words used. Unfortunately, this approach provides better success rate at the cost of exponential increase in processing time (every document classified has to be matched to every existing Markov model in the training set), making it impractical to use for a large-scale classifier.

After tokenizing, text is sometimes passed through linguistic processing; for example words may be *stemmed* - words with a common root are combined into one, so that *Word* becomes the same as *word*, *wording* and *worded*.

Word counts may be scaled or normalized in some manner, for example, by their *Inverse Document Frequency*:  $IDF(w) = \ln \frac{N}{df_w}$  where *N* is the number of documents in the collection, and *df<sub>w</sub>* is the number of documents that contain the term *w*. This weight is then multiplied by the frequency of the term (TF) in the document to achieve its TF-IDF value.

A major issue with text classification is the high dimensionality of data. In a simple model, such as Naïve Bayes, every word is a *feature* itself. The typical representation of a document as a vector where each dimension is a word in the vocabulary means that the number of dimensions is equal to the number of words in the vocabulary. Reducing this number would increase the tractability of all related tasks. Therefore some other models apply feature selection or feature construction to reduce the number of features - *dimensionality reduction*, which is a form of lossy compression where precision is traded for computation costs. It is guaranteed to lose accuracy (Lang 1995), but in most cases compensates this by a great reduction in computation time.

Dimensionality reduction by feature construction can be achieved in a number of ways, such as unsupervised clustering like *k-means* performed over the dictionary of the document collection, *Latent Semantic Indexing* (LSI) (Deerwester et al. 1990), *Semantic Hashing* (Salakhutdinov & Hinton 2007) or a random projection of the high-dimension word vector onto a much lower-dimensional space (Kaski 1998).

### 2.2 Approaches to Classification

The MNB classifier is a simple probabilistic model which classifies data instances into multiple classes using Bayesian statistics and relies on the *independent feature model* - i.e., it assumes word occurrences are independent of each other. This violates Zipf’s law (actually, texts on the web were found to follow a double Pareto distribution (Chierichetti et al. 2009)) and independence is never the case, but the model works surprisingly well even though it is very inaccurate in estimating correct probabilities. As Domingos & Pazzani (1997) point out, the algorithm needs to only find out the class with highest probability and not what this probability actually is and how exactly it compares to those of the other classes. The winning

class (the one with the *maximum likelihood*) is found as:

$$\begin{aligned} \operatorname{argmax}_c p(C=c) &= \ln \frac{p(C|D)}{p(\neg C|D)} = \\ &= \ln \frac{p(C)}{p(\neg C)} + \sum_i \ln \frac{p(w_i|C)}{p(w_i|\neg C)} \end{aligned} \quad (1)$$

where  $C$  is the class the instance belongs to,  $c$  is each class,  $p(C|D)$  is the prior probability of a document for a class and  $p(w_i|C)$  is the prior probability of word  $i$  for that class,  $i$  including all words in the document.

This works for simple cases, but unbalanced classes are a serious problem: the classifier tends to favour incorrectly the larger classes since they have a high prior probability. The naïve assumption is also a problem where classes are not only unbalanced as numbers (i.e. - one class has many more instances than another) but unbalanced as content. If documents in one class are typically longer than documents in the other classes, then word occurrence rates in it would be higher and this class would be favoured incorrectly as well. As a solution, Frank & Bouckaert (2006) propose to normalize word probabilities:

$$n'_{wd} = \alpha \times \frac{n_w d}{\sum_{w'} \sum_{d \in D_c} n_{w'd}} \quad (2)$$

where  $n_{w'd}$  are class-specific word counts (occurrences of the word in documents of class  $c$ ), replacing  $w_i$  in Eq.1. They find that  $\alpha$  can be equal to 1, so in our experiments we followed that advice (see Algorithm 1).

---

**Algorithm 1:** MNB with word count normalization

---

```

Calculate prior distributions over whole
collection
Calculate normalization based on word
occurrences in each class
ClassifyQueue
foreach document do
  foreach class do
    Calculate log-likelihoods of words in
    document
    Apply normalization
    Add global log-likelihood of class
  end
  Find class with highest probability
  Announce winner class
end
```

---

Rennie et al. (2003) propose some steps to overcome systemic errors in the MNB model. They introduce Complement Naïve Bayes (CNB) to overcome skewed training due to skewed classes (training the algorithm to distinguish a class on examples *not* in that class, as opposed to the normal practice of using instances *in* that class), and weight normalization to compensate for cases where the term independence assumption of the model doesn't hold.

Rocchio's *Relevance Feedback* (Rocchio 1971) can also be used for classification by creating a representative document vector based on all documents in a class; each instance is classified into the class with which it has the smallest cosine distance.

*Support Vector Machines* (SVM) in effect build as many classifiers as there are classes, each one separating the class from all others. Each instance is evaluated by every classifier, then the decision is made

by the one with highest confidence. Liu et al. (2005) used SVM over the Yahoo! directory and found the results "far from satisfactory", due to the same problems that we faced - noise and sparse data.

All hierarchical classifiers have a common shortcoming - if they make an error on a document at one of the higher levels, this error is then propagated down the hierarchy. Xue et al. (2008) address this by dynamically training a specialized classifier for each document in the collection, using a subset of existing categories as training data. This allowed their classifier to maintain an acceptable accuracy down to Level 5 of the classification tree. They used MNB since its accuracy is comparable to that of SVM, but has lower complexity.

Most methods treat classification as a "one off" task: they take a document collection, process it and finish. For a web directory this cannot be a solution, since it updates its document collection constantly (adds, edits and removes documents from it), as well as evolves the classification structure. Furthermore, classification of documents changes with time - sometimes instances are moved from one class to another by editors not because they were wrongly classified initially, but because the perception of what they belong to may have changed (i.e. - "Hilton" used to be a hotel chain and was associated with "Business" but is now associated with "Paris" and goes to "Entertainment" or "Junk" classes). This question of ontology evolution has been studied in terms of creating different ontologies for different periods and then comparing them (Enkhsaikhan et al. 2007); however, this does not facilitate a gradually changing system such as a web directory. An online (incremental) indexing method has been proposed: (Gorrell & Webb 2005, Gorrell 2006), the results of which converge to those of LSI but process only one instance at a time, i.e. - it accounts for streaming addition of documents to the collection, but doesn't offer a solution to documents being removed from the corpus, or evolving classes. Katakis et al. (2005) propose an algorithm which is incremental in both respects - it couples an incremental feature ranking method with an incremental learning algorithm that can consider different subsets of the feature vector during prediction; for evolving classes though it would need to be retrained from scratch.

## 2.3 Approaches to Training

Some classifiers use different heuristic criteria to minimize training by selecting only a small subset of the data to train on. It is worthwhile to see what approaches spam filters use for learning, since they also use a Bayesian filter (though not multinomial). They employ several different training strategies (Zdziarski 2005):

- *TEFT* (train-everything) is the classic approach of all text classifiers, including Bayesian, and is also the most intuitive - learn from all the data. However, it is computationally expensive and, as our experiments show, does not always provide the best results. According to Zdziarski, the experience of spam filters using this approach is that it suffers from unbalanced classes (where spam is much more than non-spam) and only works well if the ratio is not worse than 70:30. This is not the case in our experiment (see data below).
- *TOE* (train-on-error) - the algorithm runs the classifier part first, then compares the result to a (manual) label for the instance and learns from it only if it has made an error (the so-called *if it isn't broken, don't fix it* philosophy). Such filters

are more “static” than TEFT filters, i.e. - they take longer to learn. On the other hand, they work better for large datasets and for highly unbalanced classes, which matches our case.

- *TUM* (train-until-mature) filters try to be the middle ground between TEFT and TOE - initially they learn on everything, then they stop learning and only retrain when they make a mistake. As with the others, they need labeled data in order to recognize a mistake, plus some heuristics to decide when a token is *mature* so as to stop learning it.
- *TUNE* (train-until-no-errors) algorithms learn until they make no mistakes, or very few mistakes. The downside is that when they start making new mistakes, they have to be retrained over the whole document corpus.

## 2.4 Alternatives

An interesting alternative to analyzing the documents themselves is the work of Chakrabarti, Dom & Indyk (1998) who propose a method to extend a human-classified directory by applying its manual labeling to *neighbouring* unlabeled sites, where they define “neighbours” in terms of outgoing links and use these links as features. The *naïve* version of their algorithm propagates classification to all unlabeled data. This was found to work well in some restricted domains, but its performance on data from the Yahoo! web directory was extremely poor which they attributed to the generally noisy nature of the web. By iterative application of the algorithm and some engineering of features, they managed to outperform classifiers based on text classification only. Gyöngyi et al. (2006–2007) developed the idea further, but found the method extremely computationally expensive which forced them to use a host-to-host connection graph instead of page-to-page, introducing significant errors. They also found that while the method works well in some restricted domains, in others it doesn’t work at all.

Some other alternative methods worth mentioning are social bookmarking (Yanbe et al. 2007) and collaborative filters. Social bookmarking relies on people labeling all documents, while collaborative filters are trained by many people then used by an individual user, such as news.google.com (Das et al. 2007). Another interesting development is WEBSOM (Lagus et al. 2004), based on a Kohonen neural network called SOM (Self-Organizing Map) (Kohonen 1995). It is not classification but a projection of data (such as a number of documents) onto a map. Similar documents end up geographically near each other on the map, facilitating browsing by analogy. The method is computationally expensive though, and the resulting map questionable in terms of usability: to achieve good resolution for browsing, the number of map units has to be proportional to the number of documents. This leads to maps with millions of points which are not easy to navigate, defeating the purpose.

## 3 Methodology

We aim to create the basis for a large hierarchical classifier which can accommodate a web directory comparable to the Open Directory Project (DMOZ - [www.dmoz.org](http://www.dmoz.org)), where we use the DMOZ entries as labeled instances for training and will later complement them with unlabeled data downloaded by a web spider. The method should take into account the

addition and deletion of documents which in a live directory combined with a web spider would be continuous, as well as occasional relabeling of instances by a human. For our tests though, we work with static data.

DMOZ has a directory tree with over 763,000 nodes, so to duplicate it we need a hierarchical classifier with as many classes. For our prototype, we built only the first level, which has 15 categories. The development of the top level of the hierarchy should equip us with the necessary techniques for all the lower levels, since by definition it is the worst case as it contains all the lower levels in itself. We decided to use a Multinomial Naïve Bayes (MNB) classifier, because it can work well incrementally, e.g. - small changes to the document collection or the reclassification of a small number of documents would not require full re-training. We tested the “classic” MNB, as well as several variations.

### 3.1 Dataset, Processing and Systemic Problems

For training/test data we used a complete data dump from DMOZ, which contained 4.16 mln web site records and 4.60 mln manual labels (some sites were classified into more than one class). We downloaded a random sample of 120 788 documents (excluding errors) from a total of 2.24 mln in the English language categories only (to minimize the dictionary).

We did not use the DMOZ descriptions for classification, but the actual text from the downloaded sites. All documents were passed through a filter which stripped HTML tags, then discarded very short and very long phrases (e.g. - shorter than 3 words and longer than 20 words) on the basis that a) one or two words are obviously not part of a sentence so are not part of any sensible text (this removes all site navigation and other clutter), while more than 20 words without punctuation indicate a machine-generated list of keywords and not a human-written text.

Words were not stemmed or pre-processed in any other manner. Arnaud (2004) attributes the poor results of a project trying to create a browsable web index based on Self Organizing Maps to the use of a bad text pre-processor. This may be a case of just selecting the wrong pre-processor; or perhaps it is a fundamental problem: pre-processing creates some bias in the data which then reflects on the ordering algorithm. Furthermore, linguistic processing has to know the structure of the language, which in our case would mean a different pre-processor for every different language sub-tree of the directory (currently numbering 81). We decided to skip it, incurring higher computational costs. We only filtered out too short and too long words (less than 3 and more than 20 characters long), and normalized word counts by TF-IDF.

Since the data is extremely high-dimensional, the usual approach would be to apply dimensionality reduction. However, we decided against this and worked with the raw data. The reasoning was that any initially successful projection of the documents into a lower-dimension representation would deteriorate as we alter the collection, since *success* is being measured by information entropy over the collection; as the collection changes, that would change too. In other words, if we achieve a set of constructed features which best split the data, this “best split” is only guaranteed to be best for the initial data. If we then remove, for the sake of argument, all documents containing the terms which constitute one of these features, we would get a feature with a probability of zero. Evolving the document collection would mean for us that a) we need to update the dictionary



constantly, b) we need to perform dimensionality reduction again and again, c) as a result, documents will have to be re-mapped to the new low-dimension vectors constantly, and d) the classifier will need to be retrained to the new features with every change. While this is not impossible to do, it adds another level of complexity to the system and doesn't seem to save too much in the way of computation, so the precision/cost trade-off doesn't seem justified. Moreover, dimensionality reduction would project the data into (typically) 64 or 16 dimensions, and since we only have 15 classes we might as well project the data into them without this intermediary. We also have to keep in mind that at lower levels of the classifier we would need separate dimensionality reduction as the document collections there would be different.

We treat the hierarchical classifier as a hierarchically ordered series of normal classifiers where the output of one classifier is the input for those at the lower level. Each classifier splits the data instances into a number of classes, which the lower-level classifiers process further using their own training data. If the higher-level classifier moves an instance from one class to another at a later point in time, this instance gets removed from the document collection of the respective lower-level classifier and put into another one. Since each lower-level classifier works with only a part of the data, it calculates all static measures (such as TF-IDF values for words) locally - i.e., taking into account only its own part of the document collection. This means there can be no global stop-words, because they are stop words in some context only. The usual example for such a word is the article "the", which is so common in the English language that it cannot be used to distinguish between different types of texts. But, it shows that the text is in English in the first place - so it is very useful to distinguish between English and Bulgarian texts, for example (i.e. - it is a valuable feature at the top level of the classifier).

We also decided to ignore *hapaxes* (Zdziarski 2005) - tokens with very low confidence, which we defined as words that occur in less than 10 documents.

Document numbers above may seem sufficient for learning but, as Liu et al. (2005) noted, data is in fact very sparse - 76% of the categories of Yahoo! Directory when they studied it contained less than 5 documents. Similarly, DMOZ has an average of 6.02 labels per category. This is due to a very fragmented categorization structure where many nodes are either too specific and have almost no content that matches them, or are placeholders (Liu et al. call them *conceptual nodes*) serving only to organize lower-level branches, with no content of their own. An omission by design (of DMOZ, Yahoo! and all similar directories) is that sites are supposed to be indexed only in one node of the directory tree - e.g., if a site is classified in the *Business: Investing: Exchanges* subcategory, it is not listed in the higher levels, i.e. *Business* and *Business: Investing*, nor in any lower levels. This makes the above situation even worse, since even high-level nodes such as *Business* are practically empty. In our implementation we did the same as Liu et al. (2005) - we "folded" high level categories by including in them the content of their sub-categories.

As regards classification duplication, the number of labels per URL in DMOZ is 1.02, as opposed to 2.23 in Yahoo!, so the level of classification noise is much lower. We have ignored this (although it harms the classifier), but in a future implementation it should be accepted as additional data and not noise, which would mean using a fuzzy classifier.

Dimensionality of data is another problem. In theory, the document collection should only contain

words from a limited dictionary (provided we only deal with documents in one language). In practice though, even when we apply ourselves to the English-language part of the directory only, there are many occurrences of names of people or places, foreign language words inserted into English texts, foreign language sites wrongly categorized in an English-language node, typos, deliberate attempts at filter poisoning (or *Bayesian poisoning* (Graham-Cumming 2006, Zdziarski 2005)) like a large collection of "words" like "btsnwgduf", bringing the overall dictionary to over half a million words for our (rather small) sample of documents. Our full dictionary had 593,718 words, 388,329 of which with a DF of one (i.e. - they were seen in only one document). For comparison, the *20 newsgroups* document collection usually used for benchmarking has a dictionary of slightly more than 61,000 words. Ignoring words that occur in less than 10 documents brought the dictionary used for training to a more manageable 62,492 words (about 2% less in the case of train-on-error variations).

The most significant problem though is that classes are highly unbalanced in a number of ways.

Firstly, they are unbalanced as number of instances they contain: the largest top-level English-language category ("Regional") contains 1.10 mln instances, while the smallest ("News") has less than 9 thousand. Thus, the dominant category has 42.4% of all instances, with the remaining instances spread in 14 categories.

Secondly, classes are unbalanced as average length of the documents they contain - for example, documents in the "Business" category of our sample had an average of 96.54 unique terms each (after filtering), while the "Adult" category (rather surprisingly) had more - 121.76, and "News" (not so surprisingly) had 235.05.

Furthermore, as mentioned above, the MNB model's assumed term independence doesn't always hold - some terms have a strong correlation, like "San" and "Francisco" in texts on American cities (Rennie et al. 2003), creating some bias in the algorithm. If this bias applied to terms evenly distributed across classes, it would just lower the success rate in general. What happens in reality though is that different classes violate the independence assumption to different degrees (the third type of unbalancing); word count normalization doesn't compensate for this, since it accounts for length of documents only and not for the actual terms they contain.

Classes are unbalanced also by the fact that the "Regional" category contains practically a bit of everything - local business, local entertainment, local news etc. so its keywords to a large extent coincide with those from all other classes, unlike for example the "Adult" or "Business" class which have more distinctive dictionaries. A high word occurrence rate for common terms, coupled with the high prior probability of the class leads to a situation where the "classic" MNB predicts the dominant class for almost every instance and has a success rate of  $\approx 1$  for that class and 0 for some others.

The combination of all these factors results in significant variation in classification success from class to class. Each of them is usually "tackled" by a separate normalization, but the interaction between them is not well studied. We decided to work from the opposite end and compensate not for each factor separately, but for their cumulative effect as measured by the class-specific error rate.

### 3.2 Algorithm Variations

The changes we introduced include:

- *Weight decay* of learned word weights: words with a word count of 1 are removed from our training data after each pass, and counts for the remaining words are divided by 2. In this way we can remove weights for words that have not been used for some time (e.g. - when the documents that contain them were removed from a node or from the collection altogether). This measure is introduced to account for a gradually changing document collection being re-classified constantly, such as a live web directory.
- We calculate *prior distribution* differently and use stochastically adjusted values for class-specific error distribution smoothing.

The second item needs explanation in more detail, since it is the core of the novelty we propose.

Prior distribution in principle should be the distribution of documents into the respective classes, where “documents” is taken to mean all documents in the collection. Spam filters though, as discussed above, employ a *train-on-error* policy where they train only on those documents they cannot classify correctly. In effect, the classifier only sees a part of the collection and bases its statistics on it only, so the prior distribution it uses is not the distribution over the whole collection but over the errors. Since classes are very unbalanced and this approach unbalances them even further, they try to compensate that by a heuristic giving a *false positive* error more weight than a *false negative* error. This heuristic is parameter-based and there is no methodology to calculate this parameter - it is based on experiments or personal preferences (where a spam filter can be tuned by a person). This parameter is a pre-set value and does not change in the life of the filter, irrespective of how it affects its efficiency or what data it processes.

Furthermore, Zdziarski's assertion that these filters are “static” means not only that they learn slowly, but also that they unlearn slowly - for example, they will recover very slowly after a Bayesian poisoning attack. In our case such attacks are not a realistic consideration, but re-classifying documents by human editors, which is a daily occurrence, has the same effect on the classifier (it has learned some classification of a number of terms, then has to unlearn it). The junk filter in the *Thunderbird 3* mail client (part of the Mozilla suite) has a partial solution to the problem in the form of weight decay (the same as we use). However, it is triggered by a rather arbitrary event - the dictionary reaching a particular size, which is a) an arbitrarily-set parameter, and b) not guaranteed to happen at all (e.g. - if the document collection is from a restricted domain with a small dictionary).

Initially we tried this approach by running the classifier in several passes over the collection with a *train-on-error* policy, triggering weight decay and re-calculating prior distributions at the end of each pass. This led to significant fluctuations in performance - success rate dropped sharply after the first update, owing to the fact that the classifier had done very well predicting instances from the most populous class, hence it had not trained on it too often and the prior distribution and word counts for this class suddenly became too low when we initialized it for the next pass. On the next update it became too dominant again, since during this pass the algorithm had made too many errors on it. The end result was a “seesaw” between these two states.

We then decided to use a constantly updated rolling count of errors, which exhibited the best results among all the algorithms we tried.

There are three sets of data used to classify an instance: word count distribution by class (document

evidence), *prior distribution* of classes and *word count normalization* by class. The first we did not try to manipulate: we used a word count of all errors (for all time), decaying after each pass over the whole collection. Changing that would require an enormous complexity of the database, since we would have to record not only word counts, but when each individual incrementation of the counter happened so that we could undo it later.

The other two measures we redefined as *based on the last 10 000 trainings of the classifier*, e.g. - we calculate class prior distribution over the last 10 000 errors, and normalize word counts based on the word counts of the last 10 000 documents on which the classifier made an error (see Algorithm 2).

---

**Algorithm 2:** MNB with distributions over last  $N$  errors

---

```

Calculate normalization based on word
occurrences in each class
ClassifyQueue
foreach document do
    Calculate prior distributions over last  $N$ 
    classification errors (from error log)
    foreach class do
        Calculate log-likelihoods of words in
        document
        Apply normalization
        Add current log-likelihood of class
    end
    Find class with highest probability
    Announce winner class
    Compare with manual label
    if winner class and manual label are
    different then
        | Log error
    end
end

```

---

In effect, we use statistics from a non-random, stochastically selected sample of the data in order to calculate the static measures needed by the classifier, making them dynamic measures.

## 4 Experimental Study

In order to test the advantages of our approach we conducted extensive experimental evaluation and compared our method with the classic MNB, as well as a *train-on-error classifier* as used by spam filters, and one with word count normalization as in (Frank & Bouckaert 2006). We tried the classifiers on the filtered text from sites we downloaded and compared results to the manual labels from DMOZ. We did a number of test runs, each with several passes over the data (results reported here are from a typical run).

All algorithms used the same filtered data, with word counts normalized by TF-IDF:

$$TF\text{-}IDF_w = \frac{n_{i,j}}{\sum_k n_{k,j}} \ln \frac{N}{df_w}$$

where  $n_{i,j}$  is the number of occurrences of the term  $t_i$  in document  $d_j$ , the denominator is the sum of the number of occurrences of all terms in document  $d_j$ ,  $N$  is the number of documents in the collection and  $df_w$  is the number of documents that contain the term  $w$ .

### 4.1 Query Set

We designed the experiments with view to compare our stochastic approach to other variations of the MNB classifier in terms of:

- Overall accuracy (success rate of the classifier)

- Class-specific accuracy (separate success rate for each class)
- Computation cost for classification
- Computation cost for training
- Storage requirements for training data

A benchmarking mechanism was implemented to record classification results for each algorithm, classification and training times (in microseconds) and dictionary size of each algorithm's training set. To avoid bias from a fixed ordering of algorithms (file cache, database cache etc.) they were called in random order when classifying each instance. Preparatory operations (reading the document, fetching IDF values etc.) were common for all algorithms and were not taken into account.

## 4.2 Noise in the Data

The first issue we faced was the significant level of noise of several types in the data, each of which has different impact and has to be dealt with differently:

- Dead links - more than 8% of listed sites returned a 404 HTTP response code (*Document not found* error) on download, or were unavailable (connection timeout). These we removed automatically.
- Web decay - sites that have been removed but generate *soft 404 errors* and now serve other content, not what was categorized by DMOZ. They do not issue an error response code and are the most difficult to remove, as they require a human's decision for each instance. They harm the classifier by making it learn wrong classifications.
- Intentional noise - legitimately-looking sites that embed noise within their own code (comment spam, hidden text due to using *search engine optimization* techniques etc.). Same as above, only more difficult to spot and remove.
- Entry pages - sites that have only a logo and an "Enter" link on their homepage which was indexed by DMOZ, but has no usable text to classify. They add noise to the classifier's training phase and reduce its success rate at the classification phase significantly.
- Classification duplication - although contrary to DMOZ policy, many sites have been indexed in more than one category.
- Wrong classification - some sites have been classified into a category not suitable for them. This one is highly subjective and the initial intuition was to just ignore it, but it turned out to present the most serious problem of all since it is the reason for the dominance of the "Regional" category (which category, in our opinion, should not exist at all - it should be a separate hierarchy).

## 4.3 Variations Tested

We tested the following variations of the classifier:

- **MNB** Multinomial Naïve Bayesian classification by the classic formula.  
Classifier: MNB as in Equation 1.  
Training: save data for all words for all documents.

- **MNB-TOE** Multinomial Naïve Bayesian classification by the classic formula, with *train on error* policy.

Classifier: MNB as in Equation 1, using prior distribution over the whole collection.

Training: save data only for those documents which the classifier did not classify correctly.

- **MNB-WN** Multinomial Naïve Bayesian classification by the classic formula, with normalized word counts.

Classifier: As Equation 1, but word counts are additionally normalized as per Equation 2 - see Algorithm 1.

Training: save data for all words for all documents.

- **MNB-SPDA** Multinomial Naïve Bayesian classification with *train on error* policy, using a prior distribution over the last  $N$  errors as in Algorithm 2. We shall call this "MNB with Stochastic Prior Distribution Adjustment".

Classifier: As Equation 1, but word counts are additionally normalized as per Equation 2. Prior distribution is calculated over the last  $N$  documents on which the classifier made an error ( $N = 10000$  for our experiment). Word counts are normalized over the whole collection.

Training: save data only for those documents which the classifier did not classify correctly.

- **MNB-SPDWNAC** Multinomial Naïve Bayesian classification with *train on error* policy, using a stochastic prior distribution and with word count normalization as per Equation 2, both calculated over the last  $N$  errors as in Algorithm 2. We shall call this "MNB with Stochastic Prior Distribution and Word Normalization Adjustment - Corrected".

Classifier: As Equation 1, but word counts are additionally normalized as per Equation 2. Both prior distribution and word normalization are calculated over the last  $N$  documents on which the classifier made an error ( $N = 10000$  for our experiment) in 80% of cases. *Corrected* stands for an addition we made which was necessitated by some problems explained below; the correction was that for the other 20% of classified instances we applied usual word normalization and not the stochastic variant.

Training: save data only for those documents which the classifier did not classify correctly.

For all algorithms, we applied weight decay at the end of each pass.

## 5 Results and Analysis

In this section we provide results obtained from our experiments and some analysis of our findings.

### 5.1 Success Rates

Apart from our two variations using a dynamic sample of the last 10,000 unsuccessfully classified documents, all algorithms are static - i.e., they use the whole document collection and produce the same results at each pass (variation between passes is within 0.01% due to randomized order of testing). Our initial intuition was that our variations would continuously improve, since the document sample they use is stochastically built and should work towards minimizing errors. This does not happen: after 3 or 4 passes

they converge to a success rate which does not change significantly thereafter. Variation between passes is within 2.5% for MNB-SPDA - it consistently beats MNB-WN, but with a varying margin (the one reported here is an average-to-low value).

On further reflection, we saw the obvious - the stochastic adjustment does not work towards minimizing the error itself, but only minimizes its variance across classes. Overall success rates can be seen in *Table 1*.

You will note that we have not quoted MNB-SPDWNA results apart from its overall performance. It turned out that MNB-SPDWNA has an important drawback: the value for normalized class word counts is in the denominator part of Equation 2. If, for any reason, the algorithm makes no errors or very few errors in one class, the values for it become too low. Since we are dividing by them, we increase its score and the algorithm starts predicting this class for all instances.

The consequences are illustrated in Fig. 1, which shows the result of a bad initialization in a test run. We did not randomize the training sequence well enough and one of the classes was not represented in the first 100,000 trained instances. The algorithm made more than 10,000 errors in that time, which resulted in an adjusted prior distribution where this class was not present. Its word normalization values then became extremely large. Interestingly, for the next 30,000 instances the algorithm actually exhibited a better success rate than before, then it became drastically unsuccessful. The initial improvement was due to weight decay being applied, which also made the respective word counts much lower and lessened the effect of the wrong normalization. This also happened at the end of the next pass when weight decay was applied again, but to a lesser extent since weights were large already. It then converged to a somewhat higher (but still much worse than normal) success rate and remained there after successive passes, i.e. - it did not recover from the error. Unlike it, the MNB-WN variant suffered a short and sharp drop in performance which it quickly overcame with increasing word counts as more documents were classified. Later applications of weight decay, as expected, did not affect it.

We got similar outcomes on a number of other runs even with correct initialization, meaning that the MNB-SPDWNA (not corrected) algorithm in its pure form is too unreliable. There are two stable states to which it can converge: the optimal state with an even error distribution, which is a form of dynamic equilibrium that needs to be maintained constantly; and the worst case, where the last  $N$  errors do not include an error in one of the classes; in this case the algorithm predicts one class for all instances, thereby making no further errors in this class and entering a positive feedback loop from which it cannot recover.

Apparently, MNB-SPDWNA needs some form of correction. We tried to do stochastic word normalization only for 80% of classified instances and full normalization for the rest, in order to give the algorithm an opportunity to occasionally make errors in all classes and introduce some negative feedback. Unfortunately, this only partly mitigates the positive feedback effect and does not make the algorithm usable.

MNB	TOE	NW	SPDA	SPDWNA
47.91%	67.86%	69.44%	70.40%	42.84%

Table 1: Success rates

It can be seen that the “classic” MNB algorithm

without any corrections also performs badly. It is interesting to note though how it fails: it’s extremely good at guessing the dominant category and extremely bad at some of the others (with literally zero success rate for the “News” and “Reference” categories, and close to zero for “Business” and “Shopping”) - see error distributions in *Table 3*. The only one performing worse was the earlier experiment we conducted with a train-on-error algorithm using a prior over errors updated once per iteration.

Train-on-error policy by itself improved the situation enormously - a jump from 47.91% to 67.86%. Word normalization as per (Frank & Bouckaert 2006) worked better - 69.44%, but the best results were exhibited by MNB-SPDA, which had a 70.40% success rate. This may not sound impressive if compared to results reported elsewhere, but we consider it a success given the enormously noisy data we worked with.

Category	All	Stochastic	Boost
Arts	9.51%	11.46%	+20.50%
Business	9.06%	9.52%	+5.08%
Computers	4.97%	5.94%	+19.52%
Games	2.20%	2.00%	-9.09%
Health	2.37%	2.74%	+15.61%
Home	1.25%	1.18%	-5.60%
News	0.25%	0.24%	-0.04%
Recreation	3.97%	4.14%	+4.28%
Reference	2.19%	2.23%	+1.83%
Regional	42.07%	35.59%	-15.40%
Science	4.61%	4.97%	+7.81%
Shopping	3.53%	4.14%	+17.28%
Society	8.60%	10.98%	+27.67%
Sports	3.85%	3.92%	+1.82%
Adult	1.58%	0.95%	-39.87%

Table 2: Stochastic adjustment to distribution

Prior distribution over the whole collection, and the values to which MNB-SPDA converged after 8 iterations, can be seen in *Table 2*. The last column provides an explanation of the improved results of the algorithm. What we see in it is an artificial “boosting” of categories with diluted content (e.g. “Society” and “Arts” which are difficult to define) where the classifier had problems, while clear-cut categories such as “Adult” and “Games” are corrected downwards since the algorithm can guess them anyway based on word distributions alone, no matter what the prior is. The “Regional” category was also adjusted downwards, for the opposite reason - its high prior probability makes it easier to guess. The end result of the adjustment is a significantly higher success rate in the problematic categories at the expense of a slight worsening of results elsewhere. This provides a much smoother spread of the error across categories: while the standard algorithms are very good in the dominant category and very poor in some of the others, MNB-SPDA has a much smaller variation in its success rates (see *Table 3*).

For the purposes of building a web directory, we think it is much more valuable to have equal treatment of all categories (i.e. - equal error levels) than higher overall accuracy. Luckily though, MNB-SPDA provides both.

If we look at the results in terms of computation costs, it also saves significantly (more than four times) on training times. A comparison of total training times (normalized) can be seen in *Table 4*.

There is also a small saving in the training set: while the *TOE* algorithms used 61,255 words, MNB-SPDA used 59,832 to train its classifier. Both of these savings - in time and in storage requirements - are due

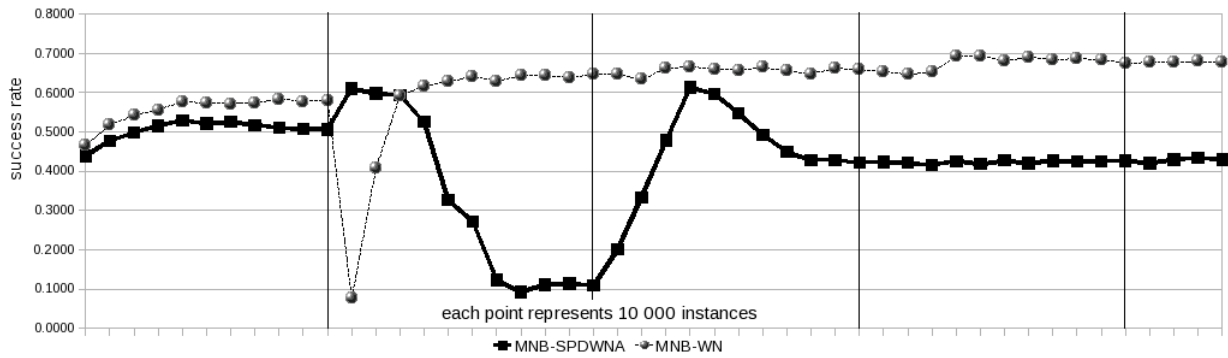


Figure 1: Consequences of bad initialization

Category	MNB	TOE	NW	SPDA
Arts	23.00%	66.27%	69.25%	65.03%
Business	00.59%	49.88%	39.92%	68.02%
Computers	13.12%	54.10%	68.06%	65.47%
Games	10.88%	49.83%	76.40%	71.36%
Health	6.12%	41.10%	61.66%	68.10%
Home	14.61%	40.90%	56.97%	73.51%
News	0.00%	0.99%	48.34%	72.19%
Recreation	0.98%	39.96%	38.31%	68.63%
Reference	0.00%	22.78%	47.00%	71.70%
Regional	99.87%	89.91%	87.29%	74.67%
Science	15.94%	55.59%	60.12%	66.80%
Shopping	0.07%	32.60%	54.71%	68.13%
Society	14.87%	62.30%	46.05%	60.82%
Sports	2.65%	50.25%	66.95%	72.25%
Adult	17.20%	54.00%	86.88%	83.22%
<b>Deviation</b>	<b>0.2394</b>	<b>0.1945</b>	<b>0.1493</b>	<b>0.0500</b>

Table 3: Class-specific success rates

MNB	TOE	NW	SPDA
4.13	1.13	4.17	1.00

Table 4: Algorithm training times

to the fact that the algorithm trains only on a subset of the documents. With a less noisy dataset, where the algorithm would make fewer errors, these savings would be much higher.

MNB	TOE	NW	SPDA
1.00	1.01	1.01	1.46

Table 5: Algorithm classification times

MNB-SPDA is heavier in classification cost though; classification times can be compared in Table 5. This is due to the fact that the algorithm performs an adjustment to its prior distributions after each training, which adds some complexity. It has to maintain a stack with the last 10,000 errors and summarize it before classifying each instance - an operation other algorithms do not need. In this area, computation can be optimized in the future.

## 6 Conclusion and Future Plans

In this work we presented the basis for a working mechanism that will make the automatic building of web directories practical. One of the variations we introduced: stochastically adjusting prior probabilities, allows the algorithm to learn from its errors and

achieve not only better overall success, but better error distribution across classes as well. This makes it equally reliable for all classes, unlike the other available methods.

The greatest challenge we faced though is not algorithm efficiency but noisy data. As often noted, the web is noisy and a significant part of the noise can be filtered out only by people. Training a successful classifier seems to need more human labeling than is freely available at this time. Our method can be used for initial training of the classifier, but then collaborative filtering based on user feedback may need to be employed for further tuning.

This study makes the following contribution to the field:

- We treat data as dynamic and achieve better classification results on this basis.
- Our method saves significantly on training time.
- Our method makes some savings in terms of storage requirements for training data.
- The classifier based on our method has smaller variation of its success rate across classes.

In our future work we intend to further enhance the ideas presented in this paper by incorporating the following: we plan to implement boosted learning which will apply the knowledge learned from labeled data to the unlabeled data obtained by a web spider; we will make our classifier fuzzy, allowing an instance to belong to many classes; we will try to optimize management of the *error stack* on which our method is based in order to make it as efficient in terms of classification times as the other algorithms; we will conduct further experiments on a larger dataset.

## References

- Arnaud, D. (2004), ‘Study of a Document Classification Framework Using Self-Organizing Maps’, [users.swing.be/aundro/websom/repository/som.pdf](http://users.swing.be/aundro/websom/repository/som.pdf).
- Attardi, G., Gulli, A., Dato, D. & Tani, C. (1999), ‘Towards automated categorization and abstracting of web sites’, [di.unipi.it/~gulli/papers/submitted/automated\\_classification.htm](http://di.unipi.it/~gulli/papers/submitted/automated_classification.htm).
- Baeza-Yates, R. & Ribeiro-Neto, B. (1999), *Modern Information Retrieval*, ACM Press Books, Harlow.
- Bar-Yossef, Z., Broder, A. Z., Kumar, R. & Tomkins, A. (2004), Sic transit gloria telae: Towards an Understanding of the Web’s Decay, in ‘Proceedings of the 13th conference on World Wide Web’, New York, NY, USA, pp. 328–337.

- Broder, A. (2002), 'A Taxonomy of Web Search', *SIGIR Forum* **36**(2), 3–10.
- Chakrabarti, S., Dom, B., Agrawal, R. & Raghavan, P. (1998), 'Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies', *The VLDB Journal* **7**(3), 163–178.
- Chakrabarti, S., Dom, B. E. & Indyk, P. (1998), Enhanced Hypertext Categorization Using Hyperlinks, in L. M. Haas & A. Tiwary, eds, 'Proceedings of SIGMOD-98, ACM International Conference on Management of Data', ACM Press, New York, US, Seattle, US, pp. 307–318.
- Chierichetti, F., Kumar, R. & Raghavan, P. (2009), Compressed Web Indexes, in '18th International World Wide Web Conference', pp. 451–451.
- Das, A., Datar, M., Garg, A. & Rajaram, S. (2007), Google News Personalization: Scalable Online Collaborative Filtering, in 'Proceedings of the Sixteenth International World Wide Web Conference (WWW2007)'.
- Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W. & Harshman, R. A. (1990), 'Indexing by Latent Semantic Analysis', *Journal of the American Society of Information Science* **41**(6), 391–407.
- Domingos, P. & Pazzani, M. J. (1997), 'On the Optimality of the Simple Bayesian Classifier under Zero-One Loss', *Machine Learning* **29**(2-3), 103–130.
- Enkhsaikhan, M., Wong, W., Liu, W. & Reynolds, M. (2007), Measuring Data-Driven Ontology Changes using Text Mining, in P. Christen, P. J. Kennedy, J. Li, I. Kolyshkina & G. J. Williams, eds, 'Sixth Australasian Data Mining Conference (AusDM 2007)', Vol. 70 of *CRPIT*, ACS, Gold Coast, Australia, pp. 39–46.
- Fetterly, D. (2007), 'Adversarial Information Retrieval: The Manipulation of Web Content', *ACM Computing Reviews*.
- Fetterly, D., Manasse, M., Najork, M. & Wiener, J. (2003), A Large-Scale Study of the Evolution of Web Pages, in 'WWW '03: Proceedings of the 12th international conference on World Wide Web', ACM, New York, NY, USA, pp. 669–678.
- Frank, E. & Bouckaert, R. R. (2006), Naïve Bayes for Text Classification with Unbalanced Classes, in 'Proc 10th European Conference on Principles and Practice of Knowledge Discovery in Databases', Berlin, Germany, Springer, pp. 503–510.
- Gorrell, G. (2006), Generalized Hebbian Algorithm for Dimensionality Reduction in Natural Language Processing, PhD thesis.
- Gorrell, G. & Webb, B. (2005), Generalized Hebbian Algorithm for Latent Semantic Analysis, in 'Proceedings of Interspeech 2005'.
- Graham-Cumming, J. (2006), 'Does Bayesian Poisoning Exist?', [virusbtn.com/spambulletin/archive/2006/02/sb200602-poison](http://virusbtn.com/spambulletin/archive/2006/02/sb200602-poison).
- Gyöngyi, Z., Berkhin, P., Garcia-Molina, H. & Pedersen, J. (2006), Link Spam Detection Based on Mass Estimation, in 'VLDB '06: Proceedings of the 32nd international conference on Very large data bases', VLDB Endowment, pp. 439–450.
- Gyöngyi, Z. & Garcia-Molina, H. (2005), 'Web Spam Taxonomy', [citeseer.ist.psu.edu/gyongyi05web.html](http://citeseer.ist.psu.edu/gyongyi05web.html).
- Gyöngyi, Z., Garcia-Molina, H. & Pedersen, J. (2006–2007), 'Web Content Categorization Using Link Information', [infolab.stanford.edu/~zoltan/publications.html](http://infolab.stanford.edu/~zoltan/publications.html).
- Kaski, S. (1998), Dimensionality Reduction by Random Mapping: Fast Similarity Computation for Clustering, in 'IJCNN98 International Joint Conference on Neural Networks', Vol. 1, Piscataway, NJ: IEEE, pp. 413–418.
- Katakis, I., Tsoumakas, G. & Vlahavas, I. (2005), 'On the Utility of Incremental Feature Selection for the Classification of Textual Data Streams', *Advances in Informatics* pp. 338–348.
- Kohonen, T. (1995), *Self-Organizing Maps*, Springer Verlag, Berlin.
- Lagus, K., Kaski, S. & Kohonen, T. (2004), 'Mining Massive Document Collections by the WEBSOM method', *Inf. Sci.* **163**(1-3), 135–156.
- Lang, K. (1995), NewsWeeder: Learning to Filter Netnews, in 'Proceedings of the 12th International Conference on Machine Learning', Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, pp. 331–339.
- Lee, U., Liu, Z. & Cho, J. (2004), Automatic Identification of User Goals in Web Search, Technical report, UCLA Computer Science.
- Liu, T., Yang, Y., Wan, H., Zeng, H., Chen, Z. & Ma, W. (2005), 'Support Vector Machines Classification with a Very Large-Scale Taxonomy', *SIGKDD Explorations* **7**, 2005.
- Rennie, J. D. M., Shih, L., Teevan, J. & Karger, D. R. (2003), Tackling the Poor Assumptions of Naïve Bayes Text Classifiers, in 'Proceedings of the Twentieth International Conference on Machine Learning', pp. 616–623.
- Rocchio, J. (1971), *Relevance Feedback in Information Retrieval*, pp. 313–323.
- Salakhutdinov, R. & Hinton, G. (2007), 'Semantic Hashing', [www.cs.utoronto.ca/~hinton/](http://www.cs.utoronto.ca/~hinton/).
- Xue, G. R., Xing, D., Yang, Q. & Yu, Y. (2008), Deep Classification in Large-Scale Text Hierarchies, in 'SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval', ACM, New York, NY, USA, pp. 619–626.
- Yanbe, Y., Jatowt, A., Nakamura, S. & Tanaka, K. (2007), Can Social Bookmarking Enhance Search in the Web?, in 'JCDL '07: Proceedings of the 2007 conference on Digital libraries', ACM Press, New York, NY, USA, pp. 107–116.
- Yoshida, T., Nakamura, S. & Tanaka, K. (2007), WeBrowSearch: Toward Web Browser with Autonomous Search, in 'WISE', pp. 135–146.
- Zawodny, J. (2003), 'PageRank is Dead', [jeremy.zawodny.com/blog/archives/000751.html](http://jeremy.zawodny.com/blog/archives/000751.html).
- Zdziarski, J. A. (2005), *Ending Spam: Bayesian Content Filtering and the Art of Statistical Language Classification*, No Starch Press.
- Zipf, G. K. (1949), *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*, Addison-Wesley.

# EP-based Robust Weighting Scheme for Fuzzy SVMs

Shaoyi Zhang<sup>1</sup>, Kotagiri Ramamohanarao<sup>1</sup> and James C. Bezdek<sup>2</sup>

<sup>1</sup>Department of Computer Science and Software Engineering, University of Melbourne, VIC 3010, Australia

<sup>2</sup>Department of Computer Science, University of West Florida, Pensacola, FL 32514, USA

{shaoyi, rao}@csse.unimelb.edu.au; jbezdek@uwf.edu

## Abstract

Support vector machine (SVM) classifiers represent one of the most powerful and promising tools for solving classification problems. In the past decade SVMs have been shown to have excellent performance in the field of data mining. The standard SVM classifier treats all instances equally. However, in many applications we have different levels of confidence in different instances that belong to a particular class. Fuzzy SVMs have been used to recognize the importance of each training instance. Although these schemes are called fuzzy SVMs, they are basically trained by weighted training instances. In this paper we propose a new robust weighting scheme for the class memberships for fuzzy SVM classifier. The weighting scheme is a sophisticated and effective method for weighting the training instances which makes use of highly discriminating patterns called emerging patterns (EPs). Our experiments show that this new weighting method has excellent performance and noise tolerance compared to the weighting scheme previously proposed.

**Keywords:** Classification, data mining, support vector machines, weighting schemes.

## 1 Introduction

The concept of *support vector machines* (SVMs) was developed by Vapnik in the early 1990s, based on *statistical learning theory* (SLT) and the principle of *structural risk minimization* (SRM) [1]. SVMs have gained wide acceptance due to their high generalization ability and better performance than many traditional learning methods over a wide range of applications. In many applications the SVM provides better generalization performance and less overfitting than other learning techniques such as *artificial neural networks* (ANNs) [1]. For example, SVMs have been effectively applied in many classification and recognition fields, such as isolated handwritten digit recognition, object recognition, speech recognition, and spatial data analysis [2].

In principal, the SVM uses a mapping  $\phi$  that transforms vectors from the original *labeled* 2-class input (object) data into vectors in a high dimensional *feature space*. In the new space it may be possible to construct a separating hyperplane between the two (imaged) classes of labeled training data. The hyperplane is then pulled back to the input space via inverse image algebra, where it becomes a

(usually) non-linear decision that separates the labeled input data. Subsequent input data are classified by their location in one of the two decision regions defined by the non-linear boundary. In practice, the input data are not mapped anywhere. Instead, inner products involving hypothetical pairs of data in the feature space are replaced by the value of a kernel function  $K$  on their “preimages”, so that  $(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) = K(\mathbf{x}_i, \mathbf{x}_j)$ . This device, universally known as the “kernel trick”, is based on a very old theorem due to Mercer [1], and renders the SVM idea feasible in practice.

The standard SVM classifier assumes that each training instance belongs unequivocally to only one class, and further, that all instances are equally important for classification. In addition, we usually assume that the class labels are accurate. But real-world data do not always belong unequivocally to one class (e.g., hybrids almost always deserve partial membership in two or more progenitor classes). Moreover, class labels are not always correct because of noise or a lack of expert knowledge. These two problems affect the optimal hyperplane obtained by an SVM. This classifier depends on only a small fraction of the instances (i.e., on the *support vectors*, SVs). So, the SVM classifier can be unduly sensitive to noise and mislabeled instances in the data [23].

In this paper, we propose a new weighting scheme for fuzzy SVM classifier that allows each training data to possess a different level of importance. We define importance of an instance by how strongly it contributes in decision making. For example, an instance that has features that strongly determine a class is generally considered more important than an instance that has weak correlation with any of the classes. We find sub-weights (class-memberships) for each training instance indicating its relationship to the two input classes. The sub-weights are then merged, becoming a single weight associated with the instance. The training data are assigned different levels of contribution towards the fuzzy SVM based on these memberships. This modification is accomplished by reformulating the constrained optimization problem upon which the fuzzy SVM classifier is based. The usual construction follows the Lagrangian to a solution for the optimal hyperplane in the primal form, found in the dual form.

We employ the recently introduced idea of *emerging patterns* (EPs) to compute sub-weights for class memberships. EPs are defined as itemsets whose supports (probabilities) increase significantly from one class to another [7, 8]. The discriminating power of EPs is in most cases proportional to their growth rates [11]. The growth rate of an EP is the ratio of its support in one class to that in the other class. EPs have had a great impact in many applications such as rare-class classification, and expansion of training data [9, 13]. Although mining EPs



(like mining association rules) is a time consuming task in data mining, they can capture significant changes between datasets [10, 12]. Hence, EPs can be used to build robust, accurate classifiers. We make use of EPs to determine the importance of each instance before building a SVM classifier. This type of SVM is as computationally efficient as the standard SVM because EPs do not play a role during decision-making.

The remainder of the paper is organized as follows. Section 2 reviews the basic theory of the standard SVM and fuzzy SVM classifiers. In section 3, we briefly review EPs. Section 4 introduces our quality weighting model based on EPs. Section 5 reports our experiments. In section 6, we compare our method with other weighting methods. Finally, we summarize our work and list some ideas for future research in section 7.

## 2 Standard & Fuzzy SVM Classifiers

In this section, we briefly introduce the theory of standard and fuzzy SVM classifiers.

### 2.1 Standard SVM Classifier

Suppose we are given a set of labeled training data

$$\{(\mathbf{x}_i, y_i) : i = 1, \dots, n\} \subset \mathcal{R}^N \times \{1, -1\} \quad (1)$$

Each input vector  $\mathbf{x}_i$  is considered to be a full member of either of two classes (called, simply, + and -), its membership indicated by the class label  $y_i \in \{-1, 1\}$  for  $i = 1, \dots, n$ . We wish to find a hyperplane

$$\mathbf{w}^T \mathbf{x}_i + b = 0 \quad (2)$$

defined by the pair  $(\mathbf{w}, b)$ , such that we can separate the points  $\mathbf{x}_i$  by the function

$$f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T \mathbf{x}_i + b) = \begin{cases} 1, & \text{if } y_i = 1; \\ -1, & \text{if } y_i = -1. \end{cases} \quad (3)$$

The set  $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  is said to be *linearly separable* if there exists  $(\mathbf{w}, b)$  such that the inequality

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \quad (4)$$

is valid for all elements of  $X$ . When  $X$  is linearly separable, we can find a unique optimal hyperplane (called the SVM) for which the margin between the projections of the training points from the two classes onto the hyperplane is maximized. If the set is not linearly separable, classification violations occur in the SVM formulation. To deal with data that are not linearly separable, the previous model is generalized by introducing  $n$  nonnegative (slack) variables  $\xi_i$  such that

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad (5)$$

where  $\xi_i \geq 0$  for those points which do not satisfy (4).

The optimal hyperplane is found as the solution to the optimization problem:

$$\text{minimize } \tau(\mathbf{w}, \xi) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i \quad (6)$$

$$\text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, n \quad (7)$$

where constant  $C$  is regarded as a regularization parameter.  $C$  is the only free parameter in the SVM classifier formulation. Tuning this parameter balances margin maximization against classifier error. We construct the Lagrangian of  $\tau$

$$L = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i (y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^n \beta_i \xi_i \quad (8)$$

Differentiating  $L$  with respect to  $\mathbf{w}$  and  $b$ , and setting the results equal to zero yields the first order necessary conditions that solutions must satisfy:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i = 0 \quad (9)$$

$$\frac{\partial L}{\partial b} = - \sum_{i=1}^n \alpha_i y_i = 0 \quad (10)$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \quad (11)$$

Solving these equations and back substituting into the original optimization problem converts it into the dual problem for data in the input space:

$$\text{maximize } Q(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \quad (12)$$

$$\text{subject to } \sum_{i=1}^n \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \dots, n \quad (13)$$

where  $\alpha = (\alpha_1, \dots, \alpha_n)$  is the vector of nonnegative Lagrange multipliers associated with the constraints. The solution for this problem satisfies

$$\alpha_i (y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i) = 0 \quad (14)$$

$$\beta_i \xi_i = 0 \quad (15)$$

If we cannot find a hyperplane that (linearly) separates the  $X$  in the input space into its two labeled classes with high classification accuracy, we consider the possibility of transforming  $X$  to a higher dimensional feature space, say  $Z = \phi(X)$ . It is our hope that the extracted data  $Z$  are linearly separable in  $\phi(\mathcal{R}^N)$ . The property of the SVM classifier that renders it feasible is that it is *not* necessary to find the nonlinear mapping  $\phi$ . Instead, we need only choose a *kernel function*  $K(\mathbf{x}_i, \mathbf{x}_j)$  that satisfies Mercer's theorem [1], for then dot products in feature space take values  $(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)) = K(\mathbf{x}_i, \mathbf{x}_j)$ . When using a kernel function, the dual problem (for vectors in feature space) becomes



$$\text{maximize } Q(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad (16)$$

$$\text{subject to } \sum_{i=1}^n \alpha_i y_i = 0, 0 \leq \alpha_i \leq C, i = 1, \dots, n \quad (17)$$

The solution also satisfies (14) and (15), and is available via quadratic programming. Two kernels are used to construct SVM and fuzzy SVM classifiers in this paper.

*Polynomial kernel:*

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + \theta)^d \quad (18)$$

*Radial-basis function (RBF) kernel:*

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) \quad (19)$$

## 2.2 Fuzzy SVM Classifier

The fuzzy SVM has proposed to extend SVM by G.C Cawley et al [26], C. Lin et al. [4] and H. Huang et al. [5]. Their schemes are identical. These authors call their models *fuzzy SVMs* because they determine the importance of each training instance from fuzzy membership values. However, their schemes still produce crisp labels, i.e., they are still crisp classifiers. In this paper we use the term “soft SVM” to avoid confusion with earlier designs. Our classifier is also crisp, but we build the fuzzy SVM classifier by replacing the fuzzy memberships of each instance with a single weight based on EPs. There are other classifiers called “fuzzy SVMs” by their authors [3, 6, 20, 21, 22], but they are not based on instance weights or fuzzy memberships. We do not discuss these alternate designs in this article, but we do compare our design to the fuzzy SVM of Lin et al. [4].

Similar to (1), suppose we are given a set of labeled training data associated with weights

$$\{(\mathbf{x}_i, y_i, \mu_i) : i = 1, \dots, n\} \subset \mathbb{R}^N \times \{1, -1\} \times [0, 1] \quad (20)$$

where  $\mu_i \in [0, 1]$  is a weight which indicates the *importance* of  $(\mathbf{x}_i, y_i)$  in the determination of a SVM classifier. Normally  $\mu_i$  is not less than  $\varepsilon$ , which is a sufficiently small positive number. We discuss in detail how to obtain reliable weights for training data in section 4.

Analogous to the standard SVM classifier, the basic idea of the fuzzy SVM classifier is to maximize the margin of separation whilst minimizing the training error, in order to achieve good generalization ability. On the other hand, unlike the SVM classifier, a fuzzy SVM classifier uses a function of the weights to reduce the effect of less important data points (i.e., increase the effect of more important points). The optimal hyperplane problem for this case, using weighted training data is the solution of the primal problem

$$\text{minimize } \tau(\mathbf{w}, \xi, \mu) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \mu_i \xi_i \quad (21)$$

$$\text{subject to } y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, i = 1, \dots, n \quad (22)$$

Note that a small  $\mu_i$  reduces the effect of the parameter  $\xi_i$  in the optimization problem. This means that the corresponding point  $(\mathbf{x}_i, y_i)$  is regarded as less important for building the optimal classifier than points with higher weight values. The Lagrangian function becomes

$$L = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \mu_i \xi_i - \sum_{i=1}^n \alpha_i (y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^n \beta_i \xi_i \quad (23)$$

By differentiating  $L$  with respect to  $\mathbf{w}$ ,  $b$  and  $\xi_i$ , and setting the results equal to zero, we obtain the first order necessary conditions for a solution:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i = 0 \quad (24)$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^n \alpha_i y_i = 0 \quad (25)$$

$$\frac{\partial L}{\partial \xi_i} = \mu_i C - \alpha_i - \beta_i = 0 \quad (26)$$

Solving these and back substituting in the usual way transform the primal optimization problem into its dual

$$\text{maximize } Q(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \quad (27)$$

$$\text{subject to } \sum_{i=1}^n \alpha_i y_i = 0, 0 \leq \alpha_i \leq \mu_i C, i = 1, \dots, n \quad (28)$$

and the solution satisfies (14) and (15).

It is clear that the only difference between the standard SVM classifier and the fuzzy SVM classifier is the upper bounds of Lagrange multipliers  $\{\alpha_i\}$  in the dual problem. In the standard SVM classifier, the  $\{\alpha_i\}$  are bounded by a single constant  $C$ , while in the weighted formulation they are bounded by dynamical values that are functions of the corresponding membership values in the fuzzy SVM classifier. The lower the membership value of a data point  $\mathbf{x}_i$  is to its own class, the narrower the feasible region is along the  $\alpha_i$  axis.

A fuzzy SVM classifier may maximize the margin like a standard SVM classifier, and correctly classify more important points (with higher weights) while preventing less important points (with lower weights, probably noise or outliers) from making the margin narrower, whether or not they are misclassified. So, different data points can have different impacts during learning of the optimal separating hyperplane. If every instance has the weight  $\mu_i=1$ , the fuzzy SVM classifier reduces to the standard SVM classifier. With different values of  $\mu_i$ , we can control the tradeoff of the corresponding training point  $\mathbf{x}_i$ . Consequently, the effectiveness of the fuzzy SVM classifier depends on the choice of the weights  $\{\mu_i\}$ . In section 4 we show how to compute the  $\{\mu_i\}$  based on *emerging patterns* (EPs).

### 3 Emerging Patterns (EPs)

For a dataset  $D$ , an instance  $I$  in  $D$  is  $I = \{(A_1=a_1), (A_2=a_2), (A_3=a_3), \dots, (A_n=a_n)\}$ , where  $a_1, a_2, a_3, \dots, a_n$  are values related to the attributes  $\{A_1, A_2, A_3, \dots, A_n\}$ . We call each pair  $(A, a)$  an *item* [8]. Let  $Z$  denote the set of all items in an encoding dataset  $D$ . Itemsets are subsets of  $Z$ . We say an instance  $Y$  contains an itemset  $X$ , if  $X \subset Y$ .

**Definition 1.** Given a dataset  $D$  and an itemset  $X$ , the support of  $X$  in  $D$ ,  $Supp_D(X)$ , is defined as

$$Supp_D(X) = \frac{count_D(X)}{|D|} \quad (29)$$

where  $count_D(X)$  is the number of instances in  $D$  containing  $X$ .

EPs are itemsets whose supports change *significantly* from one class to another [7, 13]. EPs capture sharp differences between data classes, thus affording a competitive alternative to other existing state-of-the-art classifiers [24].

**Definition 2.** Given two different datasets  $D_1$  and  $D_2$ , where instances in  $D_i$  belong to class  $C_i$ , let  $Supp_{D_i}(X)$  denote the support of the itemset  $X$  in  $D_i$ . The growth rate of  $X$  from  $D_1$  to  $D_2$ ,  $GR_{D_1 \rightarrow D_2}(X)$ , is defined as

$$GR_{D_1 \rightarrow D_2}(X) = \begin{cases} 0, & \text{if } Supp_{D_1}(X) = 0 \\ & \text{and } Supp_{D_2}(X) = 0; \\ \infty, & \text{if } Supp_{D_1}(X) = 0 \\ & \text{and } Supp_{D_2}(X) \neq 0; \\ \frac{Supp_{D_2}(X)}{Supp_{D_1}(X)}, & \text{otherwise.} \end{cases} \quad (30)$$

When  $D_1$  is clear from the context, an EP  $e$  from  $D_1$  to  $D_2$  is called an EP of  $D_2$ ; the support of  $e$  in  $D_2$ ,  $Supp_{D_2}(e)$ , is simply denoted as the support of  $e$ ,  $Supp(e)$ ; and its growth rate from  $D_1$  to  $D_2$ ,  $GR_{D_1 \rightarrow D_2}(e)$ , is denoted as the growth rate of  $e$ ,  $GR(e)$ .

Due to their high support in the home class and low support in the contrasting class, EPs can be regarded as strong signals that distinguish classes of data. Intuitively, a good instance should provide strong EPs of the same class – it should contain strong signals that are unique to this class. A bad instance (i.e., noise or outlier), however, may contain no EPs, or EPs of both contrasting classes with approximately equal strength. Thus, EPs can be used to help find the class memberships for training instances and then to build the fuzzy SVM classifier. Section 4 presents our method for constructing the fuzzy SVM.

### 4 A Weighting Method Based on EPs

We can train the fuzzy SVM classifier directly if the training data already have associated weights  $\{\mu_i \in [0, 1]\}$ . And in this case, the weights are sometimes regarded as probabilities of the instances that represent their importance or meaning confidence. However, data collected in almost all real-world applications lacks information about weights and noise. Without any, or with

little, prior information, it is very hard to generate a reliable weighting model from data and to find the true noise distribution. Therefore finding a good function to calculate the weights *from the data* is a primary concern when building a fuzzy SVM classifier. In this paper we propose a model based on using EPs.

First, we calculate a sub-weight for each class and instance, depending on the EPs contained, and then map the sub-weights of each instance into a single weight representing the importance of each point. Then we normalize these values to determine a final weight for each instance which reflects its relative importance for determining the decision surface.

#### 4.1 Using EPs to Calculate Sub-weights

We discretize continuous attributes in the training instances so that we can extract EPs. (The fuzzy SVM classifier will be built using the original training instances.) Assume that after discretization we have a set of training instances,  $D = \{I_1, I_2, I_3, \dots, I_n\}$ , and a set of classes,  $C = \{C_1, C_2, C_3, \dots, C_m\}$ . We partition  $D$  into  $m$  datasets,  $D_1, D_2, \dots, D_m$ , where  $D = D_1 \cup D_2 \cup \dots \cup D_m$ .  $E_k$  is a set of EPs extracted from the dataset related to class  $C_k$  such that the EPs in  $E_k$  have significantly higher support in  $D_k$  than in  $\bar{D}_k$ , which is the complementary set of  $D_k$ . The support of an EP  $e \in E_k$  is  $Supp_{D_k}(e)$  and the growth rate of it is  $GR_{\bar{D}_k \rightarrow D_k}(e)$ . The strength of  $e$  in class  $C_k$ ,  $Strength_k(e)$ , is defined as follows [8]:

$$Strength_k(e) = \frac{GR_{\bar{D}_k \rightarrow D_k}(e)}{GR_{\bar{D}_k \rightarrow D_k}(e) + 1} \times Supp_{D_k}(e) \quad (29)$$

$Strength_k(e)$  represents the contribution of  $e \in E_k$  in class  $C_k$ . This contribution is proportional to both the growth rate (discriminating power) of  $e$ ,  $GR_{\bar{D}_k \rightarrow D_k}(e)$ , and its support in the home class ( $C_k$ ),  $Supp_{D_k}(e)$ . An EP might have a high growth rate and low support in its home class and hence, its strength will be low. Alternatively, an EP might have a low growth rate and high support in its home class, again resulting in low strength. That is, in order for an EP to be strong, it has to have both a high growth rate and high support.

Getting all the EPs contained in an instance  $I \in D$  for class  $C_k$ , we calculate the sub-weight  $SW_k(I)$  of  $I$  for  $C_k$ , which is found by aggregating the contributions of these EPs.

$$SW_k(I) = \sum_{e \in I, e \in E_k} Strength_k(e) \quad (32)$$

Now we get the sub-weights of each instance for every class, no matter whether it is the instance's home class or not. The result of this, depicted schematically in Figure 1, is that the crisp instances are converted to weighted ones.

#### 4.2 Merging Sub-weights of Each Instance to A Total-weight

For an  $m$ -class problem, we have  $m$  sub-weights for each instance. We need a reliable way to combine them into a

	$C_1$	$C_2$	...	$C_m$
$I_1$	1	0	...	0
$I_2$	0	1	...	0
...	...	...	...	...
$I_n$	0	0	...	1

	$C_1$	$C_2$	...	$C_m$
$I_1$	$SW_1(I_1)$	$SW_2(I_1)$	...	$SW_m(I_1)$
$I_2$	$SW_1(I_2)$	$SW_2(I_2)$	...	$SW_m(I_2)$
...	...	...	...	...
$I_n$	$SW_1(I_n)$	$SW_2(I_n)$	...	$SW_m(I_n)$

**Fig. 1: Conversion of crisp instances to weighted ones.**

single total-weight for each instance to build the fuzzy SVM classifier. We can do this for each instance  $I$  in class  $C_k$  by computing a total-weight using the sub-weights as follows:

$$TW''(I) = \frac{\sum_{p=1}^{m-1} \sum_{q=p+1}^m |SW_p(I) - SW_q(I)|}{\binom{m}{2}} \quad (33)$$

The function  $TW''(I)$  in (33) uses absolute values of differences of sub-weight pairs. Although  $TW''(I)$  indicates the difference between sub-weights very well, it still has a problem, viz., the value of the total-weight for an instance will be the same, no matter which home class it belongs to. For example, two instances in class  $C_1$  with sub-weight sets (9, 1) and (1, 9) will have the same total-weight.

In real-world data there are usually noise and outliers, and some data may be misclassified. The performance of any classifier will improve if these problems are taken into account. Our method of calculating the sub-weights of classes usually results in the labeled class of a training instance having the highest sub-weight value. However, something on the order of 5%-15% of the instances in any training data seem to have a higher sub-weight in a class other than the marked class. We should not arbitrarily move such instances into the class with the highest sub-weight, but we can reduce weights of the instances whose labeled class sub-weight is much lower than the highest values. Consequently, for an instance  $I$  in home class  $C_k$ , we modify (33) by introducing the sub-weight  $SW_k(I)$  into it:

$$TW(I) = \sqrt{\frac{SW_k(I) \times \sum_{p=1}^{m-1} \sum_{q=p+1}^m |SW_p(I) - SW_q(I)|}{\binom{m}{2}}} \quad (34)$$

The square root is used to avoid “polarization” of the values of  $TW$  (some are too large and others are too close to 0).

Using this function,  $TW = \sqrt{9 \times |9 - 1| / 1} = 8.5$  for sub-weight set (9, 1) and  $TW = \sqrt{1 \times |9 - 1| / 1} = 2.8$  for sub-weight set (1, 9). Hence,  $TW$  distinguishes the two cases from each other. Moreover, if an instance is labeled as belonging to a class, and is associated with a low sub-weight for the class, it will have a low total-weight.

Alternatively, we can introduce a method similar to standard deviation (note it is *not* a standard deviation) to compute the total-weight as

$$TW(I) = \sqrt{SW_k(I) \times \frac{1}{m} \sum_{p=1}^m (SW_p(I) - \overline{SW(I)})^2} \quad (35)$$

### 4.3 Weight Normalization

Now we have a single total-weight for each instance. However, these weights cannot be directly used for training instances to build a fuzzy SVM classifier, because the number of EPs may differ significantly from one class to another. As a result, the class with the largest number of EPs will have the highest aggregated value.

To overcome this problem, the total-weights of instances in a class are normalized by the value range of the class. Having a total-weight  $TW(I)$  for training instance  $i$ , we need a normalization function that maps  $TW(I)$  from  $(-\infty, +\infty)$  to  $[0, 1]$ . Let  $TW_{\max}$  and  $TW_{\min}$  be the maximum and the minimum total-weights for a given class. We use the following mapping to get the normalized weights:

$$W(I) = \frac{TW(I) - TW_{\min}}{TW_{\max} - TW_{\min}} \quad (36)$$

where  $TW(I)$  is the final weight for training instance  $I$ . We perform this mapping class by class, thereby obtaining a normalized final weight for each instance. Figure 2 depicts the architecture of our weight assignment scheme underlying the fuzzy SVM.

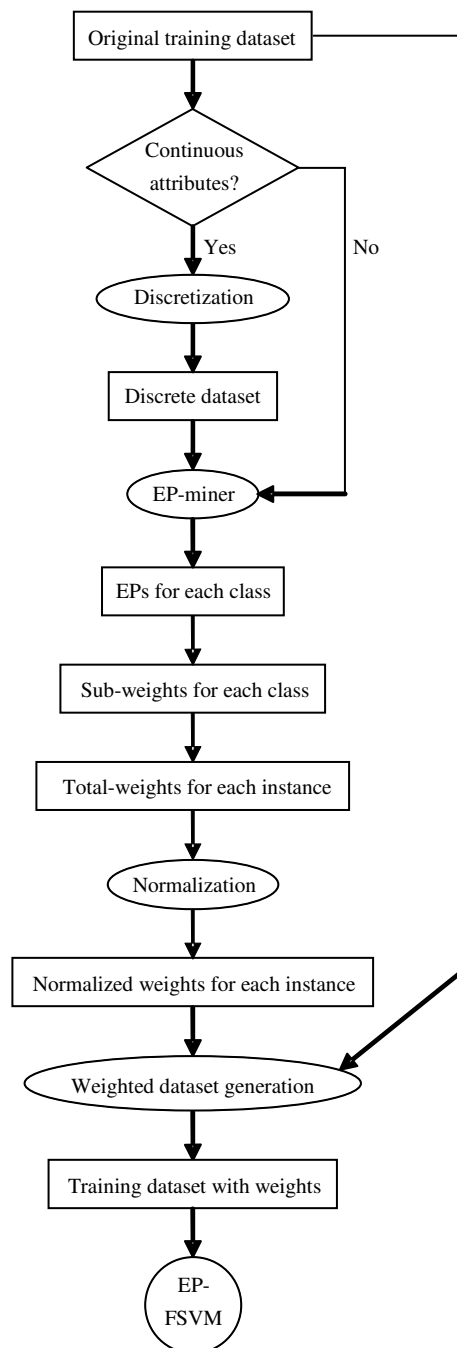
## 5 Experimental Evaluation

In order to evaluate the effectiveness of our EP-based weighting model, we carry out a number of experiments. We used 25 benchmark datasets from the UCI Machine Learning Repository [19]. Both polynomial and RBF kernels were used. We used the WEKA [17] discretization filter “weka.filters.supervised.attribute.Discretize -R first-last” for continuous attributes [25], and modified the WEKA SVM classifier to build our fuzzy SVM classifier.

Error estimates are obtained using stratified 10-fold Cross-Validation (CV-10). Results reported are the mean classification performance over the 10 folds. Here we use a range of values for hyper-parameters  $C$ ,  $d$  and  $\sigma$ , and report the best classification accuracy for each classifier.

### 5.1 EP-based Fuzzy SVM vs. Standard SVM

In Table 1 we compare the accuracy of the standard SVM classifier to our fuzzy SVM classifier with EPs-based weights calculated by (34). The comparison is effected by calculating the percent improvement ( $\%impr$ ) in accuracy ( $acc$ ) as



**Fig. 2: Procedure of building the EP-based fuzzy SVM.**

$$\%impr = \frac{acc_{w-SVM} - acc_{SVM}}{acc_{SVM}} \times 100 \quad (37)$$

Equation (37) will yield a negative % when the SVM performs better than the fuzzy SVM; conversely, a positive % indicates superior performance by the fuzzy SVM. Table 1 shows that our EP-based method to assign weights results in a fuzzy SVM classifier which almost always outperforms the standard SVM classifier using either kernel. The maximum improvement, 22.03%, is realized for the dataset Breast-c, while the maximum  $-\%impr$  is -4.17% for colic-o. Put another way, SVM is better than fuzzy SVM in only 9 out of 50 tries (18% of the tries). The parameter sets are shown in Table 2.

Dataset	Polynomial Kernel			RBF Kernel		
	SVM	FSVM	%impr	SVM	FSVM	%impr
Anneal-o	90.20	<b>98.89</b>	9.63	90.87	<b>98.23</b>	8.10
Autos	78.05	<b>81.51</b>	4.43	78.54	<b>81.85</b>	4.21
Balance-s	<b>99.36</b>	98.76	-0.60	98.72	<b>100</b>	1.30
Breast-c	69.63	<b>84.97</b>	22.03	<b>75.91</b>	75.52	-0.51
Breast-w	97.00	<b>99.42</b>	2.49	97.14	<b>99.08</b>	2.00
Colic	82.61	<b>90.07</b>	9.03	84.51	<b>85.05</b>	0.64
Colic-o	<b>78.26</b>	75.00	-4.17	78.53	<b>79.35</b>	1.04
Credit-a	85.51	<b>95.46</b>	11.64	86.96	<b>93.17</b>	7.14
Credit-g	75.70	<b>91.50</b>	20.87	78.30	<b>81.60</b>	4.21
Diabetes	77.86	<b>78.85</b>	1.27	77.73	<b>77.87</b>	0.18
Glass	74.77	<b>78.52</b>	5.02	72.90	<b>76.07</b>	4.35
Heart-c	84.49	<b>95.18</b>	12.65	86.14	<b>94.59</b>	9.81
Heart-h	84.75	<b>94.35</b>	11.33	84.05	<b>91.67</b>	9.07
Heart-s	84.44	<b>95.56</b>	13.17	84.44	<b>85.93</b>	1.76
Hepatitis	85.17	<b>93.96</b>	10.32	86.46	<b>94.43</b>	9.22
Ionosphere	<b>91.75</b>	91.45	-0.33	<b>94.89</b>	94.59	-0.32
Iris	<b>96.67</b>	<b>96.67</b>	0	<b>97.33</b>	96.00	-1.37
Kr-vs-kp	<b>99.66</b>	<b>99.66</b>	0	<b>99.72</b>	99.56	-0.16
Labor	<b>89.47</b>	87.72	-1.96	<b>92.98</b>	87.72	-5.66
Lymph	86.49	<b>87.84</b>	1.56	86.49	<b>87.84</b>	1.56
Mushroom	<b>100</b>	<b>100</b>	0	<b>100</b>	<b>100</b>	0
Sick	<b>96.66</b>	<b>96.66</b>	0	<b>97.03</b>	<b>97.03</b>	0
Sonar	84.62	<b>89.90</b>	6.24	<b>88.46</b>	87.50	-1.09
Vote	96.55	<b>97.24</b>	0.71	<b>96.55</b>	96.09	-0.48
Weather-n	<b>71.43</b>	<b>71.43</b>	0	71.43	<b>78.57</b>	10.00
Average	86.44	<b>90.82</b>	5.70	87.44	<b>89.57</b>	2.98
Best	4	<b>16</b>		7	<b>16</b>	
Wilcoxon Test (Win/Draw/Loss)						
	Polynomial Kernel			RBF Kernel		
FSVM vs. SVM	13/11/1			10/14/1		

**Table 1: %Accuracy and %improvement; SVM vs. FSVM (eq.34)**

We performed CV-10 Wilcoxon signed-rank tests (at the 0.05 level) and found that our fuzzy SVM classifier with polynomial kernels got 13 wins, 11 draws and 1 loss. And with RBF kernels, fuzzy SVM got 10 wins, 14 draws and 1 loss. By changing the constant  $C$  and CV foldnumber, we get similar results with all of the datasets.

When we use the weighting function in (35) to build the fuzzy SVM machine, the results (in Table 3) are also much better than those attained by the standard SVM. Comparing Tables 1 and 3, we see that the fuzzy SVM with weights calculated by (34) is somewhat more accurate than the fuzzy SVM machine based on (35). This is because (35) does not indicate the differences of sub-weights as well as (34). However, for some datasets with a large number of classes, (35) may be slightly more efficient than (34). Henceforth, we use only weighting scheme (34).

## 5.2 Robustness of the EP-based Fuzzy SVM Classifier

In the real world we must expect errors or noise in datasets. Therefore, robustness (or noise tolerance) is an important

Dataset	Polynomial Kernel				RBF Kernel			
	SVM		FSVM		SVM		FSVM	
	$C$	$d$	$C$	$d$	$C$	$2\sigma^2$	$C$	$2\sigma^2$
Anneal-o	5	2	10	3	1000	1	500	10
Autos	1	2	1	2	100	10	10	10
Balance-s	100	2	1000	2	1000	10	150	1
Breast-c	1	1	1	1	5	10	5	10
Breast-w	1	1	1	2	1	1	500	100
Colic	1	1	1	1	10	1000	500	1000
Colic-o	1	3	1	1	10	100	1000	1000
Credit-a	1	2	5	1	10	10	100	10
Credit-g	50	1	5	1	100	100	150	1000
Diabetes	100	1	10	1	1	1	10	1
Glass	500	3	50	2	500	1	1000	10
Heart-c	50	1	100	1	1000	100	1000	1000
Heart-h	10	1	10	1	50	10	1000	100
Heart-s	50	1	1	1	50	100	50	100
Hepatitis	1	1	1	1	100	100	150	1000
Ionosphere	10	2	1	2	10	10	10	1
Iris	5	1	500	3	100	100	5	1
Kr-vs-kp	1	3	1	3	100	10	100	10
Labor	1	2	10	1	5	10	500	10
Lymph	1	1	10	1	5	10	500	100
Mushroom	1	1	1	1	1	0.1	5	1
Sick	100	3	1000	1	150	1	1000	10
Sonar	1	3	5	1	5	1	10	1
Vote	5	1	10	1	5	10	500	100
Weather-n	10	1	1	2	5	1	5	10

Table 2: Parameter sets for models in table 1

feature of a classifier [16, 18]. Next, we investigate how well these EP-based classifiers respond to increasing noise in data, and compare the robustness of our fuzzy SVM classifier to that of the standard SVM classifier. To simulate the effect of noise, we replace the attribute values of all training instances as follows:

$$av' = av \times (1 + r \times \lambda) \quad (38)$$

where  $av$  and  $av'$  are the original and new attribute values;  $r$  is a random value in the range  $[-1, 1]$  and  $\lambda$  is the percentage of noise. We leave the testing data intact. Using these training datasets with noise, we build the standard SVM and fuzzy SVM classifiers and compare their performance.

Here, we use model difference to evaluate the noise tolerance of classifiers. The model difference between two models  $M_1$  and  $M_2$ ,  $MD(M_1, M_2)$ , is:

$$MD(M_1, M_2) = \frac{Ins\_d(M_1, M_2)}{Ins\_all} \times 100\% \quad (39)$$

where  $Ins\_d(M_1, M_2)$  is the number of instances models  $M_1$  and  $M_2$  label differently, and  $Ins\_all$  is the number of instances in the test set. The model difference  $MD(M_1, M_2)$  between models trained by noisy data and noise-free data is one way to measure the robustness of a classifier. The results on four datasets are reported in Table 4. (We

Dataset	Polynomial Kernel			RBF Kernel		
	SVM	FSVM	%impr	SVM	FSVM	%impr
	$C$	$d$		$C$	$2\sigma^2$	
Anneal-o	90.20	<b>93.32</b>	3.46	90.87	<b>92.65</b>	1.96
Autos	78.05	<b>79.02</b>	1.24	78.54	<b>80.00</b>	1.86
Balance-s	99.36	<b>99.68</b>	0.32	98.72	<b>100</b>	1.30
Breast-c	69.63	<b>81.12</b>	16.50	<b>75.91</b>	72.73	-4.19
Breast-w	97.00	<b>99.42</b>	2.49	97.14	<b>99.08</b>	2.00
Colic	82.61	<b>90.07</b>	9.03	84.51	<b>86.96</b>	2.90
Colic-o	<b>78.26</b>	75.00	-4.16	78.53	<b>79.35</b>	1.04
Credit-a	85.51	<b>96.52</b>	12.88	86.96	<b>94.35</b>	8.50
Credit-g	75.70	<b>87.20</b>	15.19	78.30	<b>79.80</b>	1.92
Diabetes	77.86	<b>78.85</b>	1.27	77.73	<b>77.87</b>	0.18
Glass	74.77	<b>76.07</b>	1.74	72.90	<b>76.07</b>	4.35
Heart-c	84.49	<b>95.18</b>	12.65	86.14	<b>94.59</b>	9.81
Heart-h	84.75	<b>95.24</b>	12.38	84.05	<b>92.86</b>	10.48
Heart-s	84.44	<b>93.33</b>	10.53	84.44	<b>85.19</b>	0.89
Hepatitis	85.17	<b>89.68</b>	5.30	86.46	<b>90.96</b>	5.20
Ionosphere	<b>91.75</b>	<b>91.75</b>	0	<b>94.89</b>	<b>94.89</b>	0
Iris	<b>96.67</b>	<b>96.67</b>	0	<b>97.33</b>	96.00	-1.37
Kr-vs-kp	<b>99.66</b>	<b>99.66</b>	0	<b>99.72</b>	99.56	-0.16
Labor	<b>89.47</b>	<b>89.47</b>	0	<b>92.98</b>	91.23	-1.88
Lymph	86.49	<b>87.16</b>	0.77	86.49	<b>87.16</b>	0.77
Mushroom	<b>100</b>	<b>100</b>	0	<b>100</b>	<b>100</b>	0
Sick	<b>96.66</b>	<b>96.66</b>	0	<b>97.03</b>	<b>97.03</b>	0
Sonar	84.62	<b>86.54</b>	2.27	88.46	<b>88.94</b>	0.54
Vote	96.55	<b>97.24</b>	0.71	<b>96.55</b>	96.09	-0.48
Weather-n	<b>71.43</b>	<b>71.43</b>	0	<b>71.43</b>	<b>71.43</b>	0
Average	86.44	<b>89.85</b>	3.94	87.44	<b>88.99</b>	1.77
Best	1	<b>17</b>		5	<b>16</b>	
Wilcoxon Test (Win/Draw/Loss)						
	Polynomial Kernel			RBF Kernel		
FSVM vs. SVM	10/14/1			7/17/1		

Table 3: %Accuracy and %improvement; SVM vs. FSVM (eq.35)

Noise Percentage (%)		0	10	20	30	40
Breast-w	SVM	0	11.59	14.16	18.03	20.03
	FSVM	0	8.44	13.02	15.74	17.02
Glass	SVM	0	9.35	17.29	21.50	26.64
	FSVM	0	4.67	9.81	13.55	16.36
Ionosphere	SVM	0	8.26	17.09	21.08	24.22
	FSVM	0	5.13	14.25	17.38	19.09
Iris	SVM	0	25.33	26.67	33.33	36.00
	FSVM	0	21.33	22.00	26.00	28.67

Table 4: %Model difference; SVM vs. FSVM with increasing noise

performed similar experiments on a large number of datasets and observed similar behavior.)

The graphs in Figure 3 show that the model differences of the fuzzy SVM classifier change less than those of the standard SVM classifier in the presence of noise. For example, on the “Glass” dataset, the model difference of our fuzzy SVM classifier increases from 0% to 16.36% when the noise level increases from 0% to 40%. Over the same range, the standard SVM classifier increases to

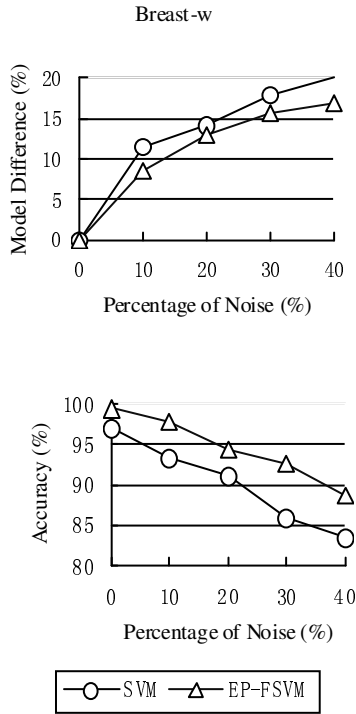


Fig. 3: Effect of increasing noise on model differences and accuracy for dataset “Breast-w”.

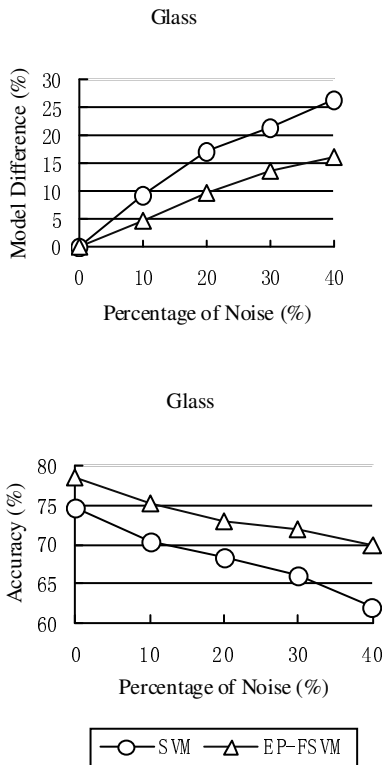


Fig. 4: Effect of increasing noise on model differences and accuracy for dataset “Glass”.

26.64%, about 10% more than the fuzzy SVM design. Figures 3 and 4 show that our fuzzy SVM classifier is more tolerant to noise and more accurate than the standard SVM classifier for these data sets, and for many others that are not reported here.

Dataset	Polynomial Kernel			RBF Kernel		
	SVM	DC FSVM	EP FSVM	SVM	DC FSVM	EP FSVM
Balance-s	<b>99.36</b>	96.59	98.76	98.72	97.20	<b>100</b>
Breast-w	97.00	95.97	<b>99.42</b>	97.14	95.97	<b>99.08</b>
Colic-o	<b>78.26</b>	75.00	75.00	78.53	78.70	<b>79.35</b>
Diabetes	77.86	76.94	<b>78.85</b>	77.73	77.61	<b>77.87</b>
Glass	74.77	75.65	<b>78.52</b>	72.90	73.58	<b>76.07</b>
Heart-s	84.44	93.50	<b>95.56</b>	84.44	85.33	<b>85.93</b>
Ionosphere	<b>91.75</b>	<b>91.75</b>	91.45	<b>94.89</b>	94.40	94.59
Iris	96.67	<b>97.09</b>	96.67	<b>97.33</b>	96.00	96.00
Sonar	84.62	86.13	<b>89.90</b>	<b>88.46</b>	87.50	87.50
Average	87.19	87.62	<b>89.35</b>	87.79	87.37	<b>88.49</b>
Best	2	1	5	3	0	6
Wilcoxon Test (Win/Draw/Loss)						
	Polynomial Kernel			RBF Kernel		
EP FSVM vs. SVM	4/4/1			3/6/0		
EP FSVM vs. DC FSVM	5/4/0			4/5/0		

Table 5: % Accuracy; numeric datasets in table 1; SVM vs. (DC FSVM) vs. (EP FSVM);

## 6 Related Work

In this section we compare our EP based weighting scheme with the weighting methods in [4, 15].

### 6.1 Data-center-based Weighting Method

In [4], the following weighting function based on data centers (DC) is used.

$$W'(I) = \begin{cases} 1 - \frac{\|x_+ - x_i\|}{\max\|x_+ - x_i\| + \delta}, & \text{if } y_i = 1; \\ 1 - \frac{\|x_- - x_i\|}{\max\|x_- - x_i\| + \delta}, & \text{if } y_i = -1. \end{cases} \quad (40)$$

where  $x_+$  and  $x_-$  are the data centers of classes + and - respectively, and  $\delta$  is a sufficiently small positive number used to avoid the case  $W'(I)=0$ . This weighting scheme is defined only for numeric feature vector data, whereas our weighting function (34) can handle datasets with either numeric or nominal attributes.

We calculate the weights based on data centers and use them to build a DC fuzzy SVM classifier. Table 5 compares this design to our fuzzy SVM classifier using EP-based weights (EP FSVM) for the numeric datasets in our study library.

Table 5 shows that our EP-based model is superior to the data-center-based scheme. The fuzzy SVM classifier combining EPs-based model got 5 wins, 4 draws, no losses (polynomial kernels), and 4 wins, 5 draws, no losses (RBF kernels). And it increases the accuracy significantly for most of the datasets. The best improvement is for dataset Glass, which enjoys improvements of 3.79% and 3.38%.

## 6.2 Other EP-based Weighting Methods

Alhammady and Ramamohanarao [15] introduced the following method for determining the sub-weights for decision trees:

$$SW'_k(I) = \frac{SW_k(I)}{\text{Median}SW_k}; \quad (41)$$

$$SW''_k(I) = \frac{SW'_k(I)}{\sum_{p=1}^m SW'_p(I)}; \quad (42)$$

where  $\text{Median}SW_k$  is the median of the sub-weight values in class  $C_k$ .  $SW_k(I)$  is the initial value of sub-weight of instance  $I$  for class  $C_k$ , and  $SW''_k(I)$  is the final value. This function normalizes the sum of all sub-weights of each instance so that it is equal to 1. Using these sub-weights to build weighted decision trees has shown good results [15]. But if we use  $SW''_k(I)$  as the final-weight of instance  $I$  to build the fuzzy SVM classifier, it poses a problem. For example, if there are two instances in class  $C_1$  with sub-weights (1, 0) and (10, 0), then the total-weights of both instances are equal to 1. Consequently, these two instances will exert equal influence during training because they have the same total weight, even though the instances are quite different. Our method overcomes this problem.

In [14] the following weighting function is used by Fan and Ramamohanarao:

$$TW'(I) = SW_k(I) - \sum_{q \neq k} SW_q(I) \quad (43)$$

This function, the sub-weight of the instance's home class minus the sum of all other sub-weights, can handle some two-class problems. But for multi-class cases (43) also poses a problem. For example, if there are two instances in class  $C_1$  with sub-weights (5, 2, 2, 1) and (5, 5, 0, 0), then the total-weights of both are equal to 0 ( $5 - 2 - 2 - 1 = 0$  and  $5 - 5 - 0 - 0 = 0$ ). Again, these two instances will exert equal influence during training because they have the same total weight, even though the instances are quite different. Indeed, they may be ignored as noise with such low total-weights. However, these two instances are quite different and should not be treated as the same, or as noise.

Using (34) with sub-weights (5, 2, 2, 1) and (5, 5, 0, 0), the total-weights become 3.16 and 4.06 respectively. Thus indicating the second set is more important than the first. Further experiments have convinced us that our weighting method is usually better than these EP-based weighting methods.

## 7 Conclusions

In this paper we have defined and developed a new, robust weighting method to build fuzzy SVM classifiers by means of emerging patterns, or EPs. We use EPs to discover class memberships for the training instances, and combine the class membership weights of each instance into a single weight which represents the instance's importance in building the classifier. Then we use the

weighted training instances rather than the original crisp ones to construct the fuzzy SVM classifier.

Our EP-based fuzzy SVM classifier performs consistently better than the standard SVM classifier over a wide variety of datasets and data types. The accuracy of our weighting method is usually better than some other recently studied weighted designs. Typical improvements are in the range of 2%-10%. Finally, our experiments suggest that our method is more tolerant to noise and outliers than the standard SVM. Of course, no amount of empirical evidence supports sweeping generalizations about classifier performance, but we think our experiments are sufficiently broad that we can recommend this design with some confidence.

In the future we will explore our weighting method with other classifiers (e.g., C4.5 and decision trees), and furthermore, in the fields other than classification (e.g., regression and clustering).

## 8 Acknowledgment

The authors would like to thank Dr Laurence Park for providing useful comments on a draft of this paper.

## 9 References

- [1] Vapnik, V.N. (1998): *Statistical Learning Theory*. New York, John Wiley.
- [2] Burges, C.J.C. (1998): A tutorial on support vector machines for pattern Recognition. *Data Mining and Knowledge Discovery* 2(2):121-167.
- [3] Inoue, T., & Abe, S. (2001): Fuzzy support vector machines for pattern classification. *Proc. Int. Joint Conf. on Neural Networks*, Washington, 2:1449-1454.
- [4] Lin, C., & Wang, S. (2002): Fuzzy support vector machines. *IEEE Trans. Neural Networks*, 13(2):464-471.
- [5] Huang, H., & Liu, Y. (2002): Fuzzy support vector machines for pattern recognition and data mining. *Int. Journal of Fuzzy Systems*, 4(3):826-835.
- [6] Abe, S., & Inoue, T. (2002): Fuzzy support vector machines for multiclass problems. *10th European Symposium on Artificial Neural Networks*, Bruges, 113-118.
- [7] Dong, G., & Li, J. (1999): Efficient mining of emerging patterns: discovering trends and differences. *Proc. 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, San Diego, 43-52.
- [8] Dong, G., Zhang, X., Wong, L., & Li, J. (1999): CAEP: Classification by aggregating emerging patterns. *Proc. 2nd Int. Conf. Discovery Science*, Tokyo. 30-42.
- [9] Li, J., Dong, G., & Ramamohanarao, K. (2001): Making use of the most expressive jumping emerging patterns for classification. *Knowledge Information Systems*, 3(2):131-145.

- [10] Zhang, X., Dong, G., & Ramamohanarao, K. (2000): Exploring constraints to efficiently mine emerging patterns from large high-dimensional data sets. *Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Boston, 310-314.
- [11] Li, J., Ramamohanarao, K., & Dong, G. (2000): The space of jumping emerging patterns and its incremental maintenance algorithms, *Proc. 17th Int. Conf. on Machine Learning*, Standord, 551-558.
- [12] Fan, H., & Ramamohanarao, K. (2003): Efficiently mining interesting emerging patterns. *Proc. 4th Int. Conf. on Web-Age Information Management*, Chengdu, 189-201.
- [13] Ramamohanarao, K., & Bailey, J. (2003): Discovery of emerging patterns and their use in classification. *Australian Conf. on Artificial Intelligence*, Perth, 1-12.
- [14] Fan, H., & Ramamohanarao, K. (2005): A weighting scheme based on emerging patterns for weighted support vector machines. *IEEE Int. Conf. on Granular Computing*, Beijing, 2:435-440.
- [15] Alhammady, H., & Ramamohanarao, K. (2006): Using emerging patterns to construct weighted decision trees. *IEEE Trans. Knowledge and Data Engineering*, **18**(7):865-876.
- [16] Sun, Q., Zhang, X., & Ramamohanarao, K. (2003): Noise tolerance of EP-based classifiers, *Australian Conf. on Artificial Intelligence*, Perth, 796-806.
- [17] Witten, I.H., & Frank, E. (1999): *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. San Francisco, Morgan Kaufmann.
- [18] Han, J., & Kamber, M. (2000): *Data Mining, Concepts and Techniques*. Burlington, Morgan Kaufmann Publishers.
- [19] Newman, D.J., Hettich, S.C., Blake, L., & Merz, C.J. (1998): *UCI repository of machine learning databases*.  
<http://www.ics.uci.edu/~mllearn/MLRepository.html>
- [20] Tsang, E.C.C., Yeung, D.S., & Chan, P.K. (2003): Fuzzy support vector machines for solving two-class problems. *Proc. 2nd Int. Conf. on Machine Learning and Cybernetics*, Xi'an, 1080-1083.
- [21] Tsujinishi, D., & Abe, S. (2003): Fuzzy least squares support vector machines for multiclass problems. *Neural Networks*, **16**(1):785-792.
- [22] Abe, S. (2004): Fuzzy LP-SVMs for multiclass problems. *12th European Symposium on Artificial Neural Networks*, Bruges, 429-434.
- [23] Zhang, X. (1999): Using class-center vectors to build support vector machines. *Proc. IEEE Workshop on Neural Networks for Signal Processing IX*, Madison, 3-11.
- [24] Cong, D., Wang, J., Yang W., & Zhang, S. (2005): Pattern Decomposition Algorithm Based on FP-tree. *Computer Engineering*, **31**(16):77-79, 88.
- [25] Fayyad, U.M., & Irani, K.B. (1993): Multi-interval discretization of continuous-valued attributes for classification learning. *Proc. 13th Int. Joint Conf. on Artificial Intelligence*, Chambéry, 1022-1029.
- [26] Cawley, G.C., & Talbot, N.L.C. (2001): Manipulation of prior probabilities in support vector classification. *Proc. Int. Joint Conf. on Neural Networks*, Washington, **4**:2433-2438.



# Exploit Keyword Query Semantics and Structure of Data for Effective XML Keyword Search

Khanh Nguyen

Jinli Cao

Department of Computer Science and Computer Engineering  
La Trobe University, Melbourne Australia  
Email: {tuan.nguyen, j.cao}@latrobe.edu.au

## Abstract

Keyword search is a natural and user-friendly mechanism for querying XML data in information systems and Web based applications. One of the key tasks is to identify and return meaningful fragments as results, due to the limited expressiveness and the ambiguity of keyword queries. In this paper, we first studied query keyword patterns in order to exploit the user's search intention behind the input keywords. The outcome of this task is that keywords in the query are classified as required information and search conditions (or predicates). In addition, unlike previous work that our work only returns desired fragments as results. Each returned result must satisfy the search conditions rather than simply contain all query keywords. To further prune irrelevant fragments we introduce a novel notion called Relevant Lowest Common Ancestor (RLCA) which effectively and precisely captures the meaningful and relevant fragments to the given keyword query. We conducted extensive experimental studies to prove the effectiveness of our approach.

**Keywords:** XML, Keyword Search.

## 1 Introduction

With XML rapidly emerging as a standard for representing, publishing and exchanging data over the Internet, it is demanding for adapting keyword search to XML data. This is because keyword search provides a friendly mechanism to access XML data without the knowledge of the structured query languages and the underlying complex data schema. The structured query languages (e.g XPath and XQuery) convey obvious semantic meanings, thus they can retrieve precise results. In contrast, keyword search may not be able to return precise results as users expected due to the limited expressiveness and the ambiguity of keyword queries. Thus, identifying meaningful and relevant results becomes a very challenging and key task of XML keyword search.

Several attempts have been made toward identifying meaningful and relevant fragments to XML keyword search (Guo et al. 2003, Xu & Papakonstantinou 2005, Cohen et al. 2003, Li et al. 2007). Based on the notion of Lowest Common Ancestor (LCA) from graph theory (Harel & Tarjan 1984, Schieber & Vishkin 1988, Bender et al. 2005) and Smallest LCA (SLCA), we can classify previous attempts into LCA-

based approaches (Guo et al. 2003, Cohen et al. 2003, Li et al. 2007) and SLCA-based approaches (Xu & Papakonstantinou 2005). Given an XML tree  $T$  and a keyword query, the semantics of LCA is to return a node  $n$  in  $T$  which the subtree rooted at  $n$  contains all query keywords, excluding other LCA nodes and their keyword nodes which are descendants of  $n$  (Guo et al. 2003). As for SLCA nodes, they are subsets of LCA nodes, however they have no descendants which also contain all keywords of the query (Xu & Papakonstantinou 2005).

A well known problem of the SLCA-based approaches is that they may miss relevant results. For example, when querying  $Q_1$  on document  $D_1$  (refer to Table 1 and Figure 1), the SLCA-based approaches consider the fragment rooted at *article* [0.2.0.1] (Figure. 3b) as an irrelevant result. This is because this fragment contains another fragment (Figure. 3a) which also consists of all query keywords. However, we can see that this fragment is definitely relevant to the query  $Q_1$ . In contrast, LCA approaches (Xu & Papakonstantinou 2005) return all fragments consisting of query keywords that may include irrelevant results. For example, suppose that  $Q_3$  issued on the document  $D_2$ , the nodes *name* [0.1.0.0] and *nationality* [0.1.1.1] are two matches to keywords "*Gasol*" and "*USA*" respectively and their LCA is [0.1]. However, this subtree (Figure. 4a) is not a desired result because *name* [0.1.0.0] and *nationality* [0.1.1.1] do not belong to the same player.

More recently, XSearch (Cohen et al. 2003), CVLCA (Li et al. 2007) and MLCA (Li, Yu & Jagadish 2008) have been proposed to improve the effectiveness of the keyword search. Rather than simply return all fragments rooted at LCA or SLCA nodes, the interconnection between nodes in the fragment is used to prune undesired fragments. To fulfil this aim, node labels and structures of XML data are two main sources of considerations. However, these approaches have the following limitations:

1. They fail to identify the meaningless relationship between nodes with different labels but having the same type.
2. The fail to recognize the meaningful relationship between nodes with same label.
3. The semantics of keyword query has not been utilized to prune irrelevant results, especially in the presence of keyword ambiguity.

The detailed discussion and analysis of these problems will be presented in Section 2.

This paper introduces a novel approach called Relevant LCA (RLCA) to effectively and accurately capture relevant fragments to XML keyword search. RLCA is a LCA-based approach, but it is different from previous approaches which mainly focus on the evaluating node labels or the structure of a fragment

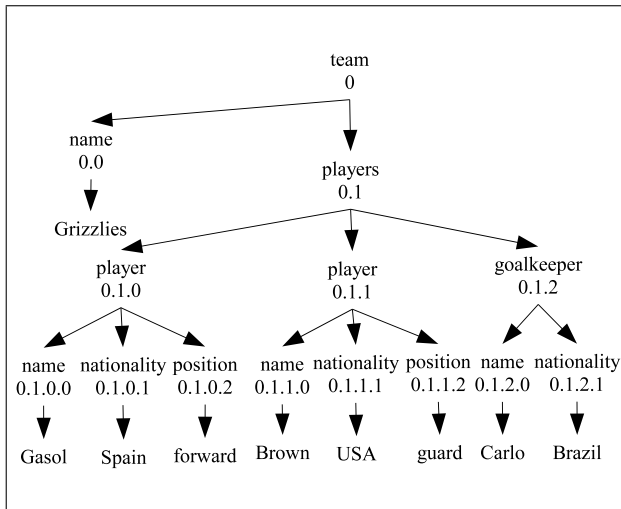
$Q_1$	Ziyang Liu, XML Keyword Search
$Q_2$	Ziyang Liu, article, title, XML Keyword Search
$Q_3$	Gasol, USA
$Q_4$	Gasol, Brazil
$Q_5$	Grizzlies, Gasol, Brown, position

Table 1: Sample keyword queries

for the relevancy. Our approach exploits the semantics of the keyword query in evaluation process. In addition, unlike previous approaches which simply base on the tag name or the label of a node to identify the relationship between nodes, we use node categories to capture node types and their equivalent node types. Therefore, our approach can tackle the aforementioned limitations of existing approaches. The contributions of this paper are described as follows.

- Proposed a novel notion called Relevant LCA (RLCA) to capture the semantics of LCA nodes which the fragments rooted at those nodes are meaningful and relevant to the keyword query.
- Exploited the both structure of data and the semantics of keyword queries to effectively identify meaningful fragments for XML keyword search.
- Addressed the problem of keyword ambiguity by exploiting the search conditions from the query.
- Implemented the algorithms and conducted extensive experiments on real data sets to show that the proposed RLCA outperforms existing approaches in term of effectiveness and precision while maintaining the efficiency.

The remainder of this paper is organized as follows. Next section discusses and analyzes the weakness of existing solutions and inspires our motivation. Section 3 introduces the notion of RLCA and its semantics. In Section 4, we present the algorithm of RLCA. Extensive experimental evaluations are provided in Section 5. Section 6 discusses several orthogonal studies could be incorporated into our work. Finally, the conclusions of the paper are presented in Section 7.

Figure 2: Sample XML document  $D_2$ .

## 2 Preliminary and Motivation

### 2.1 Data Model and Query

Each XML document is modeled as a rooted and labeled tree. Every internal node in the tree has a name

which is either an element name or a tag name from the XML document. Each leaf node in the XML tree has got a data value. We call internal nodes and leaf nodes as structure nodes and value nodes respectively. Each structure node is encoded by Dewey code as its ID.

A user query is expressed as a set of keywords which may match the labels or values of the nodes in XML trees. Without loss of generality, the 'AND' semantics is default in query results which is similar to the existing approaches (Guo et al. 2003, Xu & Papakonstantinou 2005, Li, Yu & Jagadish 2008, Li et al. 2007, Liu & Chen 2008). A keyword query search on an XML data returns a set of meaningful subtrees. Conditions for a fragment to be meaningful will be presented in next Section.

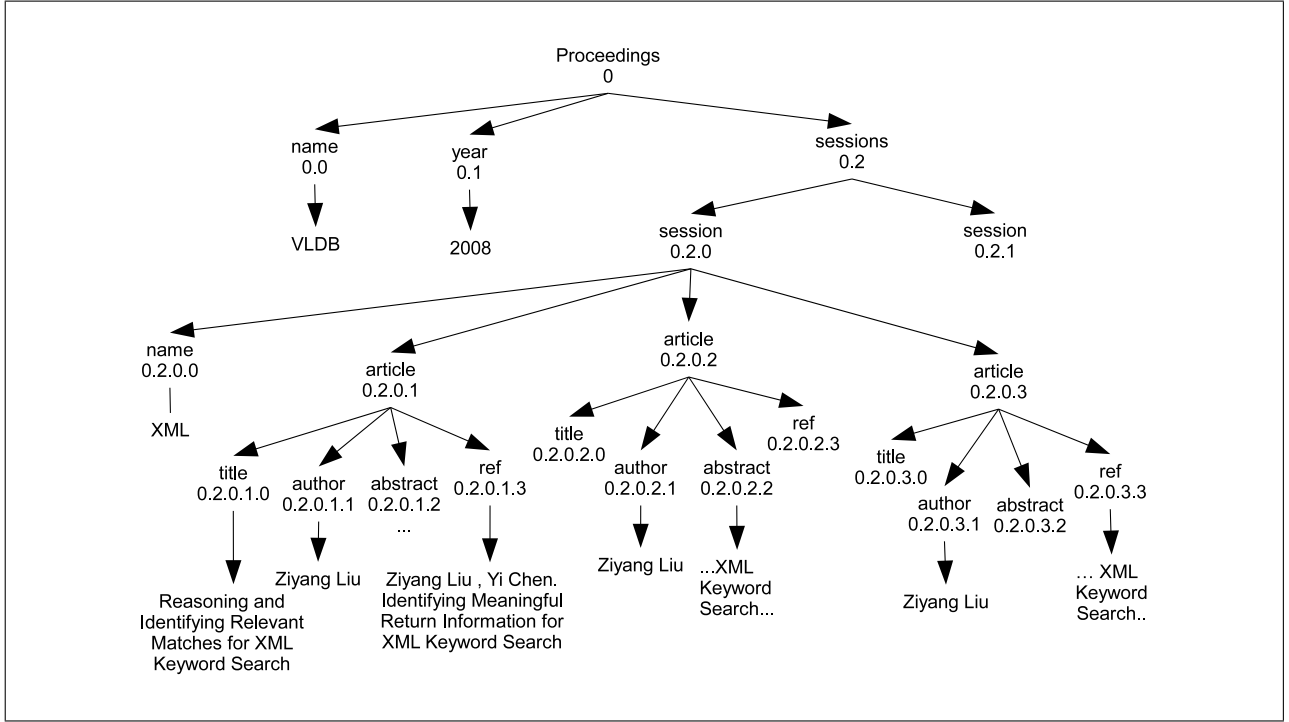
### 2.2 Analysis on Existing Approaches and Motivations

Recently, several attempts such as XSearch (Cohen et al. 2003), CVLCA (Li et al. 2007) and MLCA (Li, Yu & Jagadish 2008) have been made toward improving the meaningfulness of return fragments for XML keyword search. Specifically, in XSearch, two nodes  $u$  and  $v$  are meaningfully related if the shortest path between  $u$  and  $v$  does not have two distinct nodes with same label, except  $u$  and  $v$ . A fragment  $F$  is meaningful if it satisfies two conditions: (i)  $F$  contains the matches to all search terms and (ii) every two matches in  $F$  has to be meaningfully related (all-pair semantics) or there exists a node in  $F$  such that every match in  $F$  has to be meaningfully related to it (star-semantics). In (Li et al. 2007), an approach called Compact Valuable LCA (CVLCA) has been introduced to answer keyword search. A node  $u$  is called VLCA if the fragment  $F$  rooted at  $u$  satisfies star-semantics. A node  $u$  is a CVLCA if it is a VLCA node and  $u$  dominates every node in  $F$ , where  $u$  dominates a node  $v$  in  $F$  if there does not exist another LCA node  $u'$  which is a descendant of  $u$  and the fragment rooted at  $u'$  contains  $v$ .

Let us consider query  $Q_3 = \{\text{Gasol, USA}\}$ . The aforementioned algorithms will identify the fragment in Figure 4a meaningless to  $Q_3$  because there exist two nodes [0.1.0] and [0.1.1], and both have the same label *player* on the shortest path between the matching nodes *name* [0.1.0.0] and *nationality* [0.1.1.0]. On the other hand, these approaches may fail to identify the meaningless relationships between nodes with different labels but having the same type since they use node labels as the identifiers. Moreover, they also may fail to identify the meaningful relationship between nodes with the same labels. Therefore, they may fail to identify irrelevant fragments and also may miss relevant results.

Consider query  $Q_4 = \{\text{Grizzlies, Gasol, Brown, position}\}$  and sample document  $D_2$ , both XSearch and CVLCA fail to identify Figure 4b as an irrelevant fragment to query  $Q_4$  because there do not exist two nodes with the same label on the path between matching nodes. Now, we consider query  $Q_5 = \{\text{Grizzlies, Gasol, Brown, position}\}$  on XML document  $D_2$ . A subtree contains the nodes which satisfy all search terms depicted in Figure 4c. However, the approaches XSearch and CVLCA will exclude this subtree from the meaningful results of  $Q_5$  because there are two nodes [0.1.0] and [0.1.1] with the same label *player* on the shortest path between two matches *name* [0.1.0.0] and *nationality* [0.1.1.0]. Actually, we can see that this fragment is absolutely relevant to  $Q_5$ .

To tackle with aforementioned problems, (Li, Yu & Jagadish 2008) proposed an approach called MLCA which prunes irrelevant subtrees based on the structure, rather than using node labels as XSearch and

Figure 1: Sample XML document  $D_1$ .

CVLCA. Two nodes  $u$  and  $v$  are meaningfully related if there does not exist a node  $u'$  (or  $v'$ ) which has the same label with  $u$  (or  $v$ ) and  $LCA(u', v')$  (or  $LCA(u, v)$ ) is a descendant of  $LCA(u, v)$ . Based on this definition, MLCA can success in identifying Figure 4b as an irrelevant fragment because two nodes  $[0.1.0.0]$  and  $[0.1.2.1]$  are unmeaningfully related. The reason is that there exists node  $[0.1.0.1]$  which has the same label “nationality” with node  $[0.1.2.1]$  and  $LCA([0.1.0.0], [0.1.0.1]) = \text{node } [0.1.0]$  is a descendant of node  $[0.1] = LCA([0.1.0.0], [0.1.2.1])$ . However, MLCA will fail to identify Figure 4b as an irrelevant result when node  $[0.1.0.1]$  is null, e.g. due to being optional.

In addition, those approaches regarded the keyword query as a set of keywords. The semantics and relationships between keywords have not been exploited yet in evaluating the meaningfulness of a fragment. For example, if query  $Q_2 = \{\text{Ziyang Liu, article, title, XML Keyword Search}\}$  is issued on the sample document  $D_1$  which is to find all articles of Ziyang Liu with titles containing “XML Keyword Search”, all previous approaches identify Figure 3b, 3c and 3d as relevant fragments to query  $Q_2$ . However, we can see that only Figure 3b has the title consisting of “XML Keyword Search”. Figure 3c and Figure 3d contain “XML Keyword Search” in the node *abstract* and node *reference* respectively. Hence they are unlikely relevant to query  $Q_2$ . Current approaches simply return all these fragments, so they have very low precision.

To solve the discussed problems, this paper exploits the structure of XML data in combination with the semantics of keyword query in order to improve the precision and recall of the existing approach. The details of our work is provided in the following sections.

### 3 Relevant LCA

#### 3.1 XML Data and Node Types

As discussed previously, existing approaches use node labels to identify the relationship between nodes.

Therefore, they may fail to recognize the meaningless correlations between nodes with different labels but having equivalent types as well as the meaningful relationship between nodes with the same label. To tackle with those problems, we use the prefix path of a node, combined with the node category to form the node type. The prefix path of a node is the path from the root to its parent. For instance, the prefix path of node  $[0.1.0.0]$  in document  $D_2$  is  $[team/players/player]$  (refer to Fig 2). For node categories, we adopted the classifications defined by (Liu & Chen 2007).

1. A node represents an entity if it corresponds to a \*-node in the DTD.
2. A node denotes an attribute if it does not correspond to a \*-node, and only has one child, which is a value.
3. A node is a connection node if it represents neither an entity nor an attribute. A connection node can have a child that is an entity, an attribute or another connection node.

As pointed out by the authors, the above classifications may fail to identify some multi-attributes as entities (refer to (Liu & Chen 2007) for details). Thus, we apply human intervention to fix the misleading identified nodes after classifying the node categories.

#### Definition 1 (Node Equivalence & Same Type)

Two nodes  $u$  and  $v$  are equivalent if they have the same prefix path and the node category. Two nodes  $u$  and  $v$  have the same node type if they are equivalent and they have the same node label.

Using Definition 1, our work can distinguish that nodes *player*  $[0.1.0]$  and *goalkeeper*  $[0.1.1]$  in the document  $D_2$  are in the equivalent type because they have the same prefix path which is  $[team/players]$  and both are entities. Notice that two nodes with the same label OR the same prefix path do not guarantee they are equivalent. For example, two nodes *name*  $[0.0]$  and *name*  $[0.1.0.0]$  in Figure. 4 have the same label but

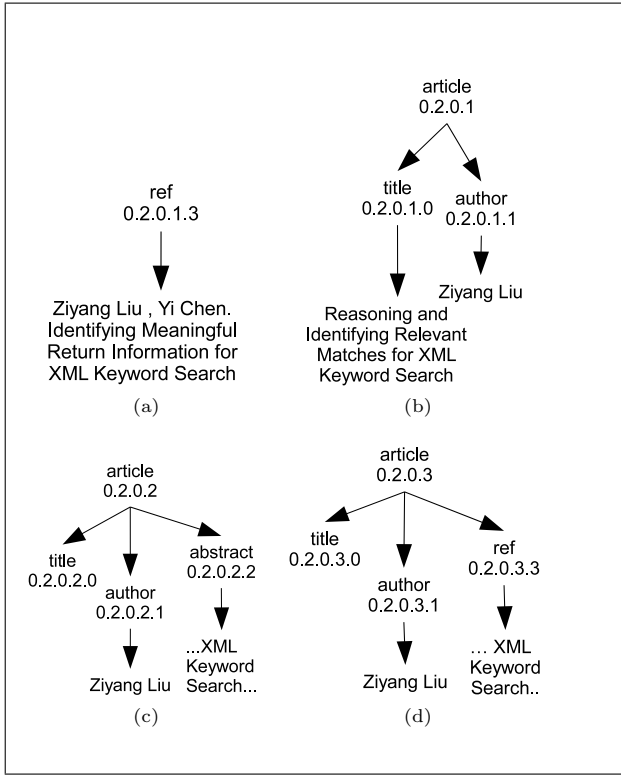


Figure 3: (a). An unique fragment to  $Q_1$  returned by SLCA-based approach; (b). A fragment to  $Q_1$  which is missed by SLCA-based approaches; (c), (d). Irrelevant fragments to  $Q_2$ .

they have different types since they have different prefix paths, which are  $[team]$  and  $[team/players/player]$  respectively. Similarly, two nodes  $name$  [0.0] and  $players$  [0.1] have the same prefix path ( $team$ ) but they have different types because  $name$  [0.0] is a attribute while  $players$  [0.1] is a grouping node. Based on node types, we formally define the meaningful relation between two nodes as follows:

#### Definition 2 (Meaningfully Related Nodes)

Two nodes  $u$  and  $v$  are meaningfully related if either of following conditions holds:

- (i)  $u$  and  $v$  have the same type.
- (ii) there do not exist two nodes with the equivalent type on the shortest path between  $u$  and  $v$ , except  $u$  and  $v$ .

Notice that our definition is different from XSearch in two main points: (i) It accepts that two nodes with the same type can be meaningfully connected in a subtree, due to the fact that a user may be interested in finding more than one entities with the same type. Considering query  $Q_5$ , for instance, the user may be interested in returned fragments which contain information about both players "Gasol" and "Brown". (ii) For queries to find information about single entity (e.g query  $Q_4$ ), we use node types from Definition 1 to detect the relevancy of fragments rather than simply use node labels. Hence, we can detect that some nodes can be equivalent even though they have different labels, such as nodes  $player$  [0.1.1] and  $goalkeeper$  [0.1.2] in document  $D_2$  (refer to Fig. 2). In contrast, some nodes with the same label (e.g  $name$  [0.0] and  $name$  [0.1.0.0] in document  $D_2$ ) can have different types.

By Definition 2, we can improve the precision and recall of previous approaches. For instance, Figure 4b can be detected as an uninteresting fragment to query  $Q_4$ . Conversely, Figure 4c is returned as a relevant

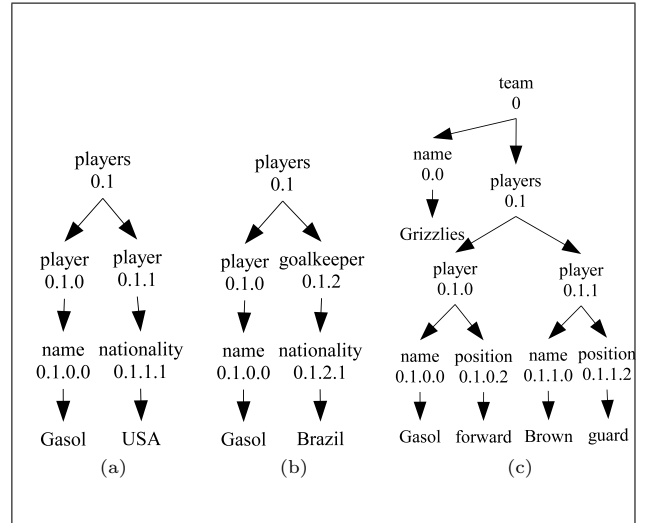


Figure 4: (a), (b) are irrelevant fragment to  $Q_3$ ; (c). Relevant fragment to  $Q_5$ .

result to query  $Q_5$ . However, both definitions 1 & 2 cannot help us identify which of those fragments in Figure 3 is relevant to query  $Q_2$  yet. To solve this problem, next section will analyze semantics of the keyword query, which is another important factor for effective XML keyword search.

### 3.2 Semantics of Keyword Query

Previous approaches treat keyword query just as a set of keywords. We adopt the classification approach proposed by (Liu & Chen 2007) for grouping keywords in a query into two categories: (i) if an input keyword  $k$  matches the value of a node or it matches the label of a node that has got a descendant matching another keyword then  $k$  specifies a predicate, corresponding to the where clause in XQuery; (ii) otherwise, it is treated as a return node, corresponding to the return clause in XQuery.

Their work focuses on identifying return nodes from a given keyword query; each predicate is just a keyword in the query. For example, query  $Q_2 = \{Ziyang Liu, article title XML Keyword Search\}$  comprises of all predicates and each keyword is a predicate. All subtrees in which each contains all keywords in the query are returned as relevant results. Following that classification, Figures 3b, 3c, 3d are all relevant results to the query  $Q_2$  because each of them contains all input keywords in the query. However, actually only Figure 3b is relevant to the query  $Q_2$ . Thus, an immediate question is how we can group a given keyword query  $Q = \{k_1, k_2, \dots, k_m\}$  to form predicates and return nodes. We observe that there are relations between keywords in the query and usually the keywords can be combined together to form a predicate or return node. For example, the query  $Q_2$  can be expressed as two predicates ( $:Ziyang Liu$  and  $title: "XML Keyword Search"$ ) and a return node ( $article$ ). From the observation, given a keyword query  $Q = \{k_1, k_2, \dots, k_m\}$  and an XML tree  $T$ , we can group keywords together to form predicates and return nodes based on the algorithm 1.

For example, consider query  $Q_2 = \{Ziyang Liu, article, title, XML Keyword Search\}$  on the document  $D_1$ , based on above Algorithm 1, we can identify that "article" is a return node, while  $:Ziyang Liu$  and "title:XML Keyword Search" are two predicates. Specifically, keywords "Ziyang Liu" matches the value of nodes [0.2.0.1.1], [0.2.0.1.3], [0.2.0.2.1] and [0.2.0.3.1] in document  $D_1$ . Keyword "article" matches the la-

**Algorithm 1** [Keyword Classification]

**Input:** a keyword query  $Q = \{k_1, k_2, \dots, k_m\}$  and an XML tree  $T$ .

**Output:** the associated predicates and return nodes

- 1: **if** a keyword  $k_i$  matches the label of a node associated with a descendant which has got a value matching keyword  $k_{i+1}$  (or a sequence of keywords  $k_{i+1}, k_{i+2}, \dots, k_{i+j}, i+j \leq m$ ) **then**
- 2:   **return**  $k_i$  and  $k_{i+1}$  (or  $k_{i+1}, k_{i+2}, \dots, k_{i+j}$ ) form a predicate, noted as  $k_i : k_{i+1}$  (or  $k_i : k_{i+1}, k_{i+2}, \dots, k_{i+j}$ ).
- 3: **else if** a keyword  $k_i$  (or sequence of keywords  $k_i, k_{i+1}, \dots, k_{i+j}, i+j \leq m$ ) matches the value of a node which has no ancestor matching another keyword  $k_l$  ( $1 \leq l < i$ ) **then**
- 4:   **return**  $k_i$  (or  $k_i, k_{i+1}, \dots, k_{i+j}$ ) form a predicate, noted as  $:k_i$  (or  $:k_i, k_{i+1}, \dots, k_{i+j}$ ).
- 5: **else**
- 6:   **return**  $k_i$  is a return node. It means  $k_i$  matches the label of a node which has no any descendants having a value matching  $k_{i+1}$  (or a sequence of keywords  $k_{i+1}, k_{i+2}, \dots, k_{i+j}, i+j \leq m$ ). by (Cohen et al. 2003)
- 7: **end if**

bel of *article* [0.2.0.1], *article* [0.2.0.2] and *article* [0.2.0.3] while none of these nodes associated with a descendant consisting of the value matching keyword “title”. While keyword “title” matches the label of [0.2.0.1.0] which has got the value matching sequence of keywords “XML Keyword Search”.

A meaningful result must satisfy all predicates and contain required information in its subtree. Thus, we can identify that only Figure 3b is meaningful to query  $Q_2$  because it is a unique fragment in document  $D_1$  which satisfies all predicates and contains all return nodes in query  $Q_2$ .

**Definition 3 (RLCA)** *Given a keyword query  $Q = \{k_1, k_2, \dots, k_m\}$  which is classified into a set of return nodes  $R$  and a set of predicates  $P$ . A RLCA fragment of  $Q$  on document  $D$  is a subtree  $T$  of  $D$ , which holds the following conditions:*

1.  $T$  satisfies every predicate in  $P$  at least once,
2.  $T$  contains every return node in  $R$  at least once,
3. for any pair of nodes  $u$  and  $v$  in  $T$ , either (i)  $u$  is meaningfully related to  $v$  or (ii) there exists a node  $v'$  having the same type with  $v$  while  $u$  is meaningfully related to  $v'$ .
4. no proper subtree of  $T$  can hold for the above conditions.

A remark in Definition 3 is that a fragment is meaningful only if it satisfies all predicates from the query. It differs from existing approaches which define a set of fragments consisting of the query keywords as relevant results. We have pointed out in previous sections that containing all the query keywords does not guarantee that all fragments will be the relevant results. Furthermore, it also can solve the node-type ambiguity problem of XSearch. For example, the fragment in Figure. 4b is not a meaningful fragment of query  $Q_4$  because two nodes *player* [0.1.0] and *player* [0.1.2] are in the equivalent type on the path between *name* [0.1.0.0] and *nationality* [0.1.2.1], and there no any other nodes have the same type with *nationality* [0.1.2.1] and meaningfully related to *name* [0.1.0.0].

However, the fragment in Figure. 4c is meaningful to query  $Q_5$  because for every pair of nodes  $u$  and  $v$ , either  $u$  is meaningfully related to  $v$  or there exists a node  $v'$  which is the same type with  $u$  while  $u$  and  $v'$  are meaningfully related. Specifically, the pair of nodes *name* [0.1.0.0] and *name* [0.1.1.0] (as well as the pair of nodes *position* [0.1.0.2] and *position* [0.1.1.2]) are meaningfully related because they have the same type. The pair of nodes of *name* [0.1.0.0] and *position* [0.1.0.2] (as well as the pair of nodes *name* [0.1.1.0] and *position* [0.1.1.2]) are meaningfully related because there do not exist two nodes with the equivalent type on the path between them. Now, considering two nodes *name* [0.1.0.0] and *position* [0.1.1.2], we can find another node *position* [0.1.0.2] having the same type with *position* [0.1.1.2] which is meaningfully related to the node *name* [0.1.0.0]. It is similar for nodes *name* [0.1.1.0] and *position* [0.1.0.2]. Finally, the node *name* [0.0] is meaningfully related to all other nodes because there do not exist two nodes with equivalent type on the paths between them.

#### 4 RLCA Algorithm

In this section, we propose an algorithm to identify all meaningful fragments of a given keyword query according to Definition 3. We divide our algorithm into two stages. In stage 1, we focus on identifying all fragments which each satisfies condition 1, 2 and 4 of Definition 3 as potential candidates. Stage 2, we identify the meaningfulness and relevancy of the potential results related to the query.

##### Stage 1: Identifying potential candidates.

Given a keyword query and an XML document modeled as a tree  $T$ , the aims of this stage are: (i) classifying the keywords into two sets of predicates and return nodes using Algorithm 1; (ii) identifying a set of potential candidates which each candidate satisfies conditions 1, 2 and 4 of Definition 3. Notice that predicates and return nodes in our work have the forms of tag-keyword queries in: (Cohen et al. 2003). Therefore, after classifying the query keywords into predicates and return nodes, we adopted the algorithm in (Cohen et al. 2003) to calculate the set of potential candidates. The details of this stage is demonstrated by Algorithm 2.

**Algorithm 2** [Potential Candidates]

**Input:** query  $Q$ , XML tree  $T$ .

**Output:** a set of potential candidates which each of them contains all predicates and return nodes in  $Q$ , XML tree  $T$ .

- 1: Classifying keywords in query  $Q$  into a set of predicates  $P$  and a set of return nodes  $R$  based on Algorithm 1.
- 2: Identifying all subtrees in  $T$  which each of them satisfies both  $P$  and  $R$  as potential candidates (adopted the Algorithm proposed by (Cohen et al. 2003)).

**Stage 2: Relevant LCA.** This stage focuses on how to evaluate the meaningfulness of a potential candidate returned from Stage 1. It includes a procedure called *Meaningful* ( $u, v$ ) (line 11 of Algorithm 3) which identifies the meaningful connection between a pair of nodes  $u$  and  $v$  as defined in Definition 2. Then, this procedure is used to identify the meaningfulness of a potential candidate. Notice that in Stage 1, we have identified a set of potential candidates already satisfying conditions 1, 2 and 4 of Definition 3. Therefore, in this stage, if a potential candidate in the returned results, only if it satisfies the condition

3 of Definition 3. Stage 2 is demonstrated by Algorithm 3. For each potential candidate (line 1), this algorithm evaluates every pair of nodes in the candidate to evaluate its relevancy (line 2, 3 and 4). At the end of Algorithm 3, a set of results returned will be guaranteed to be meaningful and relevant to the input query.

---

**Algorithm 3** [RLCA]

---

**Input:** a set of potential candidates identified from Algorithm 2.

**Output:** a set of relevant results.

```

1: for each potential candidate F do
2:   for every pair of two nodes u and v in F do
3:     if not Meaningful (u, v) then
4:       if  $\nexists v'$  in F | Meaningful(u, v') then
5:         return false
6:       end if
7:     end if
8:   end for
9:   return true
10: end for
11: Meaningful (u, v)
    {return true if u and v are meaningful related according to Definition 3, otherwise returns false}
12: if u and v have the same type then
13:   return true
14: else
15:   {find all nodes on the paths from u to v}
16:   OnPathNodes  $\leftarrow$  all nodes on the path between u and v
17:   if  $\exists u_1, u_2 | (u_1, u_2 \in \text{nodesOnPaths}) \wedge (u_1 \text{ and } u_2 \text{ are equivalent})$  then
18:     return false
19:   else
20:     return true
21:   end if
22: end if

```

---

## 5 Experiments

To evaluate the performance of the proposed RLCA algorithms, we designed and performed a comprehensive set of experiments on real data sets. Our approach is compared against the state-of-art proposals of XSearch (Cohen et al. 2003) and CVLCA (Li et al. 2007). We employ four metrics, elapsed time, precision, recall and  $\mathcal{F}$ -measure to evaluate the efficiency and effectiveness of RLCA compared with other approaches. The relevant results are manually obtained from user studies. To obtain a set of relevant result for each keyword query, we manually defined a XQuery query for each keyword query. Those XQuery queries were run on the same set of data and the returned results are used as “ground truth” for our experiments.

### 5.1 Experiment Setup

The experiments were conducted on a 3.2GHz P4 CPU running Windows XP Professional with 1GB of RAM. The algorithms were implemented in Java. We used Oracle Berkeley DB<sup>1</sup> as a tool for creating indexes.

**Data Set.** We have tested several real data sets: DBLP<sup>2</sup>, SIGMOD Record<sup>3</sup> and XMark<sup>4</sup>. The DBLP is a data set about bibliographic information on major

computer science journals and proceedings. SIGMOD Record is an XML version of SIGMOD Record articles database.

**Query Set.** Our query set consists of 60 queries in total. For each data set, we tested twenty keyword queries. The queries are selected to represent a variety of cases, where both labels and values of data are used.

### 5.2 Effectiveness Test

This section evaluates the effectiveness of our algorithms, compared with the state-of-art proposals of XSearch (Cohen et al. 2003) and CVLCA (Li et al. 2007). We conducted twenty keyword queries for each data set. The effectiveness and efficiency was measured by four standard metrics borrowed from information retrieval (IR) literature, precision, recall and  $\mathcal{F}$ -measure and elapsed time. The improvements of precision and  $\mathcal{F}$ -measure are demonstrated by Figure. 5a and Figure. 6a respectively.

#### 5.2.1 Search accuracy

We used the standard metric, *precision*, to measure the search accuracy which indicates the fraction of results in the returned answer that are correct. We can see, in Figure. 5a, RLCA outperforms CVLCA (Li et al. 2007), especially in some tested queries i.e.  $Q_2$ . In these queries, our keyword classification mechanism mostly identify correctly user's intentions which in turns successfully pruning irrelevant results. Whilst no such a classification approach employed in CVLCA algorithm, thus the existing approach may include irrelevant results in their answer set. Precision of XSearch (Li et al. 2007) is comparable to our approach in most of data sets, however, in XSearch (Li et al. 2007) users are required to submit the keyword query in forms of tag-keyword set. It means that the users have to manually specify predicates and return nodes which return results have to satisfy. In contrast, our proposed algorithm can handle this task automatically at the processing time. So, it eliminates the burden for users when using the system. Even though XSearch

#### 5.2.2 Search completeness

We employed the *recall* metric to measure the search completeness which indicates the fraction of all correct results actually captured in the returned answer. Figure. 5b demonstrates that our approach outperforms both XSearch (Cohen et al. 2003) and CVLCA (Li et al. 2007). We observed that XSearch (Cohen et al. 2003) and CVLCA (Li et al. 2007) fail to recognize meaningful fragments to a query which has more than one predicate with the same type, such as  $Q_5$  in Table 1. The reason, as analyzed in Section 2.2, is that both the existing approaches employed node labels to inferring node interconnections; meanwhile our approach proposed using node types for that aim. Overall, the advantages of our approach over all existing approaches can be demonstrated by employing F-measure for further evaluations.

#### 5.2.3 $\mathcal{F}$ -measure

To further evaluate the effectiveness of our approach and the state-of-art algorithms, we employed  $\mathcal{F}$ -measure which is the weighted harmonic mean of precision and recall.  $\mathcal{F}$ -measure is calculated as:

$$\mathcal{F} - \text{measure} = \frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times (\text{precision} + \text{recall})} \quad (1)$$

<sup>1</sup><http://www.oracle.com/technology/products/berkeley-db/index.html>

<sup>2</sup><http://dblp.uni-trier.de/xml/>

<sup>3</sup><http://www.sigmod.org/record/xml/>

<sup>4</sup><http://monetdb.cwi.nl/xml/>

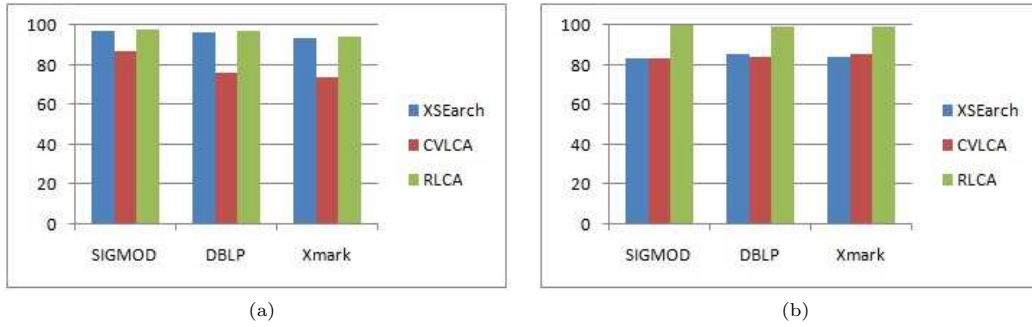
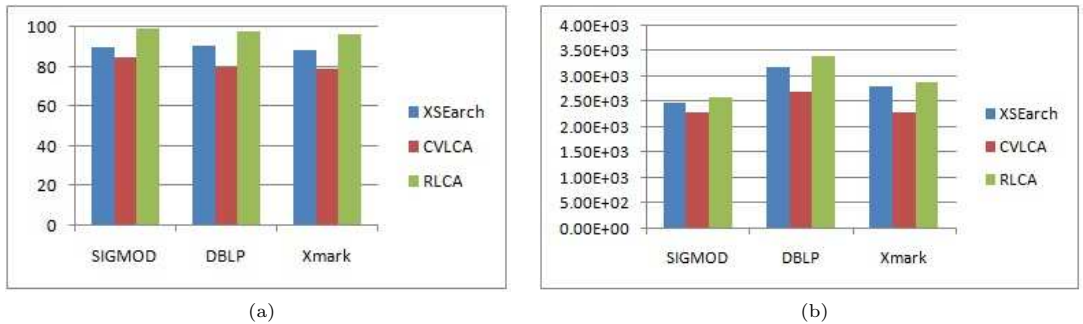


Figure 5: Precision and Recall.

Figure 6:  $\mathcal{F}$ -measure and Query processing time.

In our experiments, recall and precision are evenly weighted ( $\beta = 1$ ). The  $\mathcal{F}$ -measure of our approach and existing approaches are shown in Figure. 6a. We can see that the overall effectiveness of our algorithm outperforms the other algorithms. This result strengthens our thoughts that exploiting structural semantics of data in combination with semantics of query can improve the effectiveness and relevancy of XML keyword search.

### 5.3 Search Efficiency

The execution time of our approach compared with XSearch (Cohen et al. 2003) and CVLCA (Li et al. 2007) are presented in Figure. 6b. Our approach is quite slower than XSearch (Cohen et al. 2003). This is reasonable because in our approach query keywords are classified into predicates at the processing time. In contrast, the query in XSearch is already in form of predicates before the query is executed. In addition, our algorithm use more complicated evaluation procedures to guarantee that meaningful fragments (like Figure 6b) are not missed. CVLCA (Li et al. 2007) is the fastest one, however it treats a query as a set of words and semantics of query are not analyzed.

## 6 Discussions

We have reviewed the existing work of XML keyword search on identifying relevant matches in Section 2. In this section, we discuss several orthogonal studies which can be incorporated into our work.

**Efficiently Retrieving LCAs.** The problem of efficiently identifying the set of LCAs have been ex-

tensively studied in literature (Guo et al. 2003, He et al. 2007, Kacholia et al. 2005, Shao et al. 2007, Sun et al. 2007, Xu & Papakonstantinou 2005, 2008). Stack-based algorithm for efficiently evaluating LCAs has been proposed in (Guo et al. 2003). (He et al. 2007). It exploits a bi-level index in conjunction with bi-direction search (Kacholia et al. 2005) for pruning and accelerating the search. To reduce the index space, it partitions a data graph into blocks: The bi-level index stores summary information at the block level to initiate and guide search among blocks, and more detailed information for each block to accelerate the search within blocks. In (Xu & Papakonstantinou 2005, 2008), the characteristics of LCAs have been described to accelerate the evaluation.

**Ranking Schemes and Top-K.** The issue of ranking the results of keyword search over XML documents based on the relevant scores or user desired references has been studied in many existing work (Amer-Yahia et al. 2005, Guo et al. 2003, Lau & Ng 2008, Li, Feng, Wang, Yu & He 2008, Theobald et al. 2008). (Amer-Yahia et al. 2005, Li, Feng, Wang, Yu & He 2008) propose a scoring methods that are inspired by  $tf*idf$  and accounted for both content and structure of returned subtrees. In (Guo et al. 2003), the results of XML keyword search are ranked based on the ranking model adapted from PageRank hyperlink metric. A scoring model proposed in (Theobald et al. 2008) is an extension of the probabilistic-IR Okapi BM25 model. An adaptive ranking model has been studied in (Lau & Ng 2008), which can be adaptive to satisfy various needs and preferences in searching XML data. Whilst (Marian et al. 2005, Kimelfeld & Sagiv 2006) focus on efficiently identifying top-k relevant results for XML keyword search.



This paper focuses on the problem of effectively and accurately identifying the relevant results to XML keyword search. Other work discussed in this section are orthogonal issues and may be incorporated into our work.

## 7 Conclusions

In this paper, we have investigated the problem of keyword search over XML documents, with the aim of identifying the most relevant and meaningful fragments to XML keyword search. We proposed a novel approach namely Relevant LCA (RLCA) to accurately answer XML keyword queries. We firstly introduce the concept of equivalent types to capture a set of nodes which their types are equivalent, rather than using simple label of a node. Then we classified keywords in the query into conditions (or predicates) and return nodes. A fragment is meaningful only if it satisfies all predicates and return nodes in the query, not just only contains all query keywords. Finally, we introduced the semantics of a meaningful fragment as relevant results to the query. We have implemented our algorithms and the extensive experiments on real data sets have been conducted. The effectiveness of our algorithms, compared with the state-of-art existing proposals was measured carefully by three metrics: precision, recall and F-measure. The experimental results were analyzed in all three metrics and the results show that our approach achieves higher effectiveness than existing approaches.

## References

- Amer-Yahia, S., Koudas, N., Marian, A., Srivastava, D. & Toman, D. (2005), Structure and content scoring for xml, in 'VLDB '05: Proceedings of the 31st international conference on Very large data bases', VLDB Endowment, pp. 361–372.
- Bender, M. A., Farach-Colton, M., Pemmasani, G., Skiena, S. & Sumazin, P. (2005), 'Lowest common ancestors in trees and directed acyclic graphs', *Journal of Algorithms* **57**, 75–94.
- Cohen, S., Mamou, J., Kanza, Y. & Sagiv, Y. (2003), Xsearch: a semantic search engine for xml, in 'VLDB '2003: Proceedings of the 29th international conference on Very large data bases', VLDB Endowment, pp. 45–56.
- Guo, L., Shao, F., Botev, C. & Shanmugasundaram, J. (2003), XRank: ranked keyword search over xml documents, in 'SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data', ACM, New York, NY, USA, pp. 16–27.
- Harel, D. & Tarjan, R. E. (1984), 'Fast algorithms for finding nearest common ancestors', *SIAM J. Comput.* **13**(2), 338–355.
- He, H., Wang, H., Yang, J. & Yu, P. S. (2007), Blinks: ranked keyword searches on graphs, in 'SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data', ACM, New York, NY, USA, pp. 305–316.
- Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R. & Karambelkar, H. (2005), Bidirectional expansion for keyword search on graph databases, in 'VLDB '05: Proceedings of the 31st international conference on Very large data bases', VLDB Endowment, pp. 505–516.
- Kimelfeld, B. & Sagiv, Y. (2006), Finding and approximating top-k answers in keyword proximity search, in 'PODS '06: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems', ACM, New York, NY, USA, pp. 173–182.
- Lau, H. L. & Ng, W. (2008), 'A multi-ranker model for adaptive xml searching', *The VLDB Journal* **17**(1), 57–80.
- Li, G., Feng, J., Wang, J., Yu, B. & He, Y. (2008), Race: finding and ranking compact connected trees for keyword proximity search over xml documents, in 'WWW '08: Proceeding of the 17th international conference on World Wide Web', ACM, New York, NY, USA, pp. 1045–1046.
- Li, G., Feng, J., Wang, J. & Zhou, L. (2007), Effective keyword search for valuable lcas over xml documents, in 'CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management', ACM, New York, NY, USA, pp. 31–40.
- Li, Y., Yu, C. & Jagadish, H. V. (2008), 'Enabling schema-free xquery with meaningful query focus', *The VLDB Journal* **17**(3), 355–377.
- Liu, Z. & Chen, Y. (2007), Identifying meaningful return information for xml keyword search, in 'SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data', ACM, New York, NY, USA, pp. 329–340.
- Liu, Z. & Chen, Y. (2008), Reasoning and identifying relevant matches for xml keyword search, in 'VLDB '08: Proceedings of the 34th international conference on Very large data bases', pp. 921–932.
- Marian, A., Amer-Yahia, S., Koudas, N. & Srivastava, D. (2005), Adaptive processing of top-k queries in xml, in 'ICDE '05: Proceedings of the 21st International Conference on Data Engineering', IEEE Computer Society, Washington, DC, USA, pp. 162–173.
- Schieber, B. & Vishkin, U. (1988), 'On finding lowest common ancestors: simplification and parallelization', *SIAM J. Comput.* **17**(6), 1253–1262.
- Shao, F., Guo, L., Botev, C., Bhaskar, A., Chettiar, M., Yang, F. & Shanmugasundaram, J. (2007), Efficient keyword search over virtual xml views, in 'VLDB '07: Proceedings of the 33rd international conference on Very large data bases', VLDB Endowment, pp. 1057–1068.
- Sun, C., Chan, C.-Y. & Goenka, A. K. (2007), Multiway slca-based keyword search in xml data, in 'WWW '07: Proceedings of the 16th international conference on World Wide Web', ACM, New York, NY, USA, pp. 1043–1052.
- Theobald, M., Bast, H., Majumdar, D., Schenkel, R. & Weikum, G. (2008), 'Topx: efficient and versatile top-k query processing for semistructured data', *The VLDB Journal* **17**(1), 81–115.
- Xu, Y. & Papakonstantinou, Y. (2005), Efficient keyword search for smallest lcas in xml databases, in 'SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data', ACM, New York, NY, USA, pp. 527–538.
- Xu, Y. & Papakonstantinou, Y. (2008), Efficient lca based keyword search in xml data, in 'EDBT '08: Proceedings of the 11th international conference on Extending database technology', ACM, New York, NY, USA, pp. 535–546.



# An Analysis of Spreadsheet-Based Services Mashup

Dat Dac Hoang

Hye-young Paik

Boualem Benatallah

School of Computer Science & Engineering  
University of New South Wales,  
Sydney, NSW 2052, Australia,  
Email: {ddhoang, hpaik, boualem}@cse.unsw.edu.au

## Abstract

Spreadsheets, a popular productivity tool, has gained attention as a potential mashup development environment targeted towards end-users. In this paper, we present a general architecture of mashup tools for spreadsheets. We also present an analysis of the state-of-the art on spreadsheet-based mashup tools. The analysis result is used to guide our research in developing a lightweight semi-automatic mashup tool using spreadsheet paradigm.

**Keywords:** Web services, Mashup, Spreadsheet, End-user programming

## 1 Introduction

Mashup is a new application development method enabling users with little programming skills to create application by reusing and combining data, application functionalities and presentations from different sources. Recently, we witnessed a sharp rise of mashup tools and applications on the Web. In 2006 alone, there were hundreds of individual mashup tools released, Yahoo Pipes<sup>1</sup>, Google Mashup Editors<sup>2</sup>, Microsoft Popfly<sup>3</sup>, IBM QEDWiki<sup>4</sup>, just to name a few. Different mashup tools solve different problems so that when users are embarking onto the mashup journey, they need to pick the right tool for the right problem. For example, data mashup allows users to retrieve data from one or several data sources, process, mix the data and publish the result either as a feed or another data source. Process mashup allows users to automate processes by orchestrating services, forms and other resources in a workflow. Web page customization (i.e., presentation mashup) allows user to change web sites by removing elements, adding additional widgets or changing the user interfaces (Grammel & Storey 2008).

According to (Fischer et al. 2009), spreadsheet is one of the six programming paradigms for mashup development, along with integrated development environment, scripting languages, wiring (or piping) paradigm, programming by demonstration and automatic creation of mashup.

With spreadsheet-based mashup tool, spreadsheet users (e.g., office workers, professional accountants)

are provided an opportunity to access more information, making their job more efficient. Recently, some companies have already provided products for spreadsheet-based mashup tool (e.g., Strikelron, Extensio).

The motivations of using spreadsheet paradigm for mashup tools are elaborated as follow:

- First, spreadsheet is a widely used application with millions of users (Scaffidi et al. 2005). The popularity of spreadsheets can help increase the level of acceptance of any mashup tool built in the familiar environment.
- Second, spreadsheet is an intuitive data management, analysis and reporting tool with useful but simple-to-use functions, such as import, export, sort, visualize, etc. It could be considered as a suitable paradigm for data mashup since most of mashups are applications that reuse and combine data available from difference sources.
- Third, spreadsheet is a preferred programming environment for end users with the following features (Kandogan et al. 2005):
  - It offers a mixture of development environment and runtime environment facilitating immediate feedback to the users.
  - It supports incremental (i.e., step-by-step) development of an application.
  - It is more resilient and forgiving than most programming languages (i.e., an error in a cell only affects referring cells and does not affect the whole program causing a program crash).

In this paper, we present a qualitative survey on a set of spreadsheet-based mashup tools. We believe that understanding these mashup tools and other related issues will pave the way for research directions to effectively address some of the research problems (e.g., how users can take advantages of spreadsheet paradigm for developing mashups).

As an illustration of spreadsheet-based mashup, let us consider the following scenario<sup>5</sup>: Mary, a student, is learning Spanish. To study a word, she uses text-to-speech service (TTS), translation service (TSL), spell-checking service (SPL) and online workbook service (LOG). Instead of invoking these services separately and manually link the results, she opens a spreadsheet program, inputs an English word, writes formulas to link these services so that the completed formulas will now automatically suggest spelling corrections, pronounce the word in English, translate the word to Spanish, pronounce the word in Spanish then finally write the English and Spanish words to her

Copyright ©2010, Australian Computer Society, Inc. This paper appeared at the Twenty-First Australasian Database Conference (ADC2010), Brisbane, Australia, January 2010. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 104, Heng Tao Shen and Athman Bouguettaya, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

<sup>1</sup><http://pipes.yahoo.com>

<sup>2</sup><http://code.google.com/gme>

<sup>3</sup><http://www.popfly.com>

<sup>4</sup><http://services.alphaworks.ibm.com/qedwiki>

<sup>5</sup>This scenario is inspired by the scenario provided in (Obrenovic & Gasevic 2008)

online workbook. Obviously, the ability to quickly compose such services in spreadsheet enables her to create new applications that suit her needs and save her time, while interacting with a tool she is already familiar with.

Formula view		Evaluation view
A	B	B
1 User input	Orage	Orage
2 Spell check	=SPL(B1)	Orange
3 Text to Speech	=TTS(B2)	<TTS>
4 Translation	=TSL(B2)	Naranja
5 Text to Speech	=TTS(B4)	<TTS>
6 Logging	=LOG(B2,B5)	<LOG>
Worksheet 1		Worksheet 1

Figure 1: Motivating scenario

Mary's program is incrementally built. The "Formula view" in Figure 1 illustrates Mary's tasks. First, she specifies the location for user input by writing a label "User input" in cell A1 and the input word "Orage" (in incorrect spelling format) in cell B1. Then she invokes SPL service by putting a formula in cell B2 (i.e., =SPL(B1)) which takes the value of cell B1 as a parameter for SPL. She will get the spelling suggestion in cell B2 (i.e., "Orange") as shown in Evaluation view. Similarly, she writes the formulas in cell B3, B4, B5 and B6 step by step to get the voice from TTS, TSL and LOG. The mashup application is enacted when Mary changes the value in cell B1. Spreadsheet paradigm will automatically trigger the evaluation of formulas in dependent cells and produce results.

The remainder of this paper is organized as follows. Section 2 presents a general architecture of spreadsheet-based mashup tools. Section 3 introduces a number of mashup tools using spreadsheet paradigm. In sections 3 and 4, we discuss four dimensions used in our survey and evaluates tools according to the dimensions. We conclude the paper and have a discussion on future work in section 6.

## 2 General Architecture of Spreadsheet-Based Mashup Tools

When reviewing frameworks, it helps to have implementation architecture in mind. We describe here a general architecture of the spreadsheet-based mashup tools. Through the architecture, the readers should be able to picture how a mashup application is built in a spreadsheet. As depicted in the Figure 2, the architecture comprises of four elements:

- **Spreadsheet Interface (1).** This element is a conventional spreadsheet interface which includes a grid of cells and a formula editor. Cells are capable of storing values and formulas, while formula editor allows users to build mashup applications by specifying composition logic and layout information. The spreadsheet interface plays a role as the development environment in a mashup tool. For example, in the Mary's scenario, Mary writes her mashup application in spreadsheet interface using formula as illustrated in Figure 1.
- **Component Repository (2).** This element is a repository of all mashup components available

in the tool. Users choose an external resource (e.g., web service, file, database or application) and create a component that allows them to interact with the resource within the tool. This component can be assigned a friendly name (i.e., alias) for the spreadsheet formula editor. For example, in the scenario of Mary's, SPL is a component representing a SOAP-based service that suggests correct spelling of words. It is created based on the WSDL document of the spelling service.

- **Mashup Engine (3).** This key element is responsible for evaluating the formula (i.e., composition logic) and "wire" mashup components together. It operates in a centrally-mediated fashion and plays a role as a server to manage the execution flow among components. Since most of spreadsheet tools do not allow users to modify their formula evaluation mechanisms, mashup engine could be developed as an extension to spreadsheet evaluation engine. Mashup engine is also responsible for maintaining the formula evaluation context (i.e., the mashup result) and facilitate the reaction to cell modification by triggering the re-evaluation of dependent formula (i.e., upon a service invocation returns, the corresponding references need to be updated with the returning value). For example, in Mary's mashup application, whenever she change the input value in cell B1 (e.g., change "Orage" to "Lemoon" - both are in incorrect spelling format), the formulas in cells B2-B6 will be re-evaluated by the engine to create new results.
- **Wrappers (4).** Wrappers facilitate interoperability among resources which have different data formats (e.g., HTML, XML, RSS, etc) or use different access protocols (e.g., HTTP, SOAP-based, REST-based, etc). For example, we need different wrappers to create components to correctly serve the Mary's scenario, such as SOAP-based service wrapper for SPL service, REST-based service wrapper for LOG service or application specific wrapper for TTS service.

## 3 Spreadsheet-Based Mashup Tools

In this section, we introduce the spreadsheet-based mashup tools we studied for evaluation. For more detailed classification and description of these tools, readers are referred to (Hoang et al. 2009).

In these tools, the spreadsheet itself is a productivity tool which now equipped with functions to access external resources (e.g., services in the Web, files in desktop computer, databases, etc). Data from external resources are placed in cells, wired to each other to create a mashup application (e.g., output data of a service can be used in the input fields of another service).

We review the following tools: spreadsheet connectors for mashup engines, StrikeIron (Brauer n.d.), Extensio Extender<sup>6</sup>, A1 (Kandogan et al. 2005), AMICO-CALC (Obrenovic & Gasevic 2008), Husky (Srblic et al. 2007).

1. *Spreadsheet connectors for mashup engines.* JackBe Presto<sup>7</sup>, IBM Mashup Center<sup>8</sup>, and Kapow<sup>9</sup> are mashup engines which provide separate "spreadsheet connectors", allowing Mi-

<sup>6</sup><http://www.extensio.com/products/ExcelExtend.html>

<sup>7</sup>[http://www.jackbe.com/products/excel\\_connector.php](http://www.jackbe.com/products/excel_connector.php)

<sup>8</sup>[http://www.ibm.com/developerworks/blogs/page/etech?entry=mashup\\_center\\_import\\_for\\_microsoft](http://www.ibm.com/developerworks/blogs/page/etech?entry=mashup_center_import_for_microsoft)

<sup>9</sup><http://kapowtech.com/>

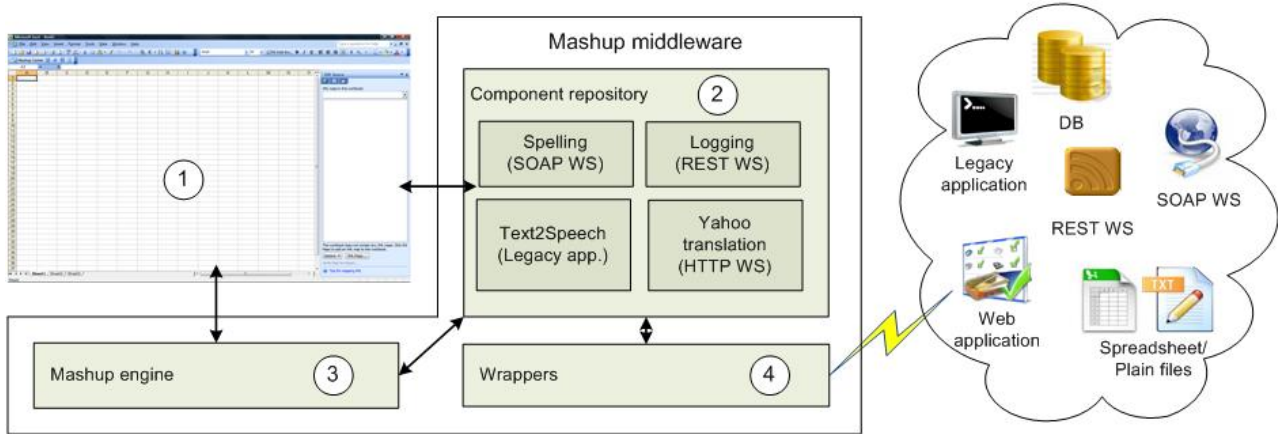


Figure 2: Common architecture of mashup tools using spreadsheet paradigm

Microsoft Excel spreadsheet users to consume existing mashup applications directly from their spreadsheets. These spreadsheet connectors allow the end-users to easily re-use already-built mashups (outside spreadsheets) in the spreadsheet environment. However, they are not fully-functional spreadsheet-based mashup tools in their own right.

2. *StrikeIron SOA Express for Excel and Extensio Extender for Microsoft Excel*<sup>10</sup> are commercial data mashup tools using spreadsheet paradigm. The basic idea of these approaches is that they allow the data contained in the web services to be pulled in Microsoft Excel workbook, “live” in cells, and integrated directly by users while still take advantages of all the analytical powers and flexibility of the spreadsheet tool. StrikeIron and Extensio Extender use SOAPful web services to create mashup applications by “hooking” the output value of one service with the input parameter of another service.
3. *AMICO:CALC* is an OpenOffice Calc extension that let users configure and connect services through a spreadsheet interface. Users manually write formula to compose the services in order to create a mashup. The execution is based on Adaptable Multi-Interface COmmunicator (AMICO) middleware platform for component integration. In AMICO:CALC, variables are considered as services in order to be used in spreadsheet. Users are provided some predefined functions to read value from and assign value to variables (i.e., to get data from and post data to services). For example, the expression *Amico\_Read*(“*spelling* – *suggestion*”) will receive a value when the spelling service finishes. AMICO provides a control panel tool where users can read description of variables exported by services, change their values or names.
4. *A1*, also known as Autonomic Task Manager for Administrators, is a research prototype from IBM. It facilitates a programming environment for system administrators. With A1, users not only use a spreadsheet-like environment with a task-specific language to access remote and heterogeneous systems but also gather, integrate status data and orchestrate control of different systems. A1 extends conventional spreadsheets by:
  - Allowing cells to contain arbitrary Java objects.

- Extending cell formulas to include calls to methods of cell objects.
- Allowing cells to contain procedural code blocks whose execution is triggered by events in the sheet.

5. *Husky* is a service composition tool. It extends the spreadsheet paradigm enabling users to intuitively express composition logic through a spatial arrangement of component services within spreadsheet cells.

## 4 Evaluation

We have installed and tried the tools mentioned in the previous section. Borrowing lessons from application integrations, we believe that a spreadsheet-based mashup framework needs the definition of four basic elements: component model, composition model, development environment, and runtime environment. Here, we summarize and compare the features across four dimensions: component model, composition model, development environment and runtime environment.

### 4.1 Component Model

This dimension determines the nature of components by means of the following properties:

#### 4.1.1 Component Data Model

A data model, as defined in (Ullman 1990), contains a notation for describing data and a set of operations used to manipulate that data. In our surveyed mashup tools, the data model extends spreadsheet data model to accommodate heterogeneous resources from different sources (e.g., web services, databases, files). Spreadsheet is a collection of cells organized in a tabular grid. Each cell is identified by its coordinates and contains either empty or atomic type values. Traditional spreadsheets have limited of atomic types, such as number, string or datetime. Some spreadsheets extend the atomic types so that the cells can contain complex types such as object or XML document. We identify two data models which are currently used in the mashup tools: *grid-based* and *object-based*.

In the grid-based model, data is presented in a range of cells and each cell contains an atomic data value (i.e., number, string, datetime). Using this data model, complex data need to be transformed into two-dimensional structure of the model. For example, in AMICO:CALC, a service invocation may return an

<sup>10</sup>We will call StrikeIron and Extensio Extender for short

XML document with complex structure (e.g., nested). In order to display the output, AMICO:CALC uses service adapters to linearize the complex structure to a set of variables, which can be easily mapped to spreadsheets cells and formulas. This transformation at the component data model level means that user does not have to deal with complex hierarchical structures.

In the object-based model, since the spreadsheet cell is extended to contain complex types, a complex XML document can be mapped to a single cell. For example, A1 extends conventional spreadsheets by allowing cells to contain arbitrary Java objects and formulas call to methods in cell objects. The following code snippet illustrates object-based data model used in A1:

```
C1: '149.171.225.4'
C2: JMX("http://example.net/IPtoCountry?wsdl")
C3: = C2.IP2Country(C1)
```

Cell C1 contains a string value represents an IP address. In cell C2, we use JMX (Java Management Extensions) binding to a SOAP service, namely IPToCountry. This web service has two operations: IP2Country() and Country2IP() which returns a country name of a given IP address and vice versa. In cell C3, we invoke IP2Country() operation of this web service and the result is returned as an XML document.

To illustrate the differences between two component data models, let us consider an example given in Figure 3 which shows an RSS feed of Google news service. The grid-based model is presented in Figure 3(a). The news, presented as an XML document, is mapped to the range of cells B1:C9. Figure 3(b) shows how the data could be presented using the object-based model. In which, the news data is mapped to cell B1. Each element of the news could be accessed by using “dot” notations in the mapping definition. For example, the “*channel language*” element can be put in cell B2 by using formula = *B1.language*.

#### 4.1.2 Component Access Model

Components are entities that allow users to interact with external resources inside a mashup tool. Resources can have heterogeneous data formats and use different access protocols. In this dimension, we investigate different data formats and access protocols supported by the mashup tools.

Some of the common data formats available in component services could be HTML, XML, RSS, Atom Syndication Format, JSON (Javascript Object Notation), spreadsheet, text file or RDF (Resource Description Framework).

The access protocols could be HTTP, TCP, UDP, SMTP, REST, SOAP, RPC or user-specific application protocols.

Most of the tools we studied provide support for REST and SOAP protocols using XML data format due to the prevalence of REST- and SOAP-based services. Some tools allow users to “mash” with other resources such as plain files, database, web applications or even legacy applications. This could greatly expand component resources from which users can make richer and better mashup application. For example, AMICO:CALC supports various communication protocols (e.g., TCP, UDP, XML-RPC, Open Sound Control, HTTP, SOAP, SQL and some application-specific interfaces such as Sesame RDF, WordNet, etc) for interconnecting services with different interfaces.

#### 4.1.3 Extensibility

Extensibility defines the ability of a mashup tool supporting users to define their own functionalities (e.g., creating components, defining new mashup operators). Normally expert users use a general purpose language (e.g., Java, C#) to achieve this extension.

For example, A1 allows users to extend the functionality of the tool to best suit their need by developing custom plug-in components. These plug-in components provide new interaction capabilities and functionalities and could be developed by creating new Java class extending A1Component class. The following code snippet illustrates an example of a user defined component named SampleComponent:

```
package com.ibm.a1.plugin.api.samples;
public class SampleComponent {
    String name;
    public SampleComponent(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

After deployed, in A1 spreadsheet’s cell users can create “*samples*” object and invoke its methods.

Extensibility can also be achieved by modifying existing functionality provided by the tool. For example, AMICO:CALC provides a generic adapter for services using TCP. Users can modify (or parameterize) this adapter to facilitate accessibility for a specific application using TCP protocol (e.g., a game).

### 4.2 Composition Model

This dimension determines how components can be glued together through the following properties:

#### 4.2.1 Mashup Strategy

This dimension identifies the mashup strategy used in spreadsheet mashup tools. We classify mashup strategy into two categories:

- The first category concerns the time when a mashup application is created. This could be at design time or run time. *Static* mashup takes place during the design time when users plan the logic and components to be used in the mashup application. Components are selected, linked together and finally compiled and deployed. This strategy is suitable with a scenario when components and services are stable (e.g., in term of availability) and rarely change (e.g., data structure).

When the component providers frequently change the structure of data or business logic, static mashup is too restrictive and requires users to re-build mashup application adapting to the changes. *Dynamic* mashup may solve this problem by adapting to unpredictable changes at the run time. The web environment is highly dynamic where new services become available and change frequently. Ideally, mashup application should be able to transparently adapt to environment changes, and to adapt to customer requirements with minimal user intervention.



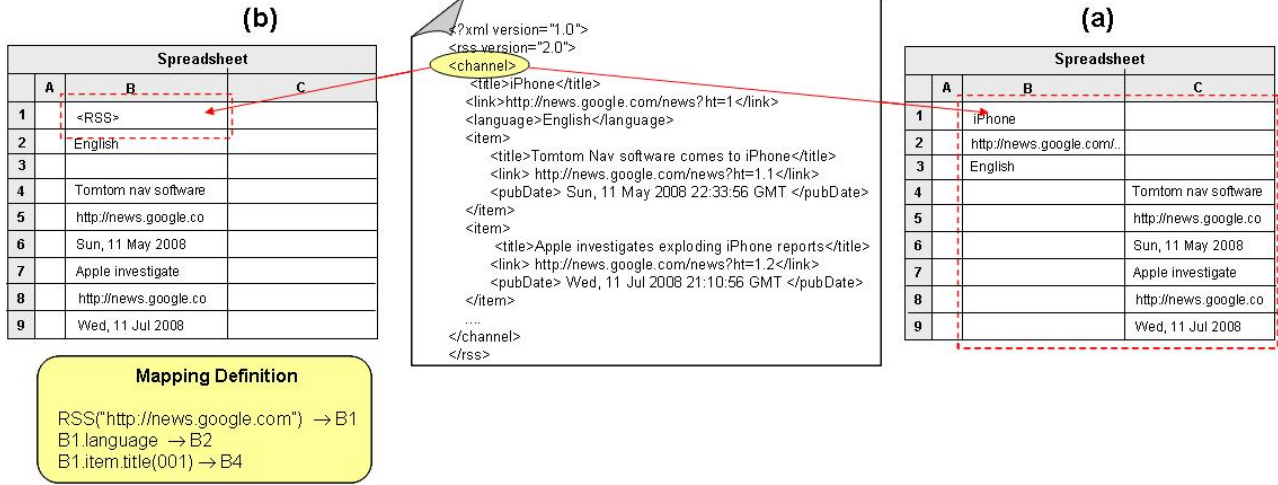


Figure 3: Component data model

- The second category concerns the way users build mashup application which could be either *manual* or *automatic*. In *manual* mashup, users manually describe the composition logic (i.e., data flow and control flow) for mashup application. This makes the development process of mashup application tedious, error-prone and time consuming. On the contrary, automatic mashup does not require users to be involved in low-level development process. Instead, users only need to provide high level requirements or goals of mashup application. The mashup tool will automatically suggest or choose composition plan based on given context or semantic of application and components.

#### 4.2.2 Orchestration Style

Orchestration describes the arrangement, coordination and management of components in mashup application. It contains business logic of mashup application specifically about execution order of components. There are three orchestration styles could be used in spreadsheet paradigm, namely *flow-based*, *layout-based* and *event-based*:

- Flow-based.** In flow-based orchestration style, mashup applications are built by using natural data flow and control flow created by cell dependencies in spreadsheet (i.e., linking one service's output into another service's input fields). For example, as shown in "Formula view" in Figure 1, Mary's scenario can be implemented by "hooking" output of SPL service in cell B2 with input field of TSL service in cell B4. This orchestration style is naturally supported by spreadsheet paradigm and easy to use. However, as discussed in (Hoang et al. 2009) it has limitations when users need to build a mashup application with complex control flow logic.
- Layout-based.** In layout-based orchestration style, users are allowed to explicitly define the execution order of components in mashup application by organizing spatial position of components. For example, in Husky, the execution order of mashup application is defined by spatial organization of services in a grid of cells. Husky transforms the spatial organization of services' events into their ordering in time. Time progresses from left to right and from top to bottom in cell blocks. A set of adjacent cells makes a sequence of events, while empty cells disjoin

the workspace into temporally independent event sequences. Let consider Mary's scenario implemented by using Husky as shown in Table 1 below: The execution flow is defined by a sequence of services in adjacent cells from C2 to C7. The conditional split defined in cell C3 will trigger the operation in cell C4 if the content of cell C2 is not empty (i.e., like "goto" or "jump" statement in some programming languages). The rest of the program is evaluated according to the following order: evaluate formula in C5, C6 and finally C7. By explicitly defining the control flow, this orchestration style gives the users the benefits of generality and extensibility. However, it may break the philosophy of spreadsheet programming and increase the learning curve.

- Event-based.** In event-based orchestration style, the execution flow of mashup application is determined by *user actions* (e.g., button clicks, key presses, etc) or *interactions with other services* (i.e., the completion of an action in one service may cause an action in other service). For example, in the A1 system, each cell is an arbitrary Java object. Each object is associated with a listener which is designed to manage events. The listener explicitly triggers the execution of a code block based on a condition. There are two listener constructs: *on()* is used to trigger an operation based on events and *when()* is used to trigger an operation based on boolean expression. Mary's scenario can be implemented in A1 as shown below in Table 2:

Services definitions are defined from cell B1 to

	A	B
1		<i>SPLservice</i> ("address")
2		<i>TTSservice</i> ("address")
3		<i>TSLservice</i> ("address")
4		<i>WorkbookService</i> ("address")
5		<i>Orage</i> ;
6		<i>Button</i> ();
7		<i>on</i> (B6) <i>B1!Spell</i> (B5);
8		<i>when</i> (B7 <> "") <i>B2!VoiceEN</i> (B7);
9		<i>when</i> (B7 <> "") <i>B3!E2S</i> (B7);
10		<i>on</i> (B9) <i>B2!VoiceSP</i> (B9);
11		<i>on</i> (B10) <i>B4!Send</i> (B7, B9);

Table 2: Event-based orchestration style in A1

cell B4. The program starts when user clicks

	A	B	C
1	Orage		
2	http://spl.com		Execute[A2] "GetSpell" [A1]
3			If [C2]<>" " Then Set Clock To [C4]
4	http://tts.com		Execute[A4] "Voice_EN" [C2]
5	http://tsl.com		Execute[A5] "Translate" [C2]
6			Execute[A4] "Voice_SP" [C5]
7	http://log.com		Execute[A7] "Send" [C2] [C5]

Table 1: Layout-based orchestration style in Husky

on the button defined in cell B6. Based on the user-generated event (i.e., button clicked), the formula in cell B7 returns a spelling suggestion. Based on the value of cell B7 (i.e., the output of spelling service), two formulas in cell B8 and B9 are evaluated. Similarly, the execution of formula defined in cell B10 and cell B11 are based on events generated by the execution of formulas in cell B9 and B10, respectively.

#### 4.2.3 Data Passing Style

This dimension identifies how the data is passed between components. There are two data passing styles:

- *Data Flow.* In this style, data is passed from one component to another component. For example, in Mary's scenario, the data from output of SPL service in cell B2 is directly transferred to input field of TSL service in cell B4. This is the default data passing style of spreadsheet applications.
- *Blackboard.* In this style, data is read from and written to a shared data repository (e.g., variable). For example, AMICO:CALC uses variables (which encapsulate services' data structure) to abstract the heterogeneity of services. Users are provided with various predefined functions (e.g., *Amico\_Read*, *Amico\_Write*, etc) to get data from and post data to services. The formula *Amico\_Read*("SPL") will receive a value when the spelling services finished. *Amico\_Write*("SPL", B1) will assign value in cell B1 to the variable SPL. This data passing style is used mainly in programming languages.

The dominant data passing style using in the evaluated tools is data flow. This could be explained by the fact that the data flow model is easy to use and exists in most of mashup tools (e.g., Yahoo pipes) and is naturally support by spreadsheet paradigm.

### 4.3 Development Environment

This dimension identifies the characteristics of the development environment offered by a mashup framework through the following properties:

#### 4.3.1 Target Users

We identify three types of users in spreadsheet-based mashup tools: developer, skilled user and novice user. A developer, who is the most skillful user, should be familiar with programming, web technologies, different APIs as well as the usage of mashup tools. A skilled user has no programming skills but does have detailed functional knowledge about some technologies and a specific mashup tool. Novice user only has knowledge on the functionality of spreadsheet and be able to use some simple spreadsheet formulas and operations (e.g., make a cell reference, copy and paste the content of a cell).

Most of the tools target at spreadsheet users who have limited programming skills. For example, users of Extensio Extender do not need to spend too much learning effort in order to use the tool since the programming metaphor of the tool is similar to spreadsheet programming style. Some tools extend the spreadsheet paradigm to accommodate new orchestration style and require users to learn a specific formula syntax in order to create a mashup application. For example, in A1, the user needs to learn how to write procedural codes within a cell formula. Or users need to learn the rule for spatial arrangement of services as well as four special objects (namely, Clipboard, Queue, TokenCenter and BrokerCenter) in Husky spreadsheet in order to define flow control of service invocations.

#### 4.3.2 Search-ability

Search-ability expresses the ability of a mashup tool helping users in finding desired components (or even mashup application) created by others. There are three types of search:

- *Text/Keyword-based.* Users can search on the title, description or tags of components by specifying keywords.
- *Browsing.* The tool displays a list of components in a typical file-finder or folder-subfolder fashion.
- *Context-specific suggestion.* The tool provides (or suggests) the needed component to users depending on the users' context without having to search.

Within the evaluated mashup tools, there is no tool that offers context-specific suggestion to the users. Having context-awareness could lead to the possibility of having a framework that relies on the context and incrementally guiding users in developing mashup applications. Browsing capability could make users feel conformable when they want to find desired components since it classifies components and services into different categories in a conventional manner. For example, StrikeIron provides browsing capabilities to its "Web service Marketplace" which has different categories (e.g., Communication, Financial, Marketing, E-Commerce, Utilities, etc). Text-base searching is useful when users know the name or description of components. AMICO:CALC provide the text-based searching capability for finding components in its middleware control panel.

#### 4.3.3 Community Features

One of the most important factors that could lead to the success of an end-user development tool is community support (Nardi 1993). Some of the desirable community support features are:

- *Sharing and reusing.* Components and mashup applications are created by one user but they can be run, copied or modified by other users.

- *Collaborating.* Users can co-operate to create or modify mashup applications.
- *Discussion.* A framework is provided with a forum or chatting capability for users to exchange ideas, help each other in order to build a community.

To support collaboration among users, application in A1 can be deployed as portlets in a J2EE-based web portal server. In the portal, a user can execute or customize program deployed by other users. This is possible because all A1 spreadsheet is created in Java Swing-based client application which can be launched within the web portal framework using Java Web-Start. Users can either save spreadsheet locally for personal use or to server repository for sharing with others. Extensio Extender and StrikeIron also support community features (e.g., sharing and discussion).

#### 4.3.4 Software Engineering Aspects

This dimension concentrates on the software engineering aspects including the following aspects:

- *Debugging.* Users can locate and fix errors (including composition logic error, syntax error, etc).
- *Testing.* Users can investigate the quality of mashup application with respect to the context in which it is intended to operate.
- *Version control.* Users can manage the changes and multiple users can access and modify a mashup application.

The surveyed mashup tools mainly focus on offering functionalities for creating mashup applications. The overall support for software engineering aspects such as debugging and version control is very limited. There is only AMICO:CALC supports debugging capability. In AMICO:CALC, debugging is enabled through the use of middleware control panel for monitoring all the variables that are exchanged among services.

#### 4.4 Runtime Environment

This dimension expresses how the results of mashup are delivered to users.

##### 4.4.1 Mashup Engine

We identify two types of mashup engine could be used in the spreadsheet-based mashup tool: (i) mashup engine that based on spreadsheet evaluation engine; and (ii) mashup engine that developed as a plug-in to spreadsheet application. On one hand, spreadsheet evaluation engine could be used for data mashup by using natural data flow created by cell dependencies in spreadsheet. The mashup application executed using this mashup engine must use spreadsheet programming metaphor (e.g., using formula, cell referencing, etc). For example, in StrikeIron, a very simple data mashup can be implemented by linking one service's output into another service's input fields. On the other hand, mashup engine can be developed separately from spreadsheet evaluation engine. The mashup application does not need to conform to spreadsheet programming metaphor. Tools which use layout-based and event-based orchestration styles usually have this type of mashup engine.

##### 4.4.2 Execution Type

We identify three types of execution could be used in mashup tools, namely *centralized*, *distributed* and *hybrid*. Centralized execution is similar to the client-server paradigm. In this case, the server is the central scheduler that controls the execution of the component services in a mashup application. The distributed paradigm in contrast expects the services to share their execution context. Each component service has its own coordinator, which has to collaborate with the coordinators of the other services, to guarantee a correct ordered execution. Hybrid form of the distributed and centralized paradigms may be coordinator that controls not only one but a set of web services (Benatallah et al. 2003).

All of surveyed mashup tools have centralized execution type. For example, the execution of mashup program in AMICO:CALC is based on a middleware named Adaptable Multi-Interface COMMunicator which is a centralized platform facilitates adaptation, abstraction, and mediation for diverse service interfaces. The middleware maintains a list of variables which encapsulate data structures used by services and can be easily mapped spreadsheet cells.

##### 4.4.3 Exception and Transaction Handling

Exception handling is a collection of mechanism supporting the detection, signaling and after-the-fact handling of unusual events whether erroneous or not (Burnett et al. 2000). In programming languages, exception handling is considered as necessity part and exists in almost all of application development tools. However, existing spreadsheet mashup tools have very limited support for exception handling. In spreadsheet, exception handling is compatible with reasoning model of spreadsheet formula. For example, in Microsoft Excel, when an operation detects an exception, it will return one of seven possible error values (e.g., *NULL*, *DIV/0*, *VALUE*, *REF*, *NAME*, *NUM*, *N/A*). The error value model used in spreadsheet is different from the use of status flag in traditional programming language in the sense that it can be ignored so that the error may not cause the whole "spreadsheet program" to crash.

A transaction is an atomic operation which may not be divided into smaller operations. Transaction handling is a mechanism supporting the ACID (i.e., Atomic, Consistent, Isolated, and Durable) properties of transactions. In our surveyed mashup tools, transaction handling is not supported.

#### 5 Summary of Evaluation

So far, we have discussed different dimensions using in this survey. In this section we discuss our analysis result and summarize it in Table 3.

The majority of tools have grid-based component data model. This design choice can be explained by the fact that using grid-based model there is no need to extend traditional spreadsheet data model to present data. The issues of how to map structured data to spreadsheet data model have been addressed by existing works. For example, (Kongdenfha et al. 2008) provides a set of widgets for presenting complex data in spreadsheet paradigm, namely content, repeater, hierarchical, index. (Brauer n.d.) provides a drag and drop interface for manually mapping data to target location in spreadsheet. Object-based data model is more sophisticated since it encapsulates object type in a cell. However, this data model requires more advanced knowledge from users and is more suitable with skilled users and developers than

novice users. This is the reason of why there are only a few number of tools used object-based model.

In order to facilitate the mashup capability with a specific resource (e.g., a specific application), user need to write their own wrapper. Within reviewed tools, only A1 and AMICO:CALC allow users to extend Java class to write plug-ins/adapters for developing new functionality of the tool. This could not be suitable with all users since it requires users to have programming skill.

All of the analyzed tools use static and manual mashup strategies. Mashup programming using these strategies is tedious, error-prone and time consuming while the mashup application is incapable of adapting to requirement changes. This limitation leads to the need to have an automatic and dynamic mashup tool supporting users in creating mashup applications.

Flow-based orchestration style is used in most of the tools since it conforms to natural spreadsheet programming metaphor (i.e., cell referencing). Layout-based orchestration is rarely used in mashup tools since it may break the functional programming nature of spreadsheet (i.e., it introduces procedural programming notions within spreadsheet). Event-based orchestration is based on the Event-Condition-Action form (i.e., on `< event >` if `< condition >` then `< operation >`) and is preferred programming model for programmers.

In summary, the reviewed mashup tools have provided many useful features for creating mashup applications. However there are plenty of rooms for further improvement.

## 6 Conclusion and Future Work

Mashup is an application development method which can be done in a lightweight manner to mix information and automate processes. There has been a plethora of mashup tools in many shapes and forms. In this paper, we analyze a number of spreadsheet-based mashup tools. We believe the spreadsheet environment has inherent advantages over other mashup environment due to its popularity and familiarity with the users.

We attempted to make a comparison among existing spreadsheet-based tools by finding common dimensions and characteristics. We hope that the analysis will be useful for the researchers and developers in this area to understand the architecture of spreadsheet-based mashup tools, the strengths and weaknesses of the current approaches.

The main limitation of our work is that we conducted a qualitative instead of quantitative survey, therefore, some of the observations could inherently be subjective. However, the results are only used to develop an initial understanding and create a foundation for further research. More rigorous surveys with quantitative measures are planned to add objectivity to the study.

Typically, components in our surveyed tools are manually created and composed at design time with user interactions. However, requirements and situations of the tasks in our daily life may change at any time so that a mashup tool need to be able to transparently adapt to environment changes, adapt to customer requirements with minimal user intervention. None of the reviewed tools can do this. This problem leads to an unsatisfied support to the users which can prevent them from being productive.

The goal of our future work is on alleviating aforementioned pain. We adopt the notion of context to be used in our framework facilitating lightweight semi-automatic mashup. Context is the information that characterizes the interactions between humans, ap-

plications and the surrounding environment. For example, *location* (e.g., geography coordinates, country, time), *user information* (e.g., name, address, email) or *client context* (e.g., hardware, software). Our future work will concentrate on achieving the following targets:

## A Lightweight Programming Model

Mashup applications are often created manually by combining available components with certain predefined operators. With the increasing demand for components, operators and functionalities, the user's task to efficiently select, combine and configure components becomes more complex, time-consuming and error prone even for experienced users.

The first target of our framework is to simplify the programming process of mashup. Mashup framework should leverage the spreadsheet model which combines functional and visual approach to deliver an intuitive environment with little or no learning barriers (Brad et al. 2004).

## Semi-automatic Mashup

Most of the current mashup approaches use manual composition and full automatic support is still the target of ongoing research activity. Our second target is to provide a semi-automatic mashup framework with user intervention. In order to provide such a framework, users' requests and components need to be stated in a way that rich semantic information can be incorporated when components are queried. We try to embed intelligence into the manual process of mashup: service selection, binding and composition through the smart use of context. We intend to construct a *user model* that reflects user preferences, such as usage characteristic, interest, expertise and *community model* that measure the interest of users on a mashup or component. Based on these two models, our framework incrementally guides users on the process of building mashup application. In addition, when a requirement changed (e.g., interest changed) the system will automatically suggest new mashup plan to the users. Our research prototype will have two basic components: a composer and an inference engine. The inference engine stores the information about known component, user model and community model in its Knowledge Base (KB) and has the capability to suggest composition plans. The composer has a spreadsheet grid interface that enable users to incrementally build their mashup. The users start the composition process by selecting preferred profile and one component service. A query is sent to the inference engine to get the suggestion on possible mashup plans. The composers then get the results and display the plans for user to select.

## References

- Benatallah, B., Sheng, Q. Z. & Dumas, M. (2003), 'The self-serv environment for web services composition', *IEEE Internet Computing* 7(1), 40–48.
- Brad, A. K., Ko, A. J., Myers, B. A. & Aung, H. H. (2004), Six learning barriers in end-user programming systems, in 'IEEE Symposium on VL/HCC 2004', pp. 199–206.
- Brauer, B. (n.d.), 'Next evolution of data integration into microsoft excel'.  
URL: <http://www.strikeiron.com/>
- Burnett, M., Agrawal, A. & van Zee, P. (2000), 'Exception handling in the spreadsheet



		Excel connectors	StrikeIron	Extensio Extender	AMICO:CALC	A1	Husky
Data model	<i>grid-based</i> <i>object-based</i>	+	+	+	+	+	+
Component access model	Protocol <sup>a</sup> Data format <sup>b</sup>	$P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}$ $D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8$	$P_6$ $D_2$	$P_1, P_2, P_3, P_4, P_6, P_7, P_8, P_9, P_{10}$ $D_1, D_2, D_3$	$P_1, P_2, P_3, P_6, P_7, P_8, P_9, P_{10}$ $D_1, D_2, D_3, D_6, D_8$	$P_1, P_4, P_8$ $D_1, D_8, D_5, D_7, D_8$	$N/A$ $D_1, D_2$
Extensibility		-	-	-	+	+	-
Mashup strategies	<i>static</i>	+	+	+	+	+	+
	<i>dynamic</i>	-	-	-	-	-	-
	<i>manual</i>	+	+	+	+	+	+
	<i>automatic</i>	-	-	-	-	-	-
Orchestration styles	<i>flow-based</i>	+	+	+	-	-	-
	<i>layout-based</i>	-	-	-	-	-	+
	<i>event-based</i>	-	-	-	-	+	-
Data passing styles	<i>dataflow</i>	+	+	+	-	+	+
	<i>blackboard</i>	-	-	-	+	-	+
Target user	<i>developer</i>	-	-	-	-	-	-
	<i>skilled user</i>	-	-	-	+	+	+
	<i>novice user</i>	+	+	+	-	-	-
Searchability	<i>text-based</i>	-	+	+	-	-	-
	<i>browsing</i>	+	+	+	+	-	-
	<i>context suggestion</i>	-	-	-	-	-	-
Community features	<i>sharing</i>	-	+	+	-	+	-
	<i>collaborating</i>	-	-	-	-	+	-
	<i>discussion</i>	-	+	+	-	-	-
Software engineering aspects	<i>debugging</i>	-	-	-	+	-	-
	<i>testing</i>	-	+	+	-	+	-
	<i>version control</i>	-	-	-	-	-	-
Execution engine	<i>spreadsheet</i>	+	+	+	-	-	-
	<i>external</i>	-	-	-	+	+	+
Execution types	<i>centralized</i>	+	+	+	+	+	+
	<i>distributed</i>	-	-	-	-	-	-
	<i>hybrid</i>	-	-	-	-	-	-
Transaction & exception handling		-	-	-	-	-	-

(+) means the dimension is directly supported; (-) means the dimension is not supported; (N/A) means the dimension is not mentioned in related publication.

<sup>a</sup>  $P_1=HTTP$ ;  $P_2=TCP$ ;  $P_3=UDP$ ;  $P_4=SMTP$ ;  $P_5=REST$ ;  $P_6=SOAP$ ;  $P_7=RPC$ ;  $P_8=application\ specific$ ;  $P_9=ODBC$ ;  $P_{10}=JDBC$ ;

<sup>b</sup>  $D_1=HTML$ ;  $D_2=XML$ ;  $D_3=RSS$ ;  $D_4=ATOM$ ;  $D_5=JSON$ ;  $D_6=XLS$ ;  $D_7=RDF$ ;  $D_8=Text$ ;

Table 3: Evaluation results

- paradigm', *IEEE Transactions on Software Engineering* **26**(10), 923–942.
- Fischer, T., Bakalov, F. & Nauertz, A. (2009), An overview of current approaches to mashup generation, in 'Wissensmanagement', pp. 254–259.
- Gammel, L. & Storey, M.-A. (2008), An end user perspective on mashup makers, Technical Report DCS-324-IR, University of Victoria.
- Hoang, D. D., Benatallah, B. & Hye-young, P. (2009), Towards a spreadsheet-based service composition framework, in 'PhD workshop, DASFAA 2009'.
- Kandogan, E., Haber, E., Barrett, R., Cypher, A., Maglio, P. & Zhao, H. (2005), A1: End-user programming for web-based system administration, in 'UIST '05', ACM, New York, NY, USA, pp. 211–220.
- Kongdenfha, W., Benatallah, B., Saint-Paul, R. & Casati, F. (2008), Spreadmash: A spreadsheet-based interactive browsing and analysis tool for data services, in 'CAiSE '08', Springer-Verlag, Berlin, Heidelberg, pp. 343–358.
- Nardi, B. A. (1993), *A Small Matter of Programming: Perspectives on End User Computing*, The MIT Press.
- Obrenovic, Z. & Gasevic, D. (2008), 'End-user service computing: Spreadsheets as a service composition tool', *IEEE Transactions on Services Computing* **1**(4), 229–242.
- Scaffidi, C., Shaw, M. & Myers, B. (2005), 'Estimating the numbers of end users and end user programmers', *VL/HCC '05* **0**, 207–214.
- Srblic, S., Zuzak, I. & Skrobo, D. (2007), 'Husky editor'. <http://www.husky.fer.hr/>.
- Ullman, J. D. (1990), *Principles of Database and Knowledge-Base Systems: Volume II: The New Technologies*, W. H. Freeman & Co., New York, NY, USA.



# Optimizing XML Data with View Fragments

Jun Liu<sup>1</sup>Mark Roantree<sup>1</sup>Zohra Bellahsene<sup>2</sup>

<sup>1</sup> Interoperable Systems Group  
Dublin City University,  
Dublin, Ireland

Email: jliu, mark.roantree  
@computing.dcu.ie

<sup>2</sup> Computer Science at the University of Montpellier II,  
161 rue Ada, F34392 Montpellier Cedex 5,  
Email: bella@lirmm.fr

## Abstract

As web-based applications and data continue to grow, large caches of XML data will result in many application domains. In sensor web applications, there are continuous streams of sensor data being generated, converted to XML and stored for domain queries and data mining purposes. The main problem with these XML caches is that existing XML database queries are very slow, especially for large databases or those with complex structures. In this work we propose a view-based system to XML optimization where the most popular or well-chosen queries are materialized and fragmented to greatly improve the performance of all XML queries.

**Keywords:** XPath View, multi-fragment, View Adaptation

## 1 Introduction

The Sensor Web is a natural extension to the WWW where small hardware devices are used to automatically generate new data. In our research area, we are monitoring a group of elite athletes using various personal monitors. The collecting, processing and integration of data was described in earlier research (Roantree et al. 2008). However in this work, we made no attempt to optimize XML queries and instead focuses on managing the sensor data.

Web-based data interfaces are used by athletes to upload this data where it is converted to XML, semantically enriched and integrated with other data streams. Databases are then queried using standard XML languages such as XPath or XQuery to query and mine these XML stores. One of the problems facing scientists and end-users is that XML queries perform poorly when compared with other database systems. There have been many approaches to XML query optimization where SQL-based optimizers are used (Boncz et al. 2006, Marks & Roantree 2009), advanced tree-structured indexes are created to prune search space (Bruno et al. 2002) or XPath axis navigation algorithms (Grust 2002).

Our solution to this problem is to adopt an existing approach to optimization from the relational database world where queries are precomputed and stored as views on data servers. Subsequent queries make use of these

views in order to execute and generate query results in far quicker times. The challenge is to make a solution for table based data stores applicable in the tree-based world of XML data.

## 1.1 Paper Structure

This paper is structured as follows: in the remainder of this section we provide the motivation for this work and outline our contribution to research in this area followed by a detailed discuss of related research in §2; we then provide a brief overview of the XML view language together with a set of sample views in §3; in §4 we introduce our *Multiple Fragment Materialization (MFM)* view graph together with a detailed description of its constructs, fragments and operators; we give an example of how the *MFM* graph is constructed using the sample XML views, and a summary of the transformation between view expressions to *MFM* view graph in §5; in §6 we show the experimental results with a comparison of our approach against the traditional full materialization approach; and finally §7 concludes the paper and outlines future work.

## 1.2 Motivation

In relational database management systems (RDBMS), materialized views are widely used for query result caching and query answering. The impact on using materialized views for query answering is significant to the improvement of query performance. Materialized views are considered to be most beneficial where systems are queried more often than updated such as in data warehouses. As a result, views have also been studied in XML database management system (XDBMS). XML views are usually formed by a subset of the XPath language denoted by  $XP\{\emptyset, *, //, /, \}$ . In the context of XDBMS, materialized XPath views can also be used to expedite the processing of XML queries. (Arion et al. 2007, Balmin et al. 2004, Lakshmanan et al. 2006) studies the single view-based query answering problem. Whereas (Cautis et al. 2008, Gao et al. 2007, Tang et al. 2008, 2009) focus on using multiple materialized XPath views for answering queries.

In addition, there has been increasing attention on the issue of XPath view updates and maintenance. (Sawires et al. 2005, Lim et al. 2003) deal with synchronizing the materialized view data with updates to the source data. However, there is very little work that has been done for handling updates when view definitions are changed. This problem is referred to as the *view adaptation* problem, which was first introduced by Gupta, et al (Gupta et al. 1995) in RDBMS. Padmapriya et al. (Ayyagari et al. 2007) claimed to be the first to focus on XML view adaptation using access rules. However in their system, view adaptation can only take place below the output node of the XPath expression or it must retrieve the data from source. This is due to the fact that only XML

Funded by Enterprise Ireland Grant No. CFTD/07/201

Copyright ©2010, Australian Computer Society, Inc. This paper appeared at the Twenty-First Australasian Database Conference (ADC2010), Brisbane, Australia, January 2010. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 104, Heng Tao Shen and Athman Bouguettaya, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

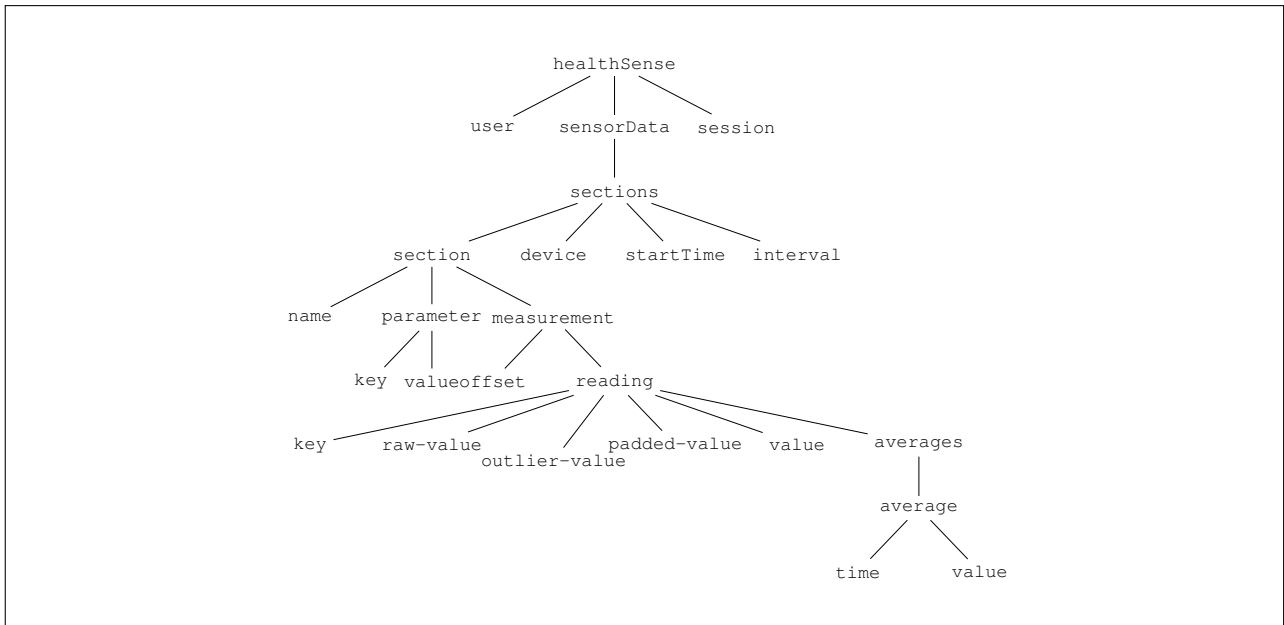


Figure 1: Sensor Database Schema

data fragment below the output node is materialized as depicted in (Tang et al. 2008). Furthermore, data reusability is rather poor as materialized data fragments cannot be shared across views.

The view adaptation problem is well understood in relational database systems. A solution by Bellahsene (Bellahsene 2004) proposed a multi-fragment based approach, where view materialization takes place on the fragment level. Here, data reusability is significantly improved as fragments are shared between different views. However, in order to apply this approach in an XML database, a query transformation approach is required to convert XML query expressions (XPath or XQuery) into SQL expressions. Since the difference between XML query languages (nested, irregular, heterogeneous and ordered) to the SQL language (flat, regular, homogeneous and unordered), the language transformation will be difficult. We adopted the idea of the fragment approach and applied it to XML databases. Our multi-fragment view materialization approach improves the data reusability by sharing common sub-expressions among views in the form of view fragments. In this way, any change takes place in view definition above the output node can be resolved by searching for materialized data fragments across different views.

### 1.3 Problem Description

The schema illustrated in Figure 1 represents a subset of the **HealthSense** dataset, compiled in our project. We now present a list of views, expressed in XPath, which form part of our materialized fragments. Each are based on the *reading* node (*outlier*, *padded*, *value* and *average*) between specified times (determined by timing offsets). We also provide a simple explanation for those readers unfamiliar with XPath.

#### Example 1

Find all **reading** data from **healthsense measurement** recorded by the sensor device where the **offset** exceeds 10 seconds.

```
//healthSense//measurement[./offset>10000]/reading
```

This XPath view materializes all subtrees rooted at nodes named **reading**, where each **reading** node must have a parent node called **measurement** with child **offset**

having values greater than 10000 (as values are recorded in milliseconds).

#### Example 2

Locate sensor **reading** nodes where **offset** exceeds 5 seconds.

```
//healthSense//measurement[./offset>5000]/reading
```

This XPath view materializes all subtrees rooted at the node **reading**, where each must have a parent node called **measurement** with a child **offset** with value great than 5000.

#### Example 3

Find sensor data with **offset** values between 5000 and 10000 milliseconds.

```
//healthSense//measurement[./offset>5000][./offset<10000]/reading
```

Example 1 demonstrates an existing XML view that stores all measurement **reading** data recorded after 10000 milliseconds. If we modify the definition of the view in Example 1 by replacing 10000 to 20000. The answer to this new query can be computed easily from the data that has already been stored in Example 1. This can be achieved by removing all reading data that were measured before 20000 milliseconds. This incremental computation is much more efficient than recomputing the view from scratch.

However, the change to the view definition is not always so easily computable. Assuming that we modify Example 1 by changing the offset time to 5000, e.g., Example 2, then the reading data measured between 5000 and 10000 are not computable using Example 1. Nevertheless, we can still reuse the old view (see Example 1) for all reading data after 10000 milliseconds, and then all the rest from the base data, XML database.

Finally, assume that Example 3 is also an existing materialized XML view. By observation, we know that Example 2 can be answered using Example 1 and Example 3. However, existing approaches for XML query answering can make use of one materialized view at a time as fragments cannot be shared between materialized views.

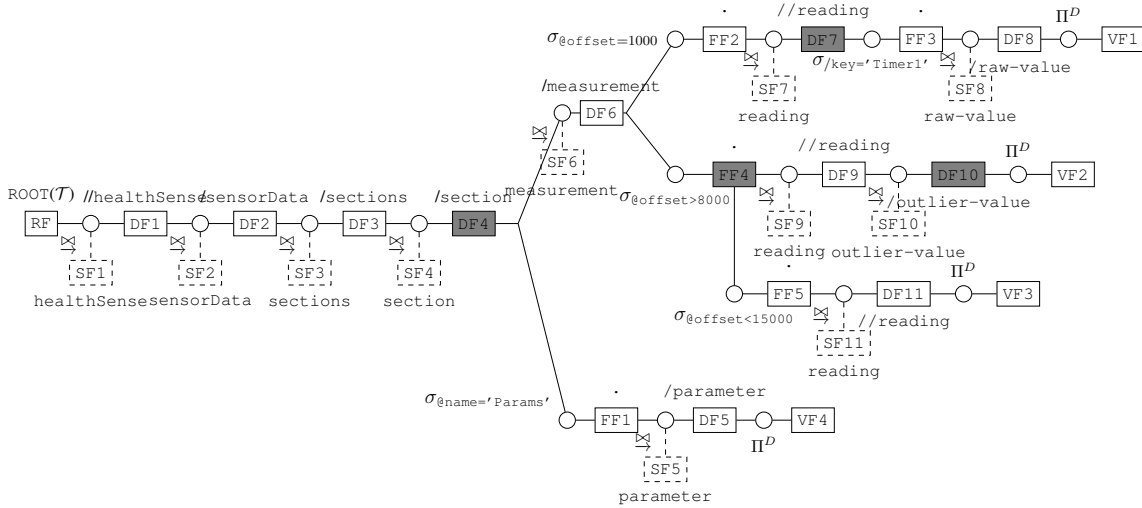


Figure 2: Multi-Fragment View Graph

## 1.4 Contribution

In this paper, we present a multi-fragment based XPath view materialization approach for an XML database that can utilize a set of materialized fragments as opposed to a collection of single materialized views. This approach can be easily applied by view adaptation algorithms.

Our key contributions can be outlined as follows:

- We provide Multiple Fragments Materialization View Graph (*MFV* view graph) that exploits the materialization of common XPath subexpressions to reduce the cost of view adaptation in terms of multi-fragment approach. While this approach has been previously employed in the set-based relational model, it has never been applied to tree-based systems.
- We present a specialized set of *operators* and *fragments* for constructing the *MFV* view graph.
- Our experimental evaluation demonstrates that significant performance improvement over the more traditional single-fragment approach can be achieved.

## 2 Related Research

Early efforts on view adaptation focused on keeping the materialized view up-to-date in response to query changes (Gupta et al. 1995). In this work, the authors demonstrate how the view adaptation problem differs from the problem of *query rewriting* by showing the new view to not always being equivalent to existing views. The view adaptation problem will only apply a sequence of *local changes*, such as add an attribute or delete a base relation. They also claim that this problem is closely related to the systems that are queried more frequently than updated. They show how these views can be adapted using an existing materialization for the cases where it is possible to do so. They identify extra information that can be kept with a materialization to facilitate redefinition.

In (Bellahsene 2004), the authors presented a fragment based view adaptation approach for the relational data model. They exploit materialization of common subexpressions to reduce the cost of view adaptation in the fragment-based approach. Nevertheless, the data independence is preserved for the views that are not affected by the change. In doing so, they provide the ability to reuse all the materialized fragments in the system to reduce the number of access commands on source data. However, to apply this approach to a semi-structured data model, a query transformation approach is required to convert the

XML query languages into the SQL language. Due to the difference between XML query languages and SQL, this requires a new set of operations to manage fragments and materialization.

In the area of semi-structured data, view updates with changes to source data has been studied by a number of research teams, e.g., (Sawires et al. 2005, Lim et al. 2003). However, the view adaptation problem in XML has attracted little focus. Recently, a related approach, *access control view*, has gained an increasing interest in supporting fine-grained XML access control (Damiani et al. 2000). Towards this end, techniques such as XPath security views (Fan et al. 2004, Kuper et al. 2005, Stoica & Farkas 2002, Ayyagari et al. 2007) are being used. However, (Ayyagari et al. 2007) are among the first to provide a solution for view adaptation in XML databases. They provide XPath access-control views with a set of comprehensive incremental view adaptation techniques. Materialized data is represented to the users according to a set of access control rules (XPath expressions). Based on these rules, data representation is restricted and dynamically changed for different users. However, this technique only operates when view adaptation starts from the output node of a query to its subtree. This is due to XPath semantics where only the XML fragment of the output node will be materialized.

We differ from these approaches as we adapted an idea from (Bellahsene 2004) utilizing multiple view fragment to efficiently maintain both common or uncommon fragments of different views in semi structured data. Due to sharing of materialized fragments, view adaptation can theoretically take place at any point in the view construct. Although our main focus is not to provide a solution to the access controls views, our approach can be easily modified to deal with the XPath security views mentioned above.

## 3 Introducing Sample Views

Existing research on XML views focused mainly on a subset of XPath expressions,  $XP\{/,//, [], *\}$ .  $/$  and  $//$  indicate the parent-child and ancestor-descendant relationships respectively, additionally, they are also the abbreviation of the child and descendant axes defined in XPath query language.  $[]$  is a predicate and  $*$  is the wildcard that represents all type of nodes. While we support all of these expressions, we also include attribute axis ( $@$ ), string value comparison and number comparison.

To assist the rest of this discussion, we provide the notation that will be used throughout the paper. An XML document can be modeled as a rooted, ordered and node-labeled tree,  $T$ . Let  $\Sigma$  denotes the alphabet of all distinct

Fragment	Name	Description
<i>RF</i>	Root Fragment	represents the starting point in a view model, a single node
<i>FF</i>	Filter Fragment	represents a node sequence after an <i>select</i> operation
<i>DF</i>	Dependency Join Fragment	represents a node sequence after an <i>d-join</i> operation
<i>SF</i>	Source Fragment	a sequence of <i>V</i> -typed within nodes an XML tree
<i>VF</i>	View Fragment	a sequence of nodes indicating the result of a view

Table 1: MFM Fragments

tag names in  $\mathcal{T}$ . We treat all nodes  $v_1, v_2, \dots, v_n$  with same label (*tag-name*) as type  $\mathcal{V}$ , where  $v_i \in \mathcal{T}$  ( $0 \leq i \leq n$ ),  $\mathcal{V} \in \Sigma$ .  $v_i$  is called a  $\mathcal{V}$ -typed node.

The MFM view graph is a combination of XPath views, where common subexpressions of XPath views are displayed once. XPath views are represented by a set of **fragments** (§4.1) and **operators** (§4.2) in the MFM view graph.

We now introduce a set of XPath views based on the schema given in Figure 1. In Example 4, raw values are returned where the `offset` is 1000 and the `key` attribute contains the value `Timer1`. In other words, all `Timer1` values with 1000 offset are contained in view *VF1*.

#### Example 4 (VF1)

```
//healthSense/sensorData/sections/section/measurement[./@offset=1000]//reading[./key='Timer1']/raw-value
```

Outliers are heart rate values that are most likely to be incorrectly determined by the sensors (often impossible heart rate values) and are generally removed before analysis can begin. In Example 5, view *VF2* returns all average time where offsets exceed 8000 milliseconds.

#### Example 5 (VF2)

```
//healthSense/sensorData/sections/section/measurement[./@offset>8000]//reading/average/time
```

In Example 6, *VF3* returns all reading data (outlier, padded, value and average) between specified times (determined by timing offsets).

#### Example 6 (VF3)

```
//healthSense/sensorData/sections/section/measurement[./@offset>8000][./@offset<15000]//reading
```

In Example 7, *VF4* contains all key and value attributes (of parameter) where the `name` attribute contains the value `Params`.

#### Example 7 (VF4)

```
//healthSense/sensorData/sections/section[./@name='Params']/parameter
```

The common parts between queries are listed once in the graph as shown in Figure 2, and the intermediate results are represented by the fragments. Fragments in gray represented materialized fragments and are selected manually for the purpose of illustration. The focus of this paper is to provide a fragment-based materialisation and automatic selection forms part of our current work.

## 4 View Fragments and Operators

In this section, we introduce the fragment set that provides the constructs for the Multiple Fragments Materialization (MFM) View Graph. A number of operators are then used to represent different XPath commands within the MFM graph.

### 4.1 MFM Fragments

Fragments are categorized into 5 types as illustrated in Table 1. Each fragment (except *Source Fragment*) represents the *result* of a single step in an XPath expression, and it is these fragments that can be shared across XML views.

All fragments contain a set of  $\mathcal{V}$ -typed instances for each node label  $\mathcal{V}$ . In the case of *RF* and *SF* fragments, this will be the entire set of instances for  $\mathcal{V}$ . For the remaining fragments, there will generally be some subset of  $\mathcal{V}$  generated for the fragment.

- **Root Fragment (*RF*)** - A Root Fragment represents a node sequence containing a single node, which is the root node of an XML tree  $\mathcal{T}$  (also known as the document node). It always represents the starting point of a *MFM* view graph. While a view graph will contain multiple query representations, they are all joined by the same root fragment, as shown in Figure 2 (for example, *RF* with rectangle box).
- **Filter Fragment (*FF*)** - A filter fragment (e.g., *FF1* in Figure 2) represents the node sequence produced by a *select* operation. In our view model, the select operation always contains a predicate used to filter an input node sequence. e.g., `@name='Params'` in Figure 2 represents the filter operation that results *FF1*.
- **Dependency Join Fragment (*DF*)** - A Dependency Join Fragment (e.g., *DF1* in Figure 2) represents the node sequence resulting from a *d-join* operation described in §4.2, e.g., the  $\bowtie$  before *DF1* represents a dependency join operation.
- **Source Fragment (*SF*)** - A Source Fragment represents the full set of  $\mathcal{V}$ -typed nodes. The major difference between this fragment and all others is that it cannot be reused and merely acts as an operand in a *d-join* operation. An example of a source fragment is shown in Figure 2 with dashed boxes.
- **View Fragment (*VF*)** - A view fragment (e.g., *VF1* in Figure 2) represents the result of a view. It always follows a deep project operation (§4.2), e.g.,  $\Pi^D$  before *VF1* indicates a deep project operation.

Each fragment within a particular view is referenced by a corresponding *VF* fragment that represents the context view. For example, *DF6* is referenced by *VF1*, *VF2* and *VF3* as it is shared by all whereas *DF8* is referenced only by *VF1*. The fragmentation approach is used to facilitate this sharing of fragments across views as each fragment indicates a potential end point (materialization candidate) for a view.

### 4.2 MFM Operators

Within our MFM view graph, fragments such as *DF*, *FF*, and *VF* are connected by one of a number of operators (see Table 2). We now present a description of these operators and explain why they are necessary.

Before we discuss the operations, we first introduce some notations that used for describing the operations. An XPath expression can be divided into several steps. Generally speaking, each step within an XPath expression contains an *Axis* and a *NameTest*, and it produces a sequence of nodes as an intermediate result to the next step in an XPath expression. A node sequence that results from a step  $s_i$  is denoted by  $S^k(\mathcal{V}_i)$ , where  $k$  indicates the number of nodes in the sequence,  $k \leq |S(\mathcal{V}_i)|$ .  $S(\mathcal{V}_i)$  is a sequence of all nodes labeled to  $\mathcal{V}_i$ . For instance,  $S(\text{reading})$  contains all nodes labeled **reading** within an XML document, whereas,  $S^3(\text{reading})$  indicates a sequence of **reading** nodes, where the size of the sequence is 3.

Operator	Name	Description	Operands	Operation Type
$\bowtie$	d-join	perform a dependency join operation between two sequences of nodes	SF and 1 of DF, FF, RF	binary
$\sigma_{pred}$	select	perform a select operation over a set of nodes with a specified condition	DF, FF	unary
$\Pi^D$	dproject	perform a deep project operation on a node sequence	DF, FF	unary

Table 2: Algebraic Operators

#### 4.2.1 Dependency Join

An XPath expression is represented by a chain of *dependency joins* (*d-joins*), represented by the algebraic operator  $\bowtie$ . The output is another sequence of nodes resulting from the axis operation and predicate filtering. In VF2, there are numerous dependency joins, for example, between `healthSense` and `sensorData`, and between `sensorData` and `section`.

##### Definition 1 (d-join operator)

A *d-join* operation is a binary operation written as  $\mathcal{S}(\mathcal{V}_{i-1}) \bowtie \mathcal{S}(\mathcal{V}_i)$ . The result of a *d-join* operation is a sequence of nodes  $\mathcal{S}^k(\mathcal{V}_i)$  which fulfills the *dependency condition* (*d-cond*) between  $s_{i-1}$  and  $s_i$ , where  $k \leq |\mathcal{S}(\mathcal{V}_i)|$ .

The result node sequence is generated using two steps: i) generates a set of 2-tuple sequences, where each satisfies the dependency condition; ii) projects only nodes ( $\mathcal{V}_i$ ) from tuples within the set. The semantics of a dependency join is listed below:

$$\begin{aligned} \mathcal{S}(\mathcal{V}_{i-1}) \bowtie \mathcal{S}(\mathcal{V}_i) &= \Pi_m(\{(n, m) \mid n \in \mathcal{S}(\mathcal{V}_{i-1}), m \in \mathcal{S}(\mathcal{V}_i), \text{REL}(n, m) \rightarrow d\text{-cond}\}) \\ &= \mathcal{S}^k(\mathcal{V}_i), \text{ where } k \leq |\mathcal{S}(\mathcal{V}_i)| \end{aligned}$$

where  $\mathcal{S}(\mathcal{V}_i)$  is dependent on  $\mathcal{S}(\mathcal{V}_{i-1})$  in terms of *d-cond* and, the method `REL` returns the relationship between two nodes  $n$  and  $m$ .

#### 4.2.2 Select

A step within an XPath expression may contain an optional set of predicates. Each predicate performs a filtering operation over the context node sequence together with a *selection operation* which selects nodes satisfying the predicate. In VF1, the select operation is `[/key='Timer1']`.

##### Definition 2 (select operator)

A *select* operation is an unary operation written as  $\sigma_{pred}(\mathcal{S}(\mathcal{V}_i))$  where *pred* is a condition of the selection. This operation selects a sequence of nodes in  $\mathcal{S}(\mathcal{V}_i)$  for which *pred* holds.

The result of a *select* operation is a subsequence of the input sequence containing the same typed nodes. The semantics of the operation are:

$$\begin{aligned} \sigma_{pred}(\mathcal{S}(\mathcal{V}_i)) &= \{n \mid n \in \mathcal{S}(\mathcal{V}_i), n \text{ satisfy } pred\} \\ &= \mathcal{S}^k(\mathcal{V}_i), \text{ where } k \leq |\mathcal{S}(\mathcal{V}_i)| \end{aligned}$$

#### 4.2.3 Deep Project

A Project operation returns a specified set of attributes. When an XPath expression generates its final set of result nodes, it will always return the entire subtree beneath each node. For this reason, we use a *deep project operation* (*dproject*) to project the *entire* subtree content of

each node in a node sequence. In VF2, the *dproject* operation returns the value for a single node as with a normal project operation. However, if `averages` was the requested node (see *Figure 1*), the Deep Project operation would return the entire `averages` subtree.

##### Definition 3 (dproject operator)

A *deep project* operation is a unary operation written as  $\Pi^D(\mathcal{S}(\mathcal{V}_i))$ . This operation projects the entire subtree content of nodes within  $\mathcal{S}(\mathcal{V}_i)$ .

The result of a *dproject* is a union of subtrees that are rooted at nodes of type  $\mathcal{V}_i$ . The semantic of a *dproject* operation is listed below. We use the method `SUB` to return the subtree of a node.

$$\begin{aligned} \Pi^D(\mathcal{S}(\mathcal{V}_i)) &= \Pi_{t_1, \dots, t_n}(\text{SUB}(v)), \text{ where } v \in \mathcal{S}(\mathcal{V}_i), t_i \in \text{SUB}(v) \\ &= \{c \mid \forall c \in \text{SUB}(v), v \in \mathcal{S}(\mathcal{V}_i)\} \\ &= \bigcup_{k=1}^n \text{SUB}(v_k), \text{ where } v_k \in \mathcal{S}(\mathcal{V}_i) \end{aligned}$$

### 5 Constructing the View Graph

In this section, we first describe a set of rules to be followed during MFM view graph construction. We then provide a detailed description of how the view graph is built using the fragments and operators introduced in §4.

#### 5.1 Construction Rules

As the MFM graph is constructed from XPath queries, the following is a set of rules used to derive the graph structure.

1. Any number of views may be defined on a database but they must all contain the same root (*RF*) fragment.
2. The *RF* fragment indicates the node sequence containing the document node (the root node of  $\mathcal{T}$ ) and must always be the first fragment in the view model. It can be followed only by a *d-join* operator.
3. A *deep project* operator can never be applied to a *RF* fragment as the result is the entire XML document.
4. A *select* operator can never be applied to a *RF* fragment. This is because applying a predicate to a document node always results in the entire XML document.
5. A *VF* fragment always occurs after an operation node representing the *deep project* operator.
6. A *VF* fragment is always the final fragment in the view model as it contains the final result set.
7. A *SF* fragment always occurs as the right operand of a dependency join operation.

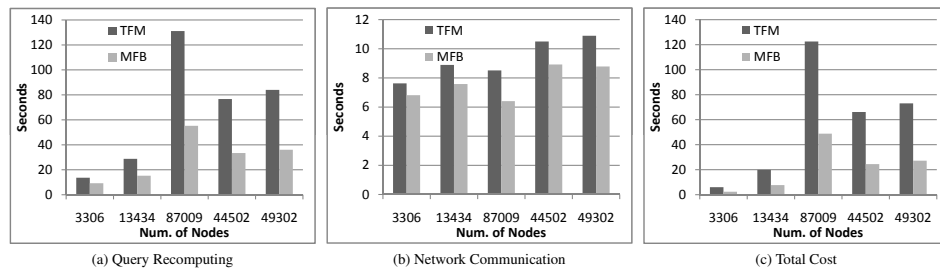


Figure 3: Add DF fragments

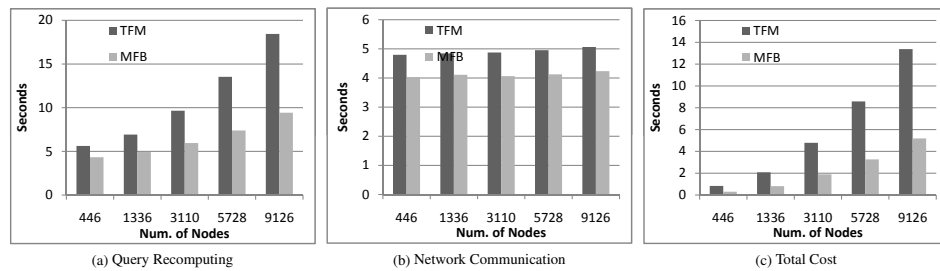


Figure 4: Add FF fragments

Assuming the four views introduced in §3 will comprise the view graph, the first step is to define a common root for views. We use the document root of the context XML document,  $ROOT(T)$ , to represent  $RF$  fragment. By observation, we can see that `healthSense` is the common part to all views in the first step. Therefore, as shown in *Figure 2* `healthSense` is joined with  $ROOT(T)$  (d-join). `healthSense` is represented by  $SF1$  containing all instances of `healthSense` nodes, and  $DF1$  indicates the result node set generated by the dependency join operation between  $RF$  and  $SF1$ . `sensorData` ( $SF2$ ) is then joined with the result generated by the first d-join operation ( $DF1$ ). This then generates  $DF2$ .

The d-join operation is repeated for all views until section (*SF4*) is encountered, which results in *DF4*. At this point, a select operation is performed for VF4 on name attribute (@name='Params'). A d-join operation is performed on VF1, VF2 and VF3, which joins section to measurement (*SF6*). As shown in *Figure 2*, this step produces a set of nodes represented by *DF6*.

The d-join and select operations are repeated until the output nodes are located for all views. DF5 represents a set of instances for the output node for VF4. A dproject operation can be performed on DF5, which retrieves all instance of output nodes *together with their subtree content*.

The transformation between XPath expressions and MFM view graph can be summarized by the following transformation rules.

## 5.2 D-join Transformation

*Transformation 1* shows the mapping from a  $d$ -join operation to our *MFM* view graph.

### Transformation 1 (djoin)

$$\mathcal{S}(\mathcal{V}_{i-1}) \boxtimes_{\rightarrow} \mathcal{S}(\mathcal{V}_i) = \mathcal{S}^k(\mathcal{V}_i) \iff \begin{array}{c} \mathcal{V}_{i-1} \quad \mathcal{V}_i \\ \boxed{F} \quad \text{---} \bigcirc \quad \boxed{DF} \\ \boxtimes_{\rightarrow} \quad \downarrow \\ \text{---} \boxed{SF} \text{---} \\ \mathcal{V}_i \end{array}$$

The transformation comprises the following steps:

1.  $\bowtie$  is transformed into the operation  $\text{d-join} \Rightarrow \bigcirc$
2.  $\mathcal{S}(\mathcal{V}_{i-1})$  is represented by a fragment,
 
$$\mathcal{S}(\mathcal{V}_{i-1}) \Rightarrow \boxed{\text{F}}^{\mathcal{V}_{i-1}}, F \in \{VF, DF, FF\}$$
3.  $\mathcal{S}(\mathcal{V}_i)$  is represented by a source fragment,
 
$$\mathcal{S}(\mathcal{V}_i) \Rightarrow \boxed{\text{SF}}^{\mathcal{V}_i}$$
4.  $\mathcal{S}^k(\mathcal{V}_i)$  is represented by a dependency join fragment,
 
$$\mathcal{S}^k(\mathcal{V}_i) \Rightarrow \boxed{\text{DF}}^{\mathcal{V}_i}$$

### 5.3 Select transformation

The mapping between a *select operation* and our *MFM* graph is shown in *Transformation 2*.

### Transformation 2 (Select)

$$\sigma_{pred}(\mathcal{S}(\mathcal{V}_i)) = \mathcal{S}^k(\mathcal{V}_i) \iff \boxed{F} \overset{\mathcal{V}_i}{\text{---}} \text{---} \text{---} \overset{\mathcal{V}_i}{\text{---}} \boxed{FF}$$

The transformation is achieved through the following steps:



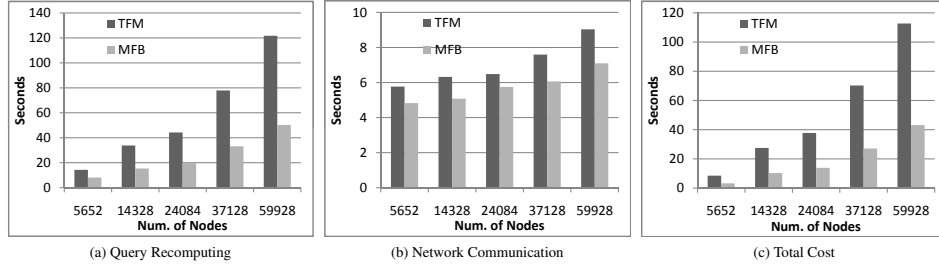


Figure 5: Delete DF fragments

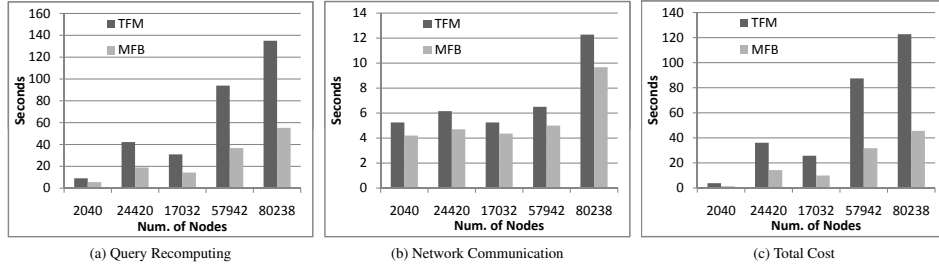


Figure 6: Delete FF fragments

1.  $\sigma_{pred}$  is transformed into the operation  $\sigma_{pred} \Rightarrow \bigcirc^{\sigma_{pred}}$
2.  $\mathcal{S}(\mathcal{V}_i)$  is represented by the fragment  $\mathcal{S}(\mathcal{V}_i) \Rightarrow \boxed{F}^{\mathcal{V}_i}, F \in \{DF, FF\}$
3.  $\mathcal{S}^k(\mathcal{V}_i)$  is represented by the fragment  $\mathcal{S}^k(\mathcal{V}_i) \Rightarrow \boxed{FF}^{\mathcal{V}_i}$

#### 5.4 dproject transformation

The mapping from the *dproject* operation to our *MFM* view graph is shown below.

##### Transformation 3 (dproject)

$$\Pi^D(\mathcal{S}(\mathcal{V}_i)) = \bigcup_{k=1}^n SUB(v_k)$$

$$\iff \boxed{F}^{\mathcal{V}_i} \text{---} \bigcirc^{\Pi^D} \text{---} \boxed{VF}^{\mathcal{V}_i}$$

The following steps are achieved to represent a *dproject* operation in our *MFM* view graph:

1.  $\Pi^D$  is encapsulated into an operation node,  $\Pi^D \Rightarrow \bigcirc^{\Pi^D}$
2.  $\mathcal{S}(\mathcal{V}_i)$  is encapsulated into an fragment,  $\mathcal{S}(\mathcal{V}_i) \Rightarrow \boxed{F}^{\mathcal{V}_i}, F \in \{DF, FF\}$
3.  $\bigcup_{k=1}^n SUB(v_k)$  is encapsulated into an fragment,  $\bigcup_{k=1}^n SUB(v_k) \Rightarrow \boxed{VF}^{\mathcal{V}_i}$

## 6 Experimental Analysis

Our fragment-based approach is well suited to the XPath view adaptation problem. We now demonstrate the performance of view adaptation using our *fragment-based* materialization approach and the traditional *full* materialization method. The experiment is initialized by materializing XPath views with data obtained from a remote XML server. For the traditional approach, views are required to be recomputed for materialization with new data obtained from this remote server. Our Multi-Fragment approach uses data from materialized fragments currently shared across queries. Clearly there are occasions when our multi-fragment approach may also need to obtain data from the remote server. This happens when query or view adaptation takes place in the steps *before* all materialized fragments. Concerning the query processing cost, the full materialisation approach clearly outperformed the fragment-based approach since some fragments are still virtual (not materialised). However, we show that when considering the global cost, including the query plus view maintenance costs, our fragment-based approach provides superior optimization. Our experiment demonstrates the general costs of adapting existing fragments in response to the view redefinition changes take place to the existing materialised XPath views.

### 6.1 Logistics

Two servers were deployed for this experiment: a remote (XML database) eXist server and a local eXist server. The remote server contains all XML source data, and the local server stores all materialized views and their fragments. We use eXist version 1.2.5 build 8668 for both remote and local eXist servers. The remote eXist server is distributed on an Intel Core(TM)2 Duo 2.66GHz workstation running Windows XP. The local eXist server is installed on an Intel Core(TM)2 Duo CPU 3.00GHz Windows XP workstation. The second server contains all XPath materialized views

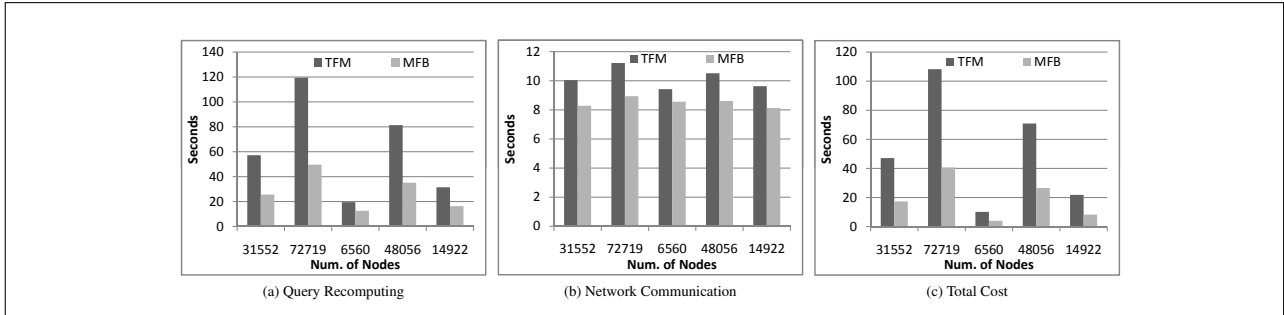


Figure 7: Modify FF fragments

with data obtained from the remote site. An actual dataset collected from heart monitors placed on a various groups of athletes, resulting in 230MB of data, was used as the underlying database.

## 6.2 Performance

The cost of view adaptation is computed as the total cost of query processing (query recomputing) and network communication between servers (data transformation between servers). In practical terms, we will always see an improvement as each example assumes that the change in view definition will *always* be accommodated in our view graph. This issue is discussed in our conclusions.

### 6.2.1 Adaptation: Reducing View Sizes

Adding an extra step to the XPath query, with or without predicates, normally requires deletion of some view data. Using the traditional approach, there is no way to determine which part of the materialized data that is now redundant. Therefore, this method requires a complete recomputation of the view and the extraction of data from the remote eXist server. With our multi-fragment approach, adding an extra step with optional set of predicates means that an additional *DF* fragment with optional *FF* fragments are inserted into the *MFM* graph. The multi-fragment approach can achieve rematerialization using the existing materialized fragments shared between different views.

Assume that *FF4* and *DF10* have been selected for materialization in Figure 2, where *FF4* is materialized by view *VF3* and *DF10* is materialized by view *VF2*. We would like to set a new condition after *FF4*, *name="Data"*, in *VF2*. In this case, *DF10* must be adapted to fulfill the new condition. *VF2* must be rematerialized using the traditional approach as the result are restricted by the new condition. However, our approach needs only to rematerialize *DF10*. Rather than retrieve more data from the server, this can be achieved by using the existing materialized fragment *FF4* in *VF3*.

Improvements in performance of our approach are illustrated in Figure 3a. This is in comparison with the full materialization approach (TFM) where entire views are always materialized. Figures 3a and 4a show that although query recomputing time is very close for both approaches, the network communication cost increases significantly for the traditional approach when the number of nodes in the sequence is increased (Figure 3b, 4b). As shown in Figure 3c and 4c, the multi-fragment approach requires approximately 50% of the time used in the traditional approach.

### 6.2.2 Adaptation: Increasing View Sizes

Deleting a step from an XPath query expression should result in a request for more data from the database server as the result is less restricted. The traditional approach must obtain the data from the remote eXist server. However,

this change does not effect the multi-fragment approach, as deleting a step from a view definition indicates that a *DF* fragment and *FF* fragments are deleted from the *MFM* graph.

Assume that *DF4* is materialised for view *VF4*, and *FF5* is materialized for view *VF3*. If we were to delete *FF4* from *VF3*, this indicates that the result of *VF3* becomes less restricted as the condition *@offset>8000* is removed. In the traditional approach, it is necessary to recompute the view but in our approach, we rematerialize only *FF5* by using *DF4* in *VF4*.

The result is that the multi-fragment approach retrieves additional data from *existing* fragments. As shown in Figure 5c and 6c, reusing existing data from different fragments is significantly faster than the traditional approach. In a worst case scenario, retrieving 59,928 nodes by the traditional approach takes approximately 3 times that of the multi-fragment approach.

### 6.2.3 Adaptation: Predicate changes

Changing view predicates could result in either more or less data required for the view. In either circumstance, the view must be rematerialized by the traditional approach. This is because only the node sequence produced by the last step is materialized. Therefore, any modification to the previous steps will always lead to view rematerialization. Using our approach, this requires a change to a single *FF* fragment. The multi-fragment approach can utilize existing materialized fragments to avoid gathering data from remote server as data has already been materialized within the fragments. As shown in Figure 7, the multi-fragment approach outperforms the traditional approach for all situations, and more so, for larger node quantities.

Assuming again that only *DF4* is materialised by view *VF2*, and *FF5* is materialised by view *VF3*. If we were to change the condition from *@offset<15000* to *@offset<10000* in *VF3*, the new predicate restricts the result of *VF3*. To update the materialized data for *VF3*, the traditional approach must access new data from the database server. Our approach updates the *FF5* by retrieving the data from *DF4* in *VF2*.

## 7 Conclusions

In this paper, we presented a multi-fragment materialization approach that works with semi-structured data. Fragments can be shared by multiple views, which improves the data reusability and reduces duplication involved in the traditional materialization approach. For the research described in this paper, we selected the view fragments for materialization by hand to show those query types that benefit from the multi-fragment approach. Our current focus is on developing a *cost-based view selection* algorithm and *view adaptation* algorithms to work with the multi-fragment platform created here. This will provide full automation of our view materialization system and with further analytical capabilities for scientists working with

the increasing volumes of data generated in today's sensor web.

## References

- Arion, A., Benzaken, V., Manolescu, I. & Papakonstantinou, Y. (2007), Structured Materialized Views for XML Queries, in 'VLDB '07: Proceedings of the 33rd International Conference on Very Large Data Bases', VLDB Endowment, pp. 87–98.
- Ayyagari, P., Mitra, P., Lee, D., Liu, P. & Lee, W.-C. (2007), Incremental adaptation of XPath access control views, in 'ASIACCS '07: Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security', ACM, New York, NY, USA, pp. 105–116.
- Balmin, A., Özcan, F., Beyer, K. S., Cochrane, R. & Pirahesh, H. (2004), A Framework for Using Materialized XPath Views in XML Query Processing, in 'VLDB', pp. 60–71.
- Bellahsene, Z. (2004), 'View Adaptation In The Fragment-Based Approach', *IEEE Trans. Knowl. Data Eng.* **16**(11), 1441–1455.
- 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 'SIGMOD Conference', ACM, pp. 479–490.
- Bruno, N., Koudas, N. & Srivastava, D. (2002), Holistic twig joins: optimal XML pattern matching, in 'SIGMOD Conference', ACM, pp. 310–321.
- Cautis, B., Deutsch, A. & Onose, N. (2008), XPath Rewriting Using Multiple Views: Achieving Completeness and Efficiency, in 'WebDB'.
- Damiani, E., de Capitani di Vimercati, S., Paraboschi, S. & Samarati, P. (2000), 'Design And Implementation Of An Access Control Processor For XML Documents', *Comput. Netw.* **33**(1-6), 59–75.
- Fan, W., Chan, C.-Y. & Garofalakis, M. (2004), Secure XML Querying With Security Views, in 'SIGMOD '04: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data', ACM, New York, NY, USA, pp. 587–598.
- Gao, J., Wang, T. & Yang, D. (2007), MQTree Based Query Rewriting over Multiple XML Views, in 'DEXA', pp. 562–571.
- Grust, T. (2002), Accelerating XPath Location Steps, in 'SIGMOD Conference', ACM, pp. 109–120.
- Gupta, A., Mumick, I. S. & Ross, K. A. (1995), Adapting Materialized Views after Redefinitions, in 'SIGMOD Conference', pp. 211–222.
- Kuper, G., Massacci, F. & Rassadko, N. (2005), Generalized XML Security Views, in 'SACMAT '05: Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies', ACM, New York, NY, USA, pp. 77–84.
- Lakshmanan, L. V. S., Wang, H. & Zhao, Z. (2006), Answering Tree Pattern Queries Using Views, in 'VLDB '06: Proceedings of the 32nd International Conference on Very Large Data Bases', VLDB Endowment, pp. 571–582.
- Lim, C.-H., Park, S. & Son, S. H. (2003), Access Control of XML Documents Considering Update Operations, in 'XMLSEC '03: Proceedings of the 2003 ACM Workshop on XML Security', ACM, New York, NY, USA, pp. 49–59.
- Marks, G. & Roantree, M. (2009), Metamodel-Based Optimisation of XPath Queries, in 'BNCOD', Vol. 5588 of *Lecture Notes in Computer Science*, Springer, pp. 146–157.
- Roantree, M., McCann, D. & Moyna, N. (2008), Integrating Sensor Streams in pHealth Networks, in '14th International Conference on Parallel and Distributed Systems (ICPADS 2008), December 8-10, 2008, Melbourne, Victoria, Australia', IEEE, pp. 320–327.
- Sawires, A., Tatemura, J., Po, O., Agrawal, D. & Candan, K. S. (2005), Incremental Maintenance of Path-Expression Views, in 'SIGMOD '05: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data', ACM, New York, NY, USA, pp. 443–454.
- Stoica, A. & Farkas, C. (2002), Secure XML Views, in 'DBSec', IFIP Conference Proceedings, Kluwer, pp. 133–146.
- Tang, N., Yu, J., Ozsu, M., Choi, B. & Wong, K.-F. (2008), Multiple Materialized View Selection for XPath Query Rewriting, in 'ICDE', IEEE, pp. 873–882.
- Tang, N., Yu, J. X., Tang, H., Özsu, M. T. & Boncz, P. A. (2009), Materialized View Selection in XML Databases, in 'DASFAA', pp. 616–630.



# A Real Time Hybrid Pattern Matching Scheme for Stock Time Series

Zhe Zhang<sup>1</sup>, Jian Jiang<sup>2</sup>, Xiaoyan Liu<sup>3</sup>, Ricky Lau<sup>4</sup>, Huaiqing Wang<sup>4</sup>, Rui Zhang<sup>3</sup>

<sup>1,4</sup>Department of Information Systems, City University of Hong Kong, Hong Kong

<sup>2</sup>Centre for Computational Finance and Economic Agents, University of Essex, UK

<sup>3</sup>Department of Computer Science and Software Engineering, University of Melbourne, Australia

<sup>1</sup>mailzhezhang@gmail.com, <sup>4</sup>{Ricky.lau, iswang}@cityu.edu.hk

<sup>2</sup>jjiangc@essex.ac.uk

<sup>3</sup>{xiaoyan1, rui}@csse.unimelb.edu.au

## Abstract

Pattern matching in stock time series is an active research area in data mining. We propose a new real-time hybrid pattern-matching algorithm in this paper. The algorithm is based on the Spearman's rank correlation, rule sets and sliding window. The concept of sliding windows enables patterns matching to be performed only based on subsequence of stock data which are freshly received. Therefore the proposed algorithm can be applied in real-time application and processing time can be reduced. Spearman's rank correlation coefficient is used to classify the preferred patterns effectively and efficiently first and use the rule sets to provide further ability for describing the query patterns so that is more effective, sensitive and constrainable in distinguishing individual patterns. Encouraging experiment is reported from the tests that the proposed scheme outperforms the other methods both effectively and efficiently, especially in differentiating the special preferred stock patterns or even distorted patterns.

**Keywords:** Spearman's rank correlation coefficient, Stock time series, Pattern matching.

## 1 Introduction

The similarity search in stock patterns has attracted the attention of the both business and technical experts in recent years. Technical analysts and traders believe that certain stock chart patterns and shapes are signals for profitable trading opportunities. So it is fundamental important to define an effective pattern matching scheme for stock time series.

Much work has been done on performing dimension reduction as the pre-processing step for similarity search to support efficient retrieval and matching of time series. Some of the commonly used methods include Discrete Fourier Transform (DFT) (Agrawal et al. 1993, Rafiei and Mendelzon 1997, Faloutsos et al. 1997), Discrete Wavelet Transform (DWT) (Popivanov and Miller 2002, Struzik and Siebes 1999), Piecewise Aggregate Approximation (PAA) (Yi and Faloutsos 2000).

To measure the similarity between sequence data, several distance metrics are commonly used, such as Euclidean distance (Agrawal et al. 1993), Dynamic Time Warping (DTW) distance (Berndt and Clifford 1994), and

the slope distance (Toshniwal and Joshi 2005) between the slopes of the lines approximating the two sequences. We call the two sequences similar if the distance between them is less than a user-specified threshold. Time sequence usually is long, so the distance computation can be time consuming. A solution is to map time sequences into the frequency domain using the Fourier transform, and to use the first few coefficients to filter out non-similar data. Besides the time consuming problem, another problem with this approach is that the user has no control over the meaning of similarity other than providing a threshold. However, there are some special patterns for stock time series as shown in Figure 1. And it is necessary to recognize and differentiate the patterns considering the amplitude of individual patterns.

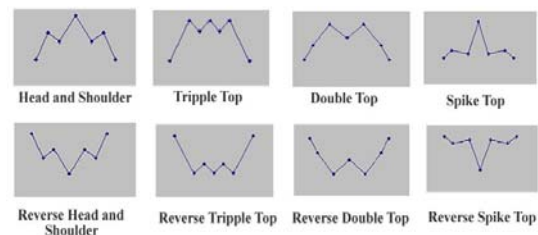


Figure 1: Eight basic patterns for stock data

Following our previous work on discovering the relationship between stock orders and stock price (Liu et al. 2010), in this paper, we explore the patterns in stock price. A flexible real time hybrid pattern-matching scheme is proposed. The sliding window based principle is involved in pattern matching to reduce the processing time for its online use. And we use Spearman's Rank Correlation Coefficient to classify the preferred patterns effectively and efficiently first, and use the rule sets to provide further ability for describing the query patterns so that is more effective, sensitive and constrainable in distinguishing individual patterns. It outperforms the other methods both effectively and efficiently, especially in differentiating the special preferred stock patterns or even distorted patterns. We will demonstrate the efficiency and effectiveness of the method via extensive experiments based on subsequence matching queries against the real stock price dataset as well as the synthetic dataset.

The remainder of the paper is organized as follows. We introduce the related work in Section 2. In Section 3, we present our proposed method in detail as well as how to use existing techniques. In Section 4, we report experimental results to compare the proposed method and the competitors. Finally, it goes to the conclusion in Section 5.

## 2 Related Work

There has been much work on similarity-based time series pattern matching. For example, Agrawal et al. (1993) propose an efficient index structure to retrieve time sequences similar to a given one. They map time sequences into the frequency domain by discrete Fourier transform and keep the first few coefficients in the index. Two sequences are considered similar if their Euclidean distance is less than a user-defined threshold.

Besides the DFT, the Discrete Wavelet Transform (DWT) has also been proposed to reduce the number of dimensions of feature vectors in time series (Popivanov and Miller 2002, Shahabi et al. 2000, Wu et al. 2000). Chan and Fu (1999) use the Haar Wavelet Transform for time series indexing. While both DFT and DWT focus on mapping the time sequences into other domain, some research focus on processing the time sequences directly in time domain. Many researchers use Piecewise approximation methods as basic to represent the feature extraction (Man and Wong 2001, Morinaka et al. 2001, Liu et al. 2008). For example, Shatkay and Zdonik (1996) propose a new notion of generalized approximate queries and proposed a framework that supports them.

Faloutsos et al. (1997) use a fixed gradient alphabet of three letters to represent the patterns, where each letter describes a movement of direction and covers a specific length of time. And Toshiniwal and Joshi (2005) consider the time series as a pattern defined by a set of regular expressions of slopes. This technique is based on the intuition that similar time sequences will have similar variations in their slopes and consequently in their time weighted slopes.

For the similarity pattern search method, the Euclidean distance is the mostly widely utilized metric to measure the similarity between the query and candidate sequence (Toshiniwal and Joshi 2005, Keogh and Pazzani 2000, Agrawal et al. 1993, Goldin and Kanellakis 1995, Rafiei and Mendelzon 1997). According to this method, if the Euclidean distance  $D(X, Y)$  between two time sequences  $X$  and  $Y$  of length  $n$  is less than a threshold  $\alpha$ , then the two sequences are said to be similar. The Euclidean distance is given as:

$$D(X, Y) = \sqrt{\sum_{i=1}^n (X_i - Y_i)^2}. \quad (1)$$

A major shortcoming in the Euclidean distance is that it only considers the whole shifts of two sequences and is not able to handle vertical and horizontal shifts separately, which exist between the time sequences under comparison.

Toshniwal and Joshi (2005) have used the cumulative variation of slopes for computing the similarity in the given time series data, which take into account of the ratio of vertical and horizontal shift. In this technique, the  $X_i, Y_i$  were substituted by the slopes  $Sc_j$  and  $Sq_j$ , which were for the  $j$ th strip in the candidate time sequence  $C$  and the query time sequence  $Q$  respectively. And Toshniwal and Joshi (2005) also presented an approach for similarity search in time series data, which is an improvement over the former one, and add time-weighted coefficient on the slope.

However, there are some special requirements for the stock data pattern matching due to their specialized shape

for frequently occurred patterns as shown in Figure 1. Therefore, it is required to consider the separate shift for each point in the patterns found in stock data for differentiating the specific patterns of stock when doing pattern similarity comparison. On the other hand, most of the patterns are distorted from the basic patterns (Fu et al. 2007), as shown in Figure 2, but they still should be retrieved during pattern searching. Therefore, we propose the new pattern-matching scheme, which can recognize and differentiate the special preferred stock patterns or even distorted patterns and make a faster and more accurate matching. In the following section, we will propose our hybrid pattern matching approach.

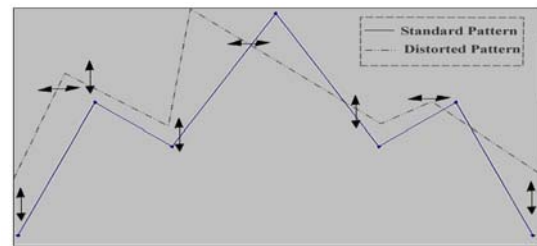


Figure 2: Distorted pattern

## 3 Stock Pattern Matching Based on Hybrid Pattern Matching Approach

Indeed, there are two main problems in pattern matching: how to define the preferred patterns for query and how to match the patterns with the pattern template in different resolution (Fu et al. 2007). We propose a flexible online pattern-matching scheme based on sliding window, which is involved in the whole matching process, including both in feature point extraction and pattern matching. We modify the feature point extraction method based on Perceptually Important Point (PIP) into sliding window based and being determined by the feed back of pattern matching outcome, fulfilling the demand of time saving and online use. On the first step, we propose a pattern-matching scheme, which based on Spearman's Rank Correlation Coefficient to classify the preferred patterns effectively and efficiently. And secondly we use the rule sets to provide further ability for describing the query patterns and is more effective, sensitive and constrainable in distinguishing individual patterns.

### 3.1 Pattern Matching Based on Sliding Window

A good representation or approximation of time series is important for time and memory efficiency of the searching algorithms (Man and Wong 2001). Time series pattern matching based on Perceptually Important Point (PIP) identification is introduced by Chung et al. (2002), which is used for feature point extraction. The principle of PIP algorithm is to capture the fluctuation of the sequence and take these highly fluctuated points as PIPs. Firstly, the first two PIPs are defined as the first and last point of input sequence  $P$ . The next PIP will be the point in  $P$  with maximum distance  $D$  to the first two PIPs. The fourth PIP will be the point in  $P$  with maximum  $D$  to its two adjacent PIPs. The process continues until the length of extracted series (SP) is equal to that of query sequence  $Q$ .

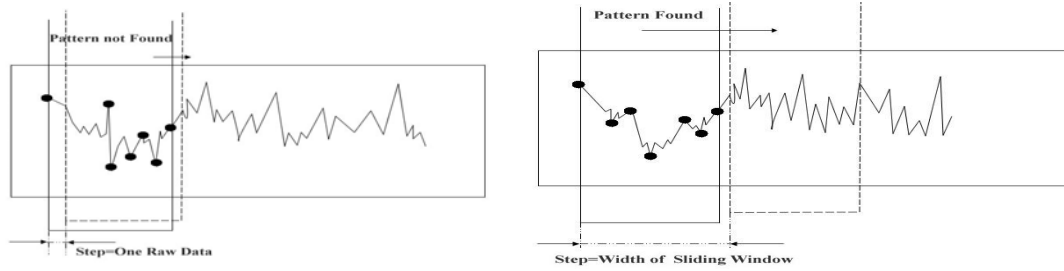


Figure 3: Process for sliding window movement

The PIP based algorithm is appropriate to be used for stock data (Zhang et al. 2006; Jiang et al. 2007). And Jiang et al. (2007) modified the maximum distance  $D$  into another coefficient to make it appropriate to stock characteristic. Fu et al. (2007) also modified the distance  $D$  into different forms and found that it was efficient and effective to extract the feature points when  $D$  was perpendicular distance between the test point and the line connecting the two adjacent PIPs. Therefore, PD was used in PIP calculation in the proposed algorithm.

We define a window width  $W$  for sliding window and put the PIP-PD based algorithm processed in each sliding window. The window moves step by step for searching the patterns. The step length of sliding window is determined by the pattern matching outcomes. There are two situations: (1) If the preferred pattern is found, the window will move the length, which is equal to the window; (2) if no preferred pattern is found, the window will move just one raw data to do pattern matching until the preferred pattern is found and it goes to situation (1). Figure 3 shows the process for sliding window movement. And Figure 4 shows the Sub procedure 1 for sliding window PIP-PD.

In such way, we can accelerate the pattern matching and make sure most of the patterns will not be skipped. And the sliding window scheme makes the algorithm available to online use while the old ones cannot. The main pseudo code for the pattern matching is shown in Figure 5. The detailed processes of Pattern Matching Procedure to sliding window series (Sub procedure 2) are shown in the next section.

**Sub Procedure 1:** Apply sliding windows on  $P[1:n]$  to extract sliding window-length feature point, within sliding window,  $SW[1:W]$

**Input:** Raw\_Dada( $P[1:n]$ ) for each window

**Output:**  $SP[1:N]$

**PIP procedure**

Set  $SP[1]=SW[1]$

Set  $SP[N]=SW[W]$

**Do**

    Select  $SW[i]$  with maximum PD to the adjacent points in  $SW$

$SP[j]=SW[i]$

**Until**  $j=N$

**End PIP**

Figure 4: Sub procedure 1 for sliding window PIP-PD

### 3.2 Pattern Matching Based on Spearman's Rank Correlation Coefficient

In this section, we will describe the sub procedure 2 in detail. Technical analysts and traders believe that certain stock chart patterns and shapes are signals for profitable trading opportunities. Many professional and amateur traders claim that they consistently make trading profits by following those signals. We propose eight basic patterns (Figure 1) which are proved to be useful in stock trading and defined by Lo (2000), in ranking format based on Spearman's rank method.

#### Main Procedure: Sliding Window Pattern Matching

**Input:** full time series data,  $RAW\_DATA (P[1:n])$

**Initialize:** Set Window Width,  $W$

Set the number of feature points extracted from time series within sliding window,  $N$

Set Pattern Template  $PAT\_TEMP [1:8]$

**Output:** Confirmed Pattern

#### DO

**Sub procedure 1:** Apply sliding windows on  $P[1:n]$  to extract sliding window-length feature point, within sliding window,  $SW[1:W]$

**End of Sub procedure 1**

**Sub procedure 2:** Apply Pattern Matching Procedure to  $SP[1:N]$  to get matched patterns from  $PAT\_TEMP[1:8]$ , the matched pattern is  $PAT$

**End of Sub procedure 2**

**If**  $PAT$  exists

    Move the window with step=  $W$

**Else**

    Move the window with step= next raw data -  $SW[1]$

**End if**

**Until** finish the length of  $P$

Figure 5: Main pseudo code for the pattern matching

The Spearman rank correlation coefficient is a nonparametric technique for evaluating the degree of correlation between two variables. It operates on the ranks of the data rather than the raw data. There are some advantages for using Spearman's rank correlation coefficient over the more common product moment correlation coefficient to define the stock pattern templates.

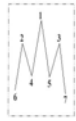
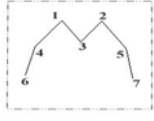

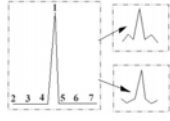

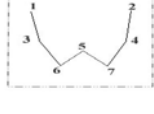

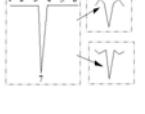
Patterns Name	Head & Shoulder	Double Top	Triple Top	Spike Top
Pattern Figure				
Position In the Descending Order	[6 2 4 1 5 3 7]	[6 4 1 3 2 5 7]	[6 1 4 2 5 3 7]	[2 3 4 1 5 6 7]
Rank	[6.5 2.5 4.5 1 4.5 2.5 6.5]	[6.5 4.5 1.5 3 1.5 4.5 6.5]	[6.5 2 4.5 2 4.5 2 6.5]	[4.5 4.5 4.5 1 4.5 4.5 4.5]
Patterns Name	Head & Shoulder (Reversed)	Double Top (Reversed)	Triple Top (Reversed)	Spike Top (Reversed)
Pattern Figure				
Position In the Descending Order	[1 5 3 7 4 6 2]	[1 3 6 5 7 4 2]	[1 5 3 6 4 7 2]	[1 2 3 7 4 5 6]
Rank	[1.5 5.5 3.5 7 3.5 5.5 1.5]	[1.5 3.5 6.5 5 6.5 3.5 1.5]	[1.5 6 3.5 6 3.5 6 1.5]	[3.5 3.5 3.5 7 3.5 3.5 3.5]

Table 1: Eight model patterns and ranks

There are several advantages for using Spearman's rank to define the stock pattern templates.

- (1) Only one sorting process is needed to treat the seven different patterns. It accelerates the speed for pattern patching while other methods need to apply different approaches to different patterns.
- (2) It can recognize more distorted patterns because it only considers the rank instead of the real data for each point. So it tolerance a shift range for each point when it ranks in the same position. For example, for the Head & Shoulder pattern, there is no restrict for whether the left shoulder should be higher than the right shoulder or not. So this method can achieve the requirement because the assigned rank number is equal for the two shoulders in our rank scheme (e.g. both are 2.5). For the Spike Top pattern, there are two situations just as shown in the pattern figure, so we define the pattern template in such way to make the pattern generalized.
- (3) It operates on the ranks of the data it is relatively insensitive to outliers and there is no requirement that the data be collected over regularly spaced intervals.

The raw scores are converted to ranks, and the differences  $\tilde{n}$  between the ranks of each observation on the two variables are calculated. We use the Eq. (2) to calculate the Spearman's Correlation Coefficient because there is no large number of tied ranks (Gauthier 2001). Each of the points with equal value should be assigned same rank. It is an average of their positions in the ascending order of the values. Then  $\tilde{n}$  is given by:

$$\rho = 1 - \frac{6 \times \sum_{i=1}^n d_i^2}{n(n^2 - 1)} \quad (2)$$

where  $d_i$  is the difference between ranks for each  $x_i, y_i$  data pair, and  $n$  is number of data pairs.

The steps for our pattern matching scheme in sub procedure 2 are: 1) Assign the rank to the feature points extracted from sub procedure 1, get SP\_R[1:N]. 2) Calculate SP\_R[1:N] with the template, PAT\_TEMP[1:8] to get the spearman coefficient, SP\_C[1:8]. 3) Comparing the SP\_C[1:8] with threshold of each template, THRE[1:8] and get the post pattern, POS\_PAT. 4) Evaluate whether the POS\_PAT can pass the rules defined for each pattern. 5) Put those POS\_PAT that pass the rules and with maximum Spearman's coefficient in MATCH\_PAT. The sub procedure 2 is shown the Figure 6.

#### Sub Procedure 2

**Input:** SP, PAT\_TEMP[1:8]

**Output:** MATCH\_PAT

Assign the Rank to SP[1:N], SP\_R[1:N]

Compare the SP\_R with the Ranks of each

PAT\_TEMP [1:8] to get Spearman Coefficient,

SP\_C[1:8]

**If** SP\_C [i] > Threshold of each template, THRE [i] (i = 1: 8)

    Include PAT\_TEMP[i] in the array of Possible patterns POS\_PAT

**End if**

#### Sub Procedure 2.1:

    Check whether POS\_PAT can pass the defined rules for each template.

#### End Sub Procedure 2.1

    Add POS\_PAT which can pass the RULES for each predefined templates of patterns AND maximum Spearman coefficient to Match\_Pat

**Return** MATCH\_PAT

**End of Sub Procedure 2**

Figure 6: Pseudo code of sub procedure 2



The disadvantage of this technology is that there is a loss of information when the data are converted to ranks. So we defined a set of rules for each pattern to provide further ability for describing the query patterns. The rule-based method can overcome the shortcoming of spearman's correlation coefficient method for it can recognize the specific patterns more explicitly. For example, in double top pattern, the two top's amplitudes should be within 15% difference according the definition of Lo et al. (2000).

The rule sets for sub procedure 2.1 is shown in Figure 7. And a set of user-defined parameters has to be set as follows.

- Max1: The maximum number of seven extracted points, SP[1:7] in sliding window
- Max2: The second maximum number of seven extracted points, SP[1:7] in sliding window
- Min1: The minimum number of seven extracted points, SP[1:7] in sliding window
- Min2: The second minimum number of seven extracted points, SP[1:7] in sliding window
- SP[i]: The ith position in seven points from the left to the right
- %: The percentage of difference between highest and lowest point

#### Time Scaling

Change the sequence length of the pattern templates from  $P[1...n]$  to  $P[1...m]$ , where  $n=7$  and  $m=25, 43$  and  $61$  (which means 4, 7 and 10 times length of original templates)

#### Time Warping

For each critical point  $p[i]$  in  $P$

Move  $p[i]$  between  $p[i-1]$  and  $p[i+1]$  randomly with width  $W$

$$W \in [t_{p[i]} - (t_{p[i]} - t_{p[i-1]})/3], t_{p[i]} + (t_{p[i+1]} - t_{p[i]})/3] \text{ or } \\ W \in [t_{p[i]} - (t_{p[i]} - t_{p[i-1]}) * 2/3, t_{p[i]} - (t_{p[i]} - t_{p[i-1]})/3] \\ \cup [t_{p[i]} + (t_{p[i+1]} - t_{p[i]})/3, t_{p[i]} + (t_{p[i+1]} - t_{p[i]}) * 2/3] \text{ or } \\ W \in [t_{p[i-1]}, t_{p[i]} - (t_{p[i]} - t_{p[i-1]}) * 2/3] \\ \cup [t_{p[i]} + (t_{p[i+1]} - t_{p[i]}) * 2/3, t_{p[i+1]}]$$

End for

#### Noise Adding

For each data point  $p[i]$  in  $P$

If randomly generated probability  
< prob (supposed to be 0.5)

diff =  $(p[i+1] - p[i]) * \text{random\_amplitude}$  between 0 to ampl

$$\text{ampl} \in [0, 0.1] \text{ or } \\ \text{ampl} \in [0.1, 0.2] \text{ or } \\ \text{ampl} \in [0.2, 0.3]$$

$$p[i] = p[i] \pm \text{diff}$$

End if

Figure 8: Pseudo code of generating synthetic pattern templates

## 4 Experiments

We evaluate our proposed method in this section. The proposed method is compared with two competitors: Euclidean distance based method and slope based method, which are popular and introduced in section 2. The accuracy is used to compare the three methods. It is defined as the percentage of the number of correctly matched patterns when a query pattern is given, and calculated as follows. We run experiments on both synthetic sequences and real stock price data, the past 21 years Hong Kong Hang Sang Index (5087 data points). And we use the definition proposed by Lo (2000) as the standard pattern:

$$\text{Accuracy} = \frac{\text{Number of correctly matched patterns}}{\text{Number of total patterns}} \quad (3)$$

### Rule sets for eight patterns in sub procedure 2.1

#### Rule Set for Double Top Pattern,

##### Orientation= Up (Down)

Rule 1: Max1 - Max 2 < 15%

(Rule 1' for Down: Min1 - Min 2 < 15%)

Rule 2: the two maximum (Rule 2': minimum) points are the 3rd and 5th point

(Rule 2' for Down: the two minimum points are the 3rd and 5th point)

Rule 3: Rank of SP2 > Rank of SP1 and rank of SP7 < Rank of SP6

(Rule 3' for Down: Rank of SP2 < Rank of SP1 and rank of SP7 > Rank of SP6)

#### Rule Set for Head & Shoulders Pattern,

##### Orientation= Up (Down)

Rule 1: |SP2 - SP6| < 15%

Rule 2: |SP3 - SP5| < 15%

Rule 3: the ranking of SP4 is first

(Rule 3' for Down: the ranking of SP4 is last)

Rule 4: the ranking of sp2 and sp6 must be 2 and 3

(Rule 4' for Down: the ranking of sp2 and sp6 must be 5 and 6)

Rule 5: the ranking of sp1 and sp7 must be 5 or 6 or 7

#### Rule Set for Triple\_tops Pattern, Orientation= Up (Down)

Rule 1: Max(|SP2 - SP4|, |SP2 - SP6|, |SP4 - SP6|) < 15% % used to separate from Head and shoulders

Rule 2: |SP3 - SP5| < 15%

Rule 3: sp2, sp4, sp6 must be 3 highest points

(Rule 3' for Down: sp2, sp4, sp6 must be 3 lowest points)

#### Rule Set for Spike\_top Pattern, Orientation= Up (Down)

Rule 1: time\_sequence of Max1 = 4

(Rule 1' for Down: time\_sequence of Min1 = 4)

Rule 2: SP4 - MAX2 >= 75%

% used to separate from Head & Shoulders

(Rule 2' for Down: SP4 - MIN2 >= 75%)

Figure 7: Rule-based sets for sub procedure 2.1

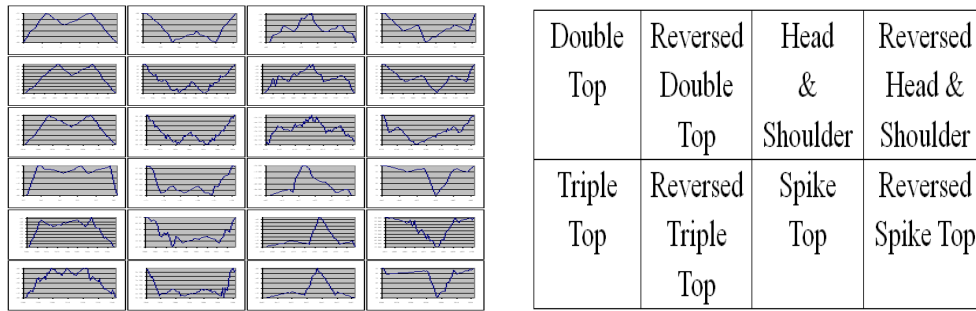


Figure 9: Sample of synthetic sequence for eight patterns

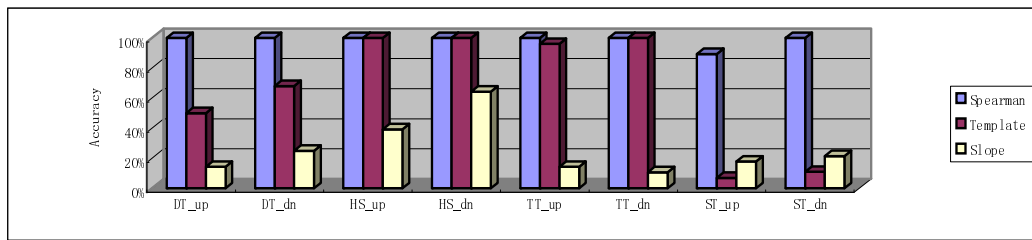


Figure 10: Accuracy for three methods

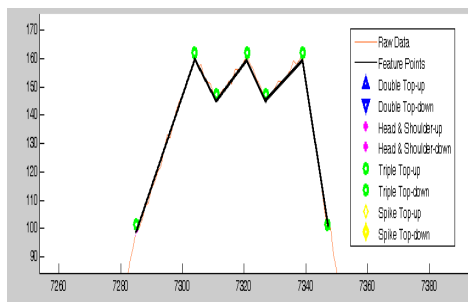


Figure 11: Triple pattern is found by our method

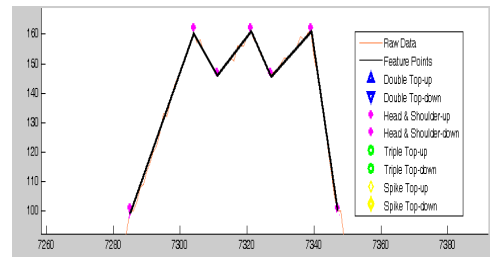


Figure 12: Slope method mistakes triple top pattern for head and shoulder pattern

#### 4.1 Synthetic Data

We generate 216 synthetic sequences for 8 patterns shown in Figure 1. There are three parameters for generating the sequences by improving more details of the parameters than Fu et al. (2007) proposed. The *time scaling* is used to scale the length of time series, and is defined for 3 levels: short scaling, middle scaling and long scaling. The *time warping* is used to change the position of critical points with three 3 levels: short width warping, middle width warping and large width warping. The noise is used to add three levels amplitudes of noise: small amplitude, middle amplitude, and large amplitude. The Pseudo code of generating the synthetic sequences is shown in Figure 8.

And we select some sample sequences for eight patterns with three parameters ranged between three levels in Figure 9.

We compare the accuracy of three methods used on the synthetic sequences. Figure 10 shows the accuracy for three different methods applied on eight patterns. Our approach outperforms the other approaches for all of the eight patterns. Because our approach pays more attention on the specific shape of each pattern rather than only

comparing the point-distance in the space just like the Euclidian distance method. And for the slope-based method, whose computation is derived from the Euclidian distance, considers the ratio of vertical and horizontal distance for each strip and pays less attention on the length of each strip. However, the specific rules are added in our approach for more detailed shape recognizing for each pattern, so it outperforms the slope-based method.

On the other hand, we can find from Figure 10 that our approach outperforms others especially for Spike Top and Triple Top pattern. Because these two patterns are similar in the shape and the only difference among them is the amplitude of each critical point.

For the Euclidian and Slope-based methods, they are easy to mistake the Triple Top pattern for Head and Shoulder pattern or for other. For example, the pattern shown in Figure 11 is Triple Top. However, the Slope Method mistakes it for Head and Shoulder in Figure 12. So their accuracies for Triple Top are lower than our method.

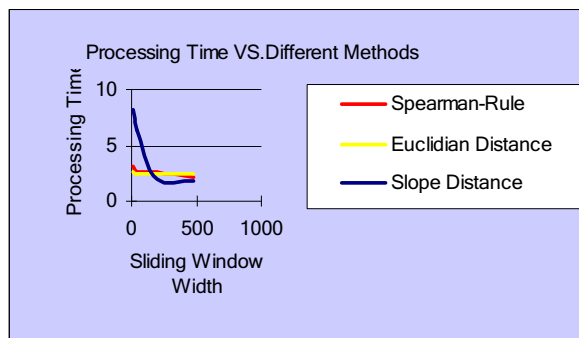
#### 4.2 Real Data

We apply our approach together with the two competitors on the past 21 years Hong Kong Hang Sang Index (5087

data points).

Firstly, we compare the processing time for three methods with different width of sliding window from small length, middle length and large length. The result is shown in Figure 13. We can find from Figure 13 that the processing time for our method is steady regardless the width of sliding window and approximately the same as the Euclidean Distance method. The processing time is less than the slope method and slightly more than the Euclidean Distance method when the length of sliding window is small. And it outperforms the Euclidean Distance method when the width of sliding window is large, and slightly takes more time than Slope method does. Because the advantage of our approach is that it pre-classifies the patterns due to the Spearman's Rank Correlation Coefficient and then specify the more detailed pattern based on rule sets. So if no pattern matches the Spearman's Coefficient in the first stage, the second stage for verify the detailed pattern is omitted and then the time is saved. So it will outperform the Euclidean Distance method when less patterns found in the roughly pattern shape determination (e.g. when using long sliding window length). When there are more patterns pass the first stage, it should only pay the additional time for the second stage so it needs slightly more time than the Euclidean Distance method. E.g. there are more patterns found when the small sliding window length is used, so our approach costs slightly more time than the Euclidean Distance method.

Another advantage of saving time for our approach is that in the second stage of our method, it reuses the parameters that results from the sorting process in the first stage such as the MAX, MIN. So there is no need to calculate other parameters again in order to save time.

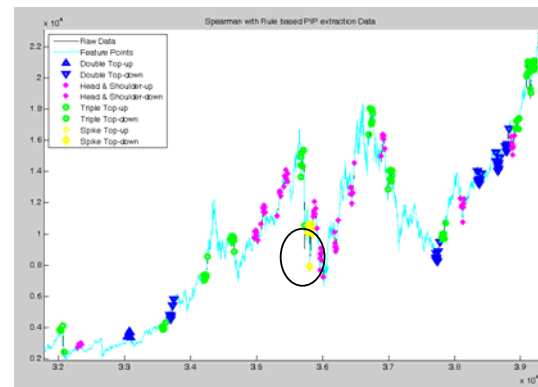


**Figure 13: Processing time for three patterns matching methods**

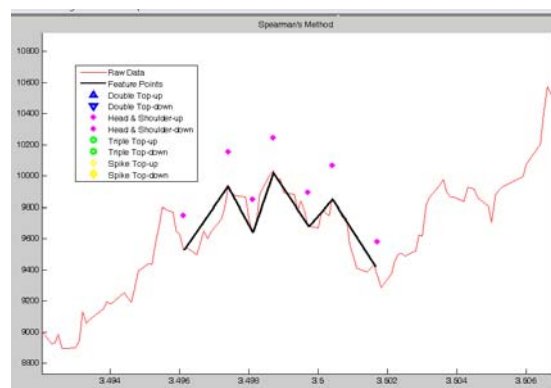
Secondly, we compare the performances of different methods for accuracy of identifying eight patterns by changing the length of sliding window. We select the sample when length of sliding window is 40, which is shown in Figure 14. And the zoom-in picture for circled area is shown in Figure 15. We can find from Figure 14 and Figure 15 that our method can recognize the eight patterns effectively.

The whole accuracy for eight patterns using three methods is shown in Figure 16. The result is similar as what is found in synthetic data. And the accuracy of spike top patterns was higher than the synthetic data because there were very little spike top patterns found in real data.

Our method outperforms the other two for each pattern. The accuracy for Head & Shoulder pattern, Triple Top Pattern by using Euclidean Distance method and Slope method are relatively low. Because the Euclidean Distance method only pays attention on the accumulated point shifts and neglects the individual shift of each point, it can not differentiate three similar patterns such as Head & Shoulder, Spike Top Pattern and Triple Top pattern when their only difference is the amplitude fluctuates between the middle point and the two side points. Neither does Slope method. Because although Slope method is a very efficient and effective method and does very well for the general time series for in time series pattern matching, it may not specialized and suitable for the stock pattern matching for stock patterns' characteristic. Therefore, it still can not differentiate the similar patterns as used in stock patterns. However, our method can fulfill the above requirement, by paying attention on the rank of each point and using rules to specify the patterns. Figure 17 shows the Head and Shoulder Pattern is correctly recognized by using our method when the Euclidean Distance method mistakes it for Triple Top pattern in Figure 18.



**Figure 14: Patterns found using our method (SW-Len=40)**

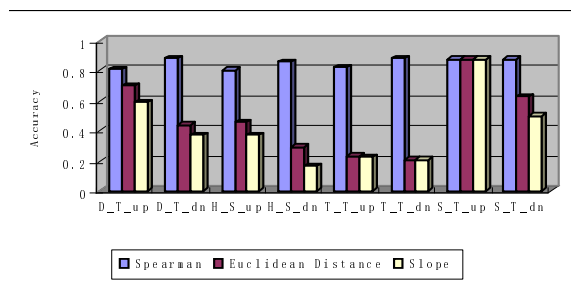


**Figure 15: Zoom-in area for recognized head and shoulder pattern**

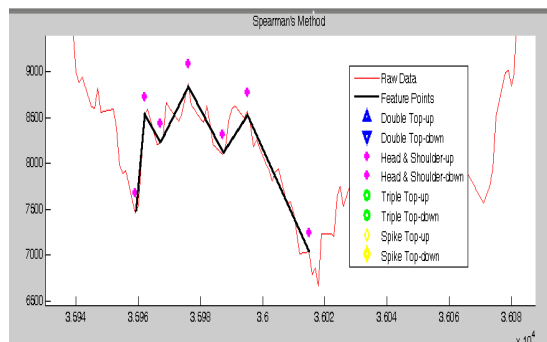
## 5 Conclusion

We propose a flexible online hybrid pattern-matching scheme, which is a combination of Spearman's Rank Correlation Coefficient and rule sets. We use Spearman's rank correlation coefficient to classify the preferred

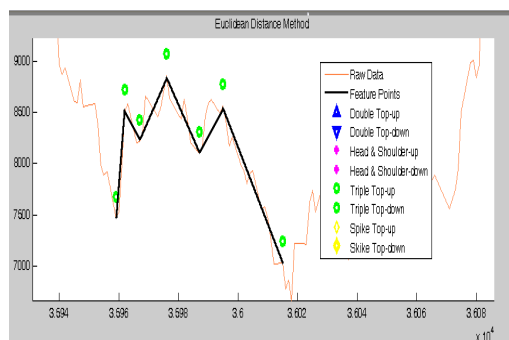
patterns effectively and efficiently first and use the rule sets to provide further ability for describing the query patterns so that is more effective, sensitive and constrainable in distinguishing individual patterns. It is much efficient and effective than the current pattern matching approaches such as Euclidian Distance based and Slope based method. The proposed scheme can be used real time for its sliding window based pattern matching and be with less processing time. It can also recognize more distorted patterns with noise. And it outperforms other methods when differentiating the special patterns for stock time series.



**Figure 16: Accuracy for finding eight patterns using three methods**



**Figure 17: Head and shoulder pattern recognized by using our method**



**Figure 18: Euclidean distance method mistakes head and shoulder pattern for triple top pattern**

## Acknowledgement

This work has been supported by the Australian Research Council (ARC) under project DP0880250.

## References

- Agrawal, R., Faloutsos, C. and Swami, A. (1993): Efficient similarity search in sequence databases. *Proc. of the 4th International Conference on Foundations of Data Organization and Algorithms*, Chicago, Illinois, USA, 69-84.
- Berndt, D. J. and Clifford, J. (1994): Using dynamic time warping to find patterns in time series. *Proc. of KDD-94: AAAI Workshop on Knowledge Discovery in Databases*, Seattle, Washington, USA, 359-370.
- Chan, K. and Fu, A. W. (1999): Efficient time series matching by wavelets. *Proc. of the 15th International Conference on Data Engineering*, Sydney, Australia, 126-133.
- Chung, F. L., Fu, T.C., Luk, R., and Ng, V. (2002): Evolutionary time series segmentation for stock data mining. *Proc. of 2<sup>nd</sup> IEEE International Conference on Data Mining*, Maebashi City, Japan, 83-90.
- Faloutsos, C., Jagadish, H., Mendelzon, A. and Milo, T. (1997): A signature technique for similarity based queries. *Proc. of the International Conference on Compression and Complexity of Sequences*, Positano-Salerno, Italy, 11-13.
- Fu, T. C., Chung, F. L., Luk, R., and Ng, C. (2007): Stock time series pattern matching: template-based vs. rule-based approaches, *Engineering Application of Artificial Intelligence*, 20(3): 347-364
- Gauthier, T. D. (2001): Detecting trends using spearman's rank correlation coefficient, *Environmental Forensics*, 2: 359-362.
- Goldin, D. Q. and Kanellakis, P. C. (1995): On similarity queries for time-series data: constraint specification and implementation. *Proc. of Constraint Programming 95*, Marseilles, France, 137-153.
- Jiang, J., Zhang, Z., and Wang, H. (2007): A new approach to improve multi-dimensional stock data Reduction, *Proc. of IADIS European Conference on Data Mining*, Lisbon, Portugal, 1-8.
- Keogh, E. and Pazzani, M. (2000): A simple dimensionality reduction technique for fast similarity search in large time series databases. *Proc. of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Kyoto, Japan, 122-133.
- Liu, X., Lin, Z. and Wang, H. (2008): Novel online methods for time series segmentation. *IEEE Transactions on Knowledge and Data Engineering*, 20(12): 1616-1626.
- Liu, X., Wu, X., Wang, H., Zhang, R., Bailey J. and Ramamohanarao K. (2010): Mining distribution change in stock order streams. To appear at *26th International Conference on Data Engineering Conference (ICDE) 2010*, Long Beach, California, USA.
- Lo, A.W., Mamaysky, H. and Wang, J. (2000): Foundations of technical analysis: computational

- algorithms, statistical inference, and empirical implementation. *Journal of Finance* 55 (4), 1705-1765.
- Man, P. W. P. and Wong, M. H. (2001): Efficient and robust feature extraction and pattern matching of time series by a lattice structure. *Proc. of the 10th International Conference on Information and Knowledge Management*, Atlanta, Georgia, USA, 271-278.
- Morinaka, Y., Yoshikawa, M., Amagasa, T. and Uemura, S. (2001): The l-index: an indexing structure for efficient subsequence matching in time sequence databases. *Proc. of Pacific-Asian Conference on Knowledge Discovery and Data Mining*, Hong Kong, China, 51-60.
- Popivanov, I. and Miller, R. J. (2002): Efficient similarity queries over time series data using wavelets. *Proc. of the 18th International Conference on Data Engineering*, San Jose, CA, USA, 273-282.
- Qu, Y., Wang, C., and Wang, X. S. (1998): Supporting fast search in time series for movement patterns in multiple scales. *Proc. of the Seventh International Conference on Information and Knowledge Management*, Bethesda, Maryland, USA, 251-258.
- Rafiei, D. and Mendelzon, A. (1997): Similarity-based queries for time series data. *Proc. of the ACM SIGMOD Conference*, Tucson, Arizona, USA, 13-25.
- Shahabi, C., Tian, X. and Zhao, W. (2000): Tsa-Tree: A wavelet-based approach to improve the efficiency of multi-level surprise and trend queries. *Proc. of 12th International Conference on Scientific and Statistical Database Management*, Berlin, Germany, 55-68.
- Shatkay, H. and Zdonik, S. (1996): Approximate queries and representations for large data sequences. *Proc. of the 12th International Conference on Data Engineering*, New Orleans, Louisiana, USA, 536-545.
- Struzik, Z. and Siebes, A. (1999): The Haar wavelet transform in the time series similarity paradigm. *Proc. Principles of Data Mining and Knowledge Discovery, 3rd European Conference*, Prague, Czech Republic, 12-22.
- Toshniwal, D. and Joshi, R. C. (2005): Similarity search in time series data using time weighted slopes, *Informatica* 29(1): 79-88.
- Wu, Y., Agrawal, D. and Abbadi, A. (2000): A comparison of DFT and DWT based similarity search in time-series databases. *Proc. of the 9th International Conference on Information and Knowledge Management*, McLean, VA, USA, 488-495.
- Yi, B. K. and Faloutsos, C. (2000): Fast time sequence indexing for arbitrary LP norms. *The VLDB Journal*, 385-394.
- Zhang, Z., Liu, X. Y. and Wang, H. Q. (2006): Discovery in stock time series based on perceptually important point algorithm. *Proc. of San Diego International Systems Conference*, San Diego, La, USA.



# Distributed Data Clustering in Multi-Dimensional Peer-To-Peer Networks

Stefano Lodi<sup>+</sup>Gianluca Moro<sup>\*</sup>Claudio Sartori<sup>+</sup>

Dept. of Electronics Computer Science and Systems  
University of Bologna,

<sup>\*</sup>Via Venezia, 52 - I-47023 Cesena (FC), Italy,

<sup>+</sup>Viale Risorgimento, 2, Bologna, Italy,

Email: {stefano.lodi, gianluca.moro, claudio.sartori}@unibo.it

## Abstract

Several algorithms have been recently developed for distributed data clustering, which are applied when data cannot be concentrated on a single machine, for instance because of privacy reasons or due to network bandwidth limitations, or because of the huge amount of distributed data. Deployed and research Peer-to-Peer systems have proven to be able to manage very large databases made up by thousands of personal computers resulting in a concrete solutions for the forthcoming new distributed database systems to be used in large grid computing networks and in clustering database management systems. Current distributed data clustering algorithms cannot be applied to such kind of networks because they expect data to be organized according to traditional distributed database management systems where the distribution of the relational schema is planned a-priori in the design phase. In this paper we describe methods to cluster distributed data across peer-to-peer networks without requiring any costly reorganization of data, which would be infeasible in such a large and dynamic overlay networks, and without reducing their performance in message routing and query processing.

We compare the data clustering quality and efficiency of three multi-dimensional peer-to-peer systems according to two well-known clustering techniques.

**Keywords:** Data Mining, Peer-to-Peer, Data Clustering, Multi-dimensional Data

## 1 Introduction

Distributed and automated recording, analysis and mining of data generated by high-volume information sources is becoming common practice in medium sized and large enterprises and organizations. Whereas distributed core database technology has been an active research area for decades, distributed data analysis and mining have been investigated only since the early nineties (Žaki & Ho 2000, Kargupta & Chan 2000) motivated by issues of scalability, bandwidth, privacy, and cooperation among competing data owners.

An important distributed data mining problem which has been investigated recently is the *distributed data clustering* problem. The goal of data clustering is to extract new potential useful knowledge from a

generally large data set by grouping together similar data items and by separating dissimilar ones according to some defined dissimilarity measure among the data items themselves. In a distributed environment, this goal must be achieved when data cannot be concentrated on a single machine, for instance because of privacy concerns or due to network bandwidth limitations, or because of the huge amount of distributed data. Several algorithms have been developed for distributed data clustering (Johnson & Kargupta 1999, Kargupta, Huang, Sivakumar & Johnson 2001, Klusch, Lodi & Moro 2003, da Silva, Klusch, Lodi & Moro 2006, Merugu & Ghosh 2003, Tasoulis & Vrahatis 2004). A common scheme underlying all approaches is to first locally extract suitable aggregates, then send the aggregates to a central site where they are processed and combined into a global approximate model. The kind of aggregates and combination algorithm depend on the data types and distributed environment under consideration, e.g. homogeneous or heterogeneous data, numeric or categorical data.

Among the various distributed computing paradigms, *peer-to-peer* (P2P) computing is currently the topic of one of the largest bodies of both theoretical and applied research. In P2P computing networks, all nodes (*peers*) cooperate with each other to perform a critical function in a decentralized manner, and all nodes are both users and providers of resources (Milojicic, Kalogeraki, Lukose, Nagaraja, Pruyne, Richard, Rollins & Xu 2002, Moro, Ouksel & Sartori 2002). In data management applications, deployed peer-to-peer systems have proven to be able to manage very large databases made up by thousands of personal computers. Many proposals in the literature have significantly improved the existing P2P systems in several aspects, such as searching performance, query expressivity, multi-dimensional distributed indexing. The ensuing solutions can be effectively employed in the forthcoming new distributed database systems to be used in large grid computing networks and in clustering database management systems.

In light of the foregoing, it is natural to foresee an evolution of P2P networks towards supporting distributed data mining services, by which many peers spontaneously negotiate and cooperatively perform a distributed data mining task. In particular, the data clustering task matches well the features of P2P networks, since clustering models exploit local information, and consequently clustering algorithms can be effective in handling topological changes and data updates. Current distributed data clustering algorithms cannot be directly applied to data stored in P2P networks because they expect data to be organized according to traditional distributed database management systems where the distribution of the relational schema is planned a-priori in the design phase.

Copyright ©2010, Australian Computer Society, Inc. This paper appeared at the Twenty-First Australasian Database Conference (ADC2010), Brisbane, Australia, January 2010. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 104, Heng Tao Shen and Athman Bouguettaya, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

In this paper we describe methods to cluster data distributed across peer-to-peer networks by using the same peer-to-peer systems with some revisions, namely without requiring any costly reorganization of data, which would be infeasible in such large and dynamic overlay networks, and without reducing their performance in message routing and query processing. Moreover, we compare the data clustering quality and efficiency of three multi-dimensional peer-to-peer systems with a well-known traditional clustering algorithm. The comparisons have been done by conducting extensive experiments on the peer-to-peer systems together with the clustering algorithm we have fully implemented.

## 2 Related Works

Extensions of P2P networks to data analysis and mining services have been dealt with by relatively few research contributions to date. In (Wolff & Schuster 2004) the problem of association rule mining is extended to databases which are partitioned among a very large number of computers that are dispersed over a wide area (large-scale distributed, or LSD, systems), including databases in P2P and grid systems. The core of the approach is the LSD-Majority protocol, an anytime distributed algorithm expressly designed for large-scale, dynamic distributed systems by which peers can decide if a given fraction of the peers has a data bit set or not. The Majority-Rule Algorithm for the discovery of association rules in P2P databases adopts a direct rule generation approach and incorporates LSD-Majority, generalized to frequency counts, in order to decide which association rules globally satisfy given support and confidence. The authors show that their approach exhibits good locality, fast convergence and low communication demands.

In (Klampanos & Jose 2004, Klampanos, Jose & van Rijsbergen 2006) the problem of P2P information retrieval is addressed by locally clustering documents residing at each peer and subsequently clustering the peers by a one-pass algorithm: Each new peer is assigned to the closest existing cluster, or initiates a new peer cluster, depending on a distance threshold. Notwithstanding the approach produces a clustering of the documents in the network, these works do not compare directly to ours, since their main goal is to show how simple forms of clustering can be exploited to reorganize the network to improve query answering effectiveness. The work (Agostini & Moro 2004) describes a method for inducing the emergence of communities of peers semantically related, which corresponds to the clustering of the P2P network by document contents. In this approach as queries are resolved, the routing strategy of each peer, initially based on syntactic matching of keywords, becomes more and more trust-based, namely, based on the semantics of contents, leading to resolve queries with a reduced number of hops.

Recently distributed data clustering approaches have also been developed for wireless sensor networks, such as in (Lodi, Monti, Moro & Sartori 2009), where the peculiarity, differently from large wired peer-to-peer systems, is to satisfy severe constraints according to the kind of resources, such as energy consumption, short range connectivity, computational and memory limits.

As of writing, there is only one study on P2P data clustering not in relation to automatic, content-based reorganization of the network for efficiency purposes. In (Li, Lee, Lee & Sivasubramaniam 2006) the PENS algorithm is proposed to cluster data stored in P2P networks with a CAN overlay, employing a density-

based criterion. Initially, each peer executes locally the DBSCAN algorithm. Then, for each peer, neighbouring CAN zones which contain clusters that can be merged to local clusters contained in the peer's zone are discovered, by performing a cluster expansion check. The check is performed bottom-up in the virtual tree implicitly defined by CAN's zone-splitting mechanism. Finally, arbiters appropriately selected in the tree merge the clusters. The authors show that the communication cost of their approach is linear in the number of peers. Like the methods we have considered in our analysis, the approach of this work assumes a density-based clustering model. However, clusters emerge by bounding the space embedding the data along contours of constant density, as in the DBSCAN algorithm, whereas the algorithms considered in the present paper utilize either a gradient-based criterion, similar to the one proposed in (Hinneburg & Keim 1998) to define center-based clusters, or a mean density criterion.

## 3 Multi-dimensional Peer-To-Peer Systems

In this section we review three different P2P networks which have been proposed in the literature: CAN, MURK-CAN, and MURK-SF. In Section 4, data clustering algorithms for each of these networks will be described and experimentally evaluated.

A CAN (Content-Addressable Network) overlay network (Ratnasamy, Francis, Handley, Karp & Schenker 2001) is a type of distributed hash table by which (key,value) pairs are mapped to a  $d$ -dimensional toroidal space by a deterministic hash function. The toroidal hash space is partitioned into "zones" which are assigned uniquely to nodes of the network. Every node keeps a routing table as a list of pointers to its immediate neighbours and of the boundaries of their zones. Using this information, query messages are routed from node to node by always choosing the neighbour which decreases distance to the query point most, until the node which owns the zone containing the query point is reached. A peer joining the network randomly selects an existing zone and sends (using routing) a split request to the owning node, which splits it into two sub-zones along one dimension (the dimension is chosen as the next dimension in a fixed ordering) and transfers to the new peer both the ownership of the sub-zone and the (key,value) pairs hashed to the sub-zone. A peer leaving the network hands over its zone and the associated (key,value) pairs to one of its neighbors. In both cases, the routing tables of the nodes owning the zones which are adjacent to the affected zone are updated.

A MURK (MULTi-dimensional Rectangulation with Kd-trees) network (Ganesan, Yang & Garcia-Molina 2004) manages a nested, rectangular partition in a similar way, but in contrast to CAN, the partition is defined in the data space directly, which is assumed to be a multi-dimensional vector space. Moreover, when a node arrives, the zone is split into two sub-zones, containing the same number of objects; that is, MURK balances load whereas CAN balances volume. Two different variants of MURK are introduced in (Ganesan et al. 2004), MURK-CAN and MURK-SF, which differ in the way nodes are linked by the routing tables. In MURK-CAN, neighbouring nodes are linked exactly as in CAN, whereas in MURK-SF, links are determined by a skip structure. A space-filling curve (the Hilbert curve) is used to map the partition centroids of the zones to one-dimensional space. The images of all centroids induce a linear ordering of the nodes which is used to build the skip graph.



#### 4 Density-Based Clustering

Data clustering is a descriptive data mining task which aims at decomposing or partitioning a usually multivariate data set into groups such that the data objects in one group are similar to each other and are different as possible from those in other groups. Therefore, a clustering algorithm  $A()$  is a mapping from any data set  $S$  of objects to a clustering of  $S$ , that is, a collection of pairwise disjoint subsets of  $S$ . Clustering techniques inherently hinge on the notion of distance between data objects to be grouped, and all we need to know is the set of interobject distances but not the values of any of the data object variables. Several techniques for data clustering are available but must be matched by the developer to the objectives of the considered clustering task [Grabmeier and Rudolph, 2002].

In partition-based clustering, for example, the task is to partition a given data set into multiple disjoint sets of data objects such that the objects within each set are as homogeneous as possible. Homogeneity here is captured by an appropriate cluster scoring function. Another option is based on the intuition that homogeneity is expected to be high in densely populated regions of the given data set. Consequently, searching for clusters may be reduced to searching for dense regions of the data space which are more likely to be populated by data objects.

We assume a set  $S = \{\vec{O}_i \mid i = 1, \dots, N\} \subseteq \mathbb{R}^d$  of data points or objects. *Kernel estimators* formalize the following idea: The higher the number of neighbouring data objects  $\vec{O}_i$  of some given  $\vec{O} \in \mathbb{R}^d$ , the higher the density at  $\vec{O}$ . The influence of  $\vec{O}_i$  may be quantified by using a so called kernel function. Precisely, a *kernel function*  $K(\vec{x})$  is a real-valued, non-negative function on  $\mathbb{R}^d$  having unit integral over  $\mathbb{R}^d$ . Kernel functions are often non-increasing with  $\|\vec{x}\|$ . When the kernel is given the vector difference between  $\vec{O}$  and  $\vec{O}_i$  as argument, the latter property ensures that any element  $\vec{O}_i$  in  $S$  exerts more influence on some  $\vec{O} \in \mathbb{R}^d$  than elements which are farther from  $\vec{O}$  than the element. Prominent examples of kernel functions are the standard multivariate normal density  $(2\pi)^{-d/2} \exp(-\frac{1}{2} \vec{x}^T \vec{x})$ , the uniform kernel  $K_u(\vec{O})$  and the multivariate Epanechnikov kernel  $K_e(\vec{O})$ , defined by

$$K_u(\vec{O}) = \begin{cases} c_d^{-1} & \text{if } \vec{x}^T \vec{x} < 1, \\ 0, & \text{otherwise,} \end{cases} \quad (1)$$

$$K_e(\vec{O}) = \begin{cases} \frac{1}{2} c_d^{-1} (d+2)(1 - \vec{x}^T \vec{x}) & \text{if } \vec{x}^T \vec{x} < 1, \\ 0, & \text{otherwise,} \end{cases} \quad (2)$$

where  $c_d$  is the volume of the unit  $d$ -dimensional sphere. A *kernel estimator* (KE)  $\hat{\varphi}[S](\vec{O}): \mathbb{R}^d \rightarrow \mathbb{R}_+$  is defined as the sum over all data objects  $\vec{O}_i$  of the differences between  $\vec{O}$  and  $\vec{O}_i$ , scaled by a factor  $h$ , called *window width*, and weighted by the kernel function  $K$ :

$$\hat{\varphi}[S](\vec{O}) = \frac{1}{Nh^d} \sum_{i=1}^N K\left(\frac{1}{h}(\vec{O} - \vec{O}_i)\right). \quad (3)$$

The estimate is therefore a sum of exactly one “bump” placed at each data object, dilated by  $h$ . The parameter  $h \in \mathbb{R}_+$  controls the smoothness of the estimate. Small values of  $h$  result in merging fewer bumps and a larger number of local maxima. Thus, the estimate reflects more accurately slight local variations in the density. Increasing  $h$  causes the distinctions between

regions having different local density to progressively blur and the number of local maxima to decrease, until the estimate is unimodal.

An objective criterium to choose  $h$  which has gained wide acceptance is to minimize the mean integrated square error (MISE), that is, the expected value of the integrated squared pointwise difference between the estimate and the true density  $\varphi$  of the data. An approximate minimizer is given by

$$h_{\text{opt}} = A(K) N^{-1/(d+4)}, \quad (4)$$

where  $A(K)$  depends also on the dimensionality of the data  $d$  and the unknown true density  $\varphi$ . In particular, for the unit multivariate normal density

$$A(K) = \left(\frac{4}{2d+1}\right)^{1/(d+4)}. \quad (5)$$

For a multivariate Gaussian density

$$h = h_{\text{opt}} \sqrt{d^{-1} \sum_{j=1}^d s_{jj}} \quad (6)$$

where  $s_{jj}$  is the data variance on the  $j$ -th dimension (Silverman 1986).

In some applications, including data clustering, it may be useful to locally adapt the degree of smoothing of the estimate. In clustering, for instance, a single dataset may both contain large, sparse clusters, and smaller, dense clusters, possibly not well separated. The estimate given by (3) is not suitable in such cases. In fact, a fixed global value of the window width would either merge the smaller clusters, or make emerge spurious details in the larger ones.

Adaptive density estimates have been proposed both as generalizations of kernel estimates and nearest neighbour estimates. In the following we will recall the latter family of estimators. The nearest neighbour estimator in  $d$  dimensions is defined as:

$$\hat{\psi}[S](\vec{O}) = \frac{k/N}{c_d r_k(\vec{O})^d} \quad (7)$$

where  $r_k(\vec{O})$  equalling  $k$ , the number of data objects in the smallest sphere including the  $k$ -th neighbour of  $\vec{O}$  to the expected number of such objects,  $N\hat{\psi}[S](\vec{O})c_d r_k(\vec{O})^d$ . Equation (7) can be viewed as a special case for  $K = K_u$  of a kernel estimator having  $r_k(\vec{O})$  as window width:

$$\hat{\varphi}[S](\vec{O}) = \frac{1}{N r_k(\vec{O})^d} \sum_{i=1}^N K\left(\frac{\vec{O} - \vec{O}_i}{r_k(\vec{O})}\right). \quad (8)$$

The latter estimate is called a *generalized nearest neighbour estimate* (GNNE).

A simple property of kernel density estimates that is of interest for P2P computing is locality. In order to obtain a meaningful estimate, the window width  $h$  is usually much smaller than the data range on every coordinate. Moreover, the value of commonly used kernel functions is negligible for distances larger than a few  $h$  units; it may even be zero if the kernel has bounded support, as is the case for the Epanechnikov kernel. Therefore, in practice the number of distances that are needed for calculating the kernel density estimate at a given object  $\vec{O}$  may be much smaller than the number of data objects  $N$ , and the involved objects span a small portion of the data space.

Once the kernel density estimate of a data set has been computed, there is a straightforward strategy

to cluster its objects: Detect disjoint regions of the data space where the value of the estimate is high and group all data objects of each region into one cluster. Data clustering is thus reduced to space partitioning, and the different ways “high” can be defined induce different clustering schemes.

In the approach of Koontz, Narendra and Fukunaga (Koontz, Narendra & Fukunaga 1976), as generalized in (Silverman 1986), each data object  $\vec{O}_i$  is connected by a directed edge to the data object  $\vec{O}_j$ , within a distance threshold, that maximizes the average steepness of the density estimate between  $\vec{O}_i$  and  $\vec{O}_j$ , and such that  $\hat{\varphi}[S](\vec{O}_i) > \hat{\varphi}[S](\vec{O}_j)$ . Clusters are defined by the connected components in the resulting graph. More recently, Hinneburg and Keim (Hinneburg & Keim 1998) have proposed two types of cluster. *Center-defined* clusters are based on the idea that every local maximum of  $\hat{\varphi}$  having a sufficiently large density corresponds to a cluster including all data objects which can be connected to the maximum by a continuous, uphill path in the graph of  $\hat{\varphi}$ . An *arbitrary-shape* cluster (Hinneburg & Keim 1998) is the union of center-defined clusters such that their maxima are connected by a continuous path whose density exceeds a threshold. A *density-based* cluster (Ester, Kriegel, Sander & Xu 1996) collects all data objects included in a region where the value of a kernel estimate with uniform kernel exceeds a threshold.

## 5 Density-Based Clustering in P2P Systems

When applying kernel-based clustering to P2P overlay networks, some observations are in order.

- It is mandatory to impose a bound on the distance  $H$  in hops of the zones containing the objects that contribute to the estimate in a given zone. A full calculation of summation (3) would require to answer an unacceptable number of point queries. Note that, depending on the overlay network, the lower bound on the distance from the center of a zone to an object in a zone beyond  $H$  hops may be not greater than the radius of the zone itself. Thus, although the contribution to the estimate at  $\vec{O}$  of objects located more than, say,  $4h$  is negligible, if  $4h$  is greater than zone radius, some terms of the estimate may be missed. There will be a trade-off between network messaging costs and clustering accuracy, and clustering results must be experimentally compared with the ideal clustering obtained when  $H$  is large enough to reach all objects.
- Different peers may prefer different parameters for clustering the network’s data, e.g., different values of  $h$ , kernel functions, maximum number of hops, whether to use an adaptive estimate. Therefore, a peer interested in clustering the data acts as a clustering *initiator*, i.e., it must take care of all the preliminary steps needed to make its choices available to the network, and to gather information useful to make those choices, e.g., descriptive statistics.

In this paper, we investigate two approaches to P2P density-based clustering. In both approaches, the computation of clusters is based on the generalized approach in (Silverman 1986) as described in Section 4. The first one,  $\mathcal{M}_1$ , uses kernel or generalized nearest neighbour estimates, and it can be summarized as follows.

- I. If the estimate (3) is used, then the initiator collects from every zone its object count, object

sum, and square sum to globally choose a window width  $h$  according to Equations (4)–(6).

- II. At every node: For every local data point  $\vec{O}$ , compute the density estimate value  $\hat{\varphi}[S](\vec{O})$ , in the form (3) or (8), from the local data set and the remote data points which are reachable by routing a query message for at most  $H$  hops, where  $H$  is an integer parameter
- III. At every node: Query the location and value of all local maxima of the estimate located within other zones
- IV. At every node: associate each local data point to the maximum which maximizes the ratio between the value of the maximum and its distance from the point

The second approach,  $\mathcal{M}_2$ , exploits data space partitions implicitly generated by the data management subdivision among peers as described in Section 3. In this approach, the data are not purposely reorganized or queried to compute a density estimate to perform a clustering. In this case, the density value at data objects in a zone can be set as the ratio between the number of objects in the zone and the volume of the zone.

- A. At every node: For every local data object  $\vec{O}$ , compute the density estimate value  $\hat{\varphi}[S](\vec{O})$  from the local data set only, as the mean zone density, that is, the object count in the node’s zone divided by its volume
- B. At every node: Define the maximum of the node’s zone as the mean density of the zone, and its location as the geometric center of the zone
- C. At every node: query the maximum of all zones and their locations; associate each local data point to the maximum which maximizes, over all zones, the ratio between the value of the maximum and its distance from the point

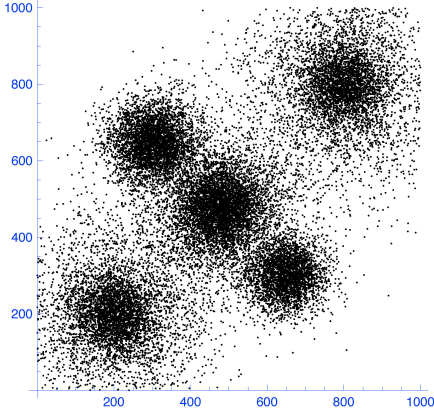
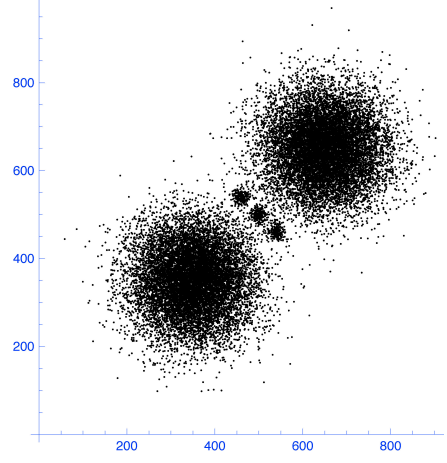
In this approach, no messages are sent over the network for computing densities, but only for computing the clusters. Therefore, it is expected to be much more efficient than the previous one, but less accurate, due to the approximation in computing the estimates maxima.

## 6 Data Clustering Efficacy and Efficiency

The main goal of the experiments described in the next section is to compare the accuracy of the clusters produced by three P2P systems, namely their efficacy, as a function of the network costs, that is their efficiency as clustering algorithms.

To determine the accuracy of clustering, we have compared the clusters generated by each P2P system as a function of the number of hops, with the ideal clustering computed by the system when routing through a large number of hops in order to include the entire network; for our experiments we have chosen 1024. In the latter case, all zones are reachable from every other zone, thus simulating a density-based algorithm operating as if all distributed data were centralized in a single machine, as far as query results are concerned. Limiting the number of hops means the computed estimate is an approximation of the true estimate computed by routing queries to the entire network, which therefore yields a “reference” clustering.

We have employed the *Rand index* (Rand 1971) as a measure of clustering accuracy. Let  $S =$

Figure 1: Dataset  $S_0$ Figure 2: Dataset  $S_1$ 

$\{\vec{O}_1, \dots, \vec{O}_N\}$  be a dataset of  $N$  objects and  $X$  and  $Y$  two data clusterings of  $S$  to be compared. The Rand index can be determined by computing the variables  $a, b, c, d$  defined as follows:

- $a$  is the number of objects in  $S$  that are in the same partition in  $X$  and in the same partition in  $Y$ ,
- $b$  is the number of objects in  $S$  that are not in the same partition in  $X$  and not in the same partition in  $Y$ ,
- $c$  is the number of objects in  $S$  that are in the same partition in  $X$  and not in the same partition in  $Y$ ,
- $d$  is the number of objects in  $S$  that are not in the same partition in  $X$  but are in the same partition in  $Y$ .

The sum  $a + b$  can be regarded as the number of agreements between  $X$  and  $Y$ , and  $c + d$  as the number of disagreements between  $X$  and  $Y$ . The Rand index  $R \in [0, 1]$  expresses the number of agreements as a fraction of the total number of pairs of objects:

$$R = \frac{a + b}{a + b + c + d} = \frac{a + b}{\binom{N}{2}}$$

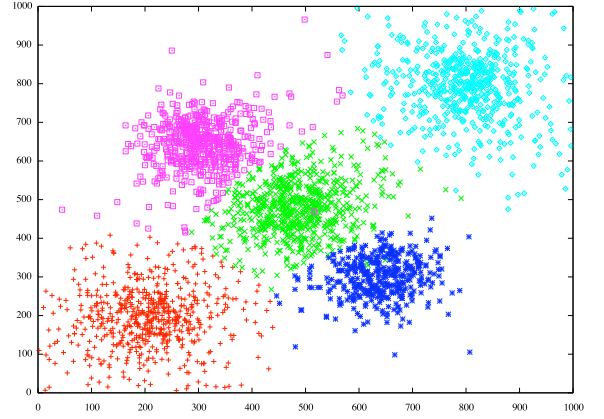
In our case, one of the two data clustering is always the one computed when  $H = 1024$ .

We have implemented in Java a simulator of the three P2P systems described in Section 3, each coupled with the two density-based clustering algorithms described in Section 4.

We have conducted extensive experiments on a desktop workstation equipped with two Intel dual-core Xeon processors at 2.6GHz and 2GB internal memory.

Two generated datasets of two-dimensional real vectors have been used in our experiments. The first dataset,  $S_0$  shown in Figure 1, has 24000 vectors generated from 5 normal densities. The second dataset,  $S_1$ , is shown in Figure 2. It has 24000 vectors generated from 5 normal densities. Three groups of 200 vectors each have been generated very close in mean, with a deviation of 10. Two groups of 10700 vectors each have been generated with a deviation of 70.

The experiments have been performed on both  $S_0$  and  $S_1$  for both method  $\mathcal{M}_1$ , with KE and GNNE estimates, and  $\mathcal{M}_2$ . Each experiment compares the three P2P networks as the number of hops varies from 1 hop to 8. For each experiment we have analysed (i) how the Rand index improves as the number of hops  $H$  increases (i.e. efficacy) and (ii) the efficiency

Figure 3: Clustering of  $S_0$  by  $\mathcal{M}_1$  with GNNE estimate and 1024 hops

measured by counting the number of messages among peers generated by the computation of density and clustering. The number of peers has been set to 1000, with 100 objects each on average.

Figure 3 shows a clustering computed on  $S_0$  by  $\mathcal{M}_1$  with GNNE estimate and 1024 hops.

## 7 Experimental Results

Figure 4 illustrates the clustering accuracy, computed by using method  $\mathcal{M}_1$  with KE density on  $S_0$ , on the increase of the number of hops (in the  $x$  axis). All P2P systems attain a very good accuracy, over 0.95, with 8 hops. The best is MURK-CAN with 0.98.<sup>1</sup> At low hop counts, MURK-SF is significantly more accurate than the other systems. Similar results, in terms of absolute accuracy, have been obtained on the same dataset by  $\mathcal{M}_1$  with GNNE density, as shown in Figure 5. In this case, MURK-SF is consistently the best system, although by a small margin. The accuracy of method  $\mathcal{M}_2$  on  $S_0$ , shown in Figure 6, is much poorer.

On dataset  $S_1$ , the same set of experiments shows a less accurate behaviour of all P2P systems and clustering methods, particularly for low hop counts, as illustrated by Figures 7, 8, 9. This is due to the higher complexity of dataset  $S_1$ , which contains both sparse and dense cluster of different size.

The first set of experiments provides some evidence for a superior efficacy of MURK-SF over CAN and MURK-CAN.

<sup>1</sup>In the sequel we will use MURK- and Torus- as synonyms.

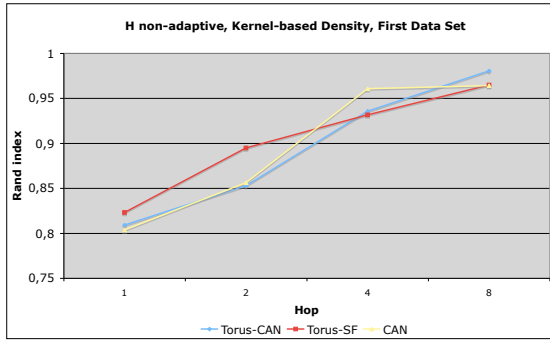


Figure 4: Accuracy of method  $\mathcal{M}_1$  with KE density on  $S_0$

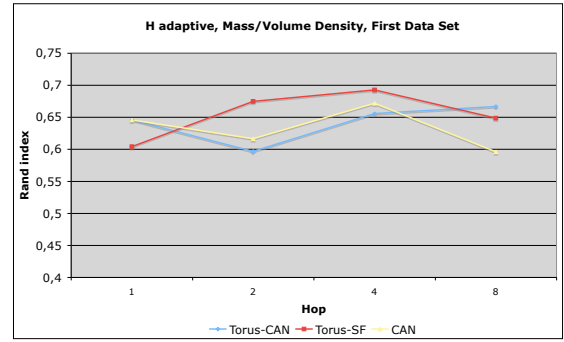


Figure 6: Accuracy of method  $\mathcal{M}_2$  on  $S_0$

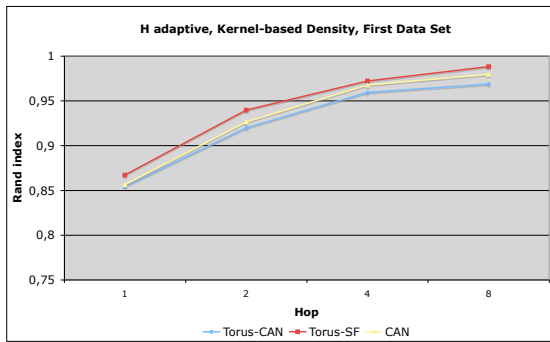


Figure 5: Accuracy of method  $\mathcal{M}_1$  with GNNE density on  $S_0$

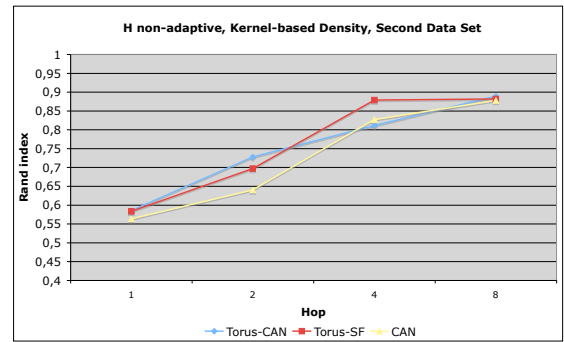


Figure 7: Accuracy of method  $\mathcal{M}_1$  with KE density on  $S_1$

Figures 10, 11, 12, 13 illustrate the network costs of  $\mathcal{M}_1$  on both datasets.

The number of messages for  $\mathcal{M}_2$  equals the number of messages for  $\mathcal{M}_1$ . However, the size of a single message is  $1/(2b/3)$  the size of a message routed in method  $\mathcal{M}_1$ , where  $b$  is the bucket size. Therefore, assuming 100 object/peer, on average network costs are lower by a factor 66. In view of this relation, the figures of network costs for  $\mathcal{M}_2$  have been omitted.

The better clustering quality of MURK-SF can be simply explained by the strategy adopted to select its neighbours according to which each peer has more neighbours than CAN and MURK-CAN better distributed in the data space. In fact, while the neighbours of a peer in CAN and MURK-CAN are those that manage a direct adjacent space partitions, the neighbours of a peer in MURK-SF can manage non contiguous space partitions guaranteeing a better view of the data space.

However, as it is shown in Figures 10–13, the network costs of MURK-SF are almost always greater than the other two P2P systems and for 4 hops the number of messages sent is more than 30% higher than the number of messages sent by MURK-CAN and CAN. At 8 hops the three systems are essentially equivalent from the view point of network costs.

To be more precise, the number of messages depicted in the Figures corresponds to the network traffic necessary to compute the density and then the clustering. The weight in terms of byte of each message depends basically on which density computa-

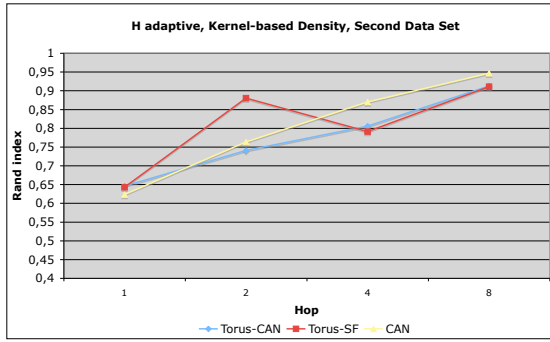
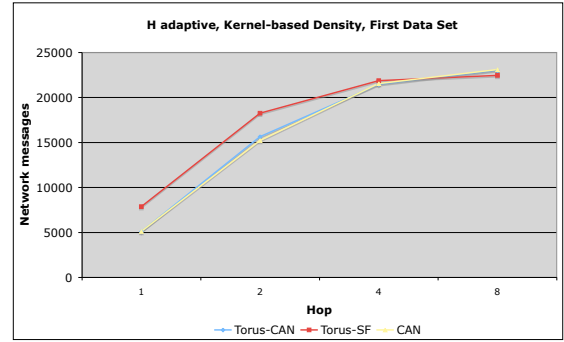
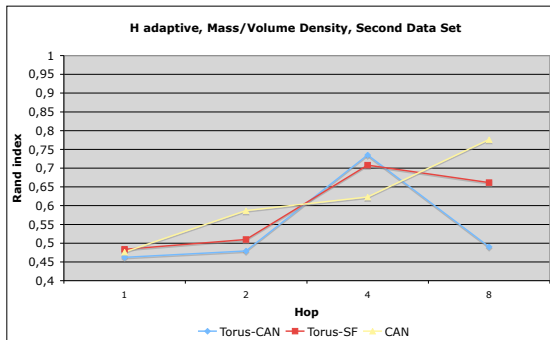
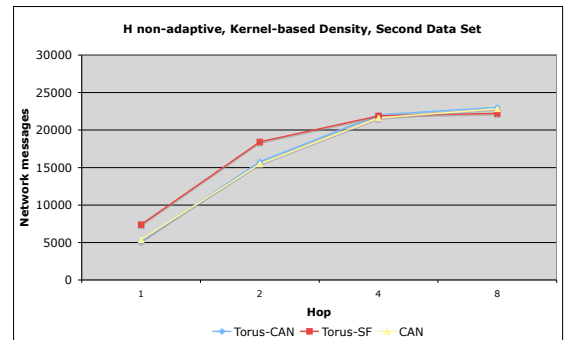
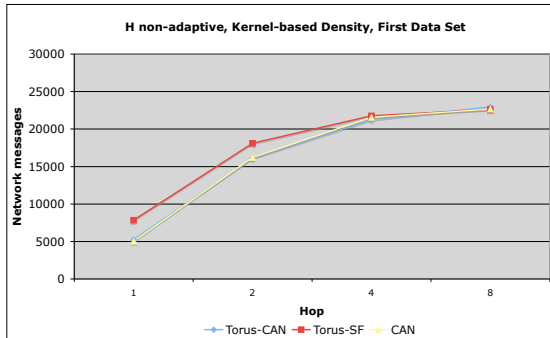
tion is adopted, in fact the traditional  $\mathcal{M}_1$  requires to transfer among peers entire data space partitions, while density in  $\mathcal{M}_2$  does not cost anything, since it is computed locally at the peer. The weight of clustering messages, independently on which density computation is selected, is negligible because peers exchange a real number corresponding to their local maximum density.

## 8 Conclusions

In this paper we have described methods to cluster data in multi-dimensional P2P networks without requiring a specific reorganization of the network and without altering or compromising the basic services of P2P systems, which are the routing mechanism, the data space partition among peers and the search capabilities.

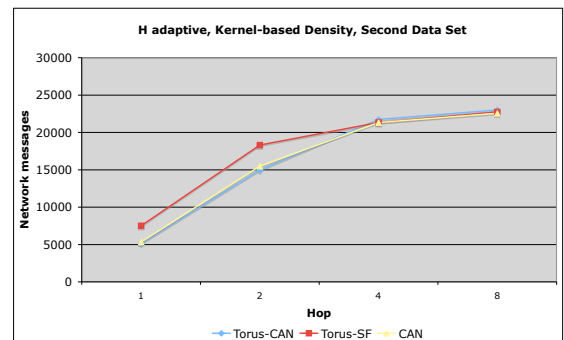
We have applied our approach, which is a density-based solution, to CAN, MURK-CAN and MURK-SF developing a simulator of the three systems. Besides a traditional computation of the density, we have experimented a novel technique in P2P systems which consist of calculating the density locally at the peer as the ratio between the mass, i.e., the number of local data objects, and the volume of local partitions.

The experiments have reported a difference of clustering quality of the two density approaches much smaller than their difference in network costs; in fact the network transmissions of the mass/volume technique are several orders of magnitude less than the traditional density-based approach, while their best

Figure 8: Accuracy of method  $\mathcal{M}_1$  with GNNE on  $S_1$ Figure 11: Network costs of  $\mathcal{M}_1$  with GNNE density on  $S_0$ Figure 9: Accuracy of method  $\mathcal{M}_2$  on  $S_1$ Figure 12: Network costs of  $\mathcal{M}_1$  with KE density on  $S_1$ Figure 10: Network costs of  $\mathcal{M}_1$  with KE density on  $S_0$ 

clustering show a quality difference of about 16 percentage points.

The methods described in this work can be extended in several directions, among which the possibility of improving the clustering quality of the mass/volume-based technique by including in the density calculated locally at the peer, an influence of its neighbour peers according to their local density. Other developments of the approach regard the adoption of new multi-dimensional indexing designed for distributed systems, both for wired environments, such as in (Moro & Ouksel 2003), and in wireless sen-

Figure 13: Network costs  $\mathcal{M}_1$  with GNNE density on  $S_1$ 

sor networks like in (Monti & Moro 2008, Monti & Moro 2009).

## References

Agostini, A. & Moro, G. (2004), Identification of communities of peers by trust and reputation, in C. Bussler & D. Fensel, eds, 'AIMSA', Vol. 3192 of *Lecture Notes in Computer Science*, Springer, pp. 85–95.

- da Silva, J. C., Klusch, M., Lodi, S. & Moro, G. (2006), 'Privacy-preserving agent-based distributed data clustering', *Web Intelligence and Agent Systems* 4(2), 221–238.
- Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. (1996), A density-based algorithm for discovering clusters in large spatial databases with noise, in 'Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)', Portland, OR, pp. 226–231.
- Ganesan, P., Yang, B. & Garcia-Molina, H. (2004), One torus to rule them all: multi-dimensional queries in p2p systems, in 'Proceedings of the 7th International Workshop on the Web and Databases (WebDB 2004)', ACM Press New York, NY, USA, pp. 19 – 24.
- Hinneburg, A. & Keim, D. A. (1998), An efficient approach to clustering in large multimedia databases with noise, in 'Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)', AAAI Press, New York City, New York, USA, pp. 58–65.
- Johnson, E. & Kargupta, H. (1999), Collective, hierarchical clustering from distributed heterogeneous data, in M. Zaki & C. Ho, eds, 'Large-Scale Parallel KDD Systems', Vol. 1759 of *Lecture Notes in Computer Science*, Springer, pp. 221–244.
- Kargupta, H. & Chan, P., eds (2000), *Distributed and Parallel Data Mining*, AAAI Press / MIT Press, Menlo Park, CA / Cambridge, MA.
- Kargupta, H., Huang, W., Sivakumar, K. & Johnson, E. L. (2001), 'Distributed clustering using collective principal component analysis', *Knowledge and Information Systems* 3(4), 422–448.  
URL: <http://citeseer.nj.nec.com/article/kargupta01distributed.html>
- Klampanos, I. A. & Jose, J. M. (2004), An architecture for information retrieval over semi-collaborating peer-to-peer networks, in 'Proceedings of the 2004 ACM symposium on Applied computing', ACM Press New York, NY, USA, pp. 1078–1083.
- Klampanos, I. A., Jose, J. M. & van Rijsbergen, C. J. K. (2006), Single-pass clustering for peer-to-peer information retrieval: The effect of document ordering, in 'INFOSCALE '06. Proceedings of the First International Conference on Scalable Information Systems', ACM, Hong Kong.
- Klusch, M., Lodi, S. & Moro, G. (2003), Distributed clustering based on sampling local density estimates, in 'Proceedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI-03', AAAI Press, Acapulco, Mexico, pp. 485–490.
- Koontz, W. L. G., Narendra, P. M. & Fukunaga, K. (1976), 'A graph-theoretic approach to nonparametric cluster analysis', *ieeetrans* C-25(9), 936–944.
- Li, M., Lee, G., Lee, W.-C. & Sivasubramaniam, A. (2006), PENS: An algorithm for density-based clustering in peer-to-peer systems, in 'INFOSCALE '06. Proceedings of the First International Conference on Scalable Information Systems', ACM, Hong Kong.
- Lodi, S., Monti, G., Moro, G. & Sartori, C. (2009), Peer-to-peer data clustering in self-organizing sensor networks, in 'Intelligent Techniques for Warehousing and Mining Sensor Network Data', IGI Global, Information Science Reference, December 2009, Hershey, PA, USA.
- Merugu, S. & Ghosh, J. (2003), Privacy-preserving distributed clustering using generative models, in 'Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003)', 19–22 December 2003, Melbourne, Florida, USA', IEEE Computer Society.
- Milojicic, D. S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S. & Xu, Z. (2002), Peer-to-peer computing, Technical Report HPL-2002-57, HP Lab.
- Monti, G. & Moro, G. (2008), Multidimensional range query and load balancing in wireless ad hoc and sensor networks, in K. Wehrle, W. Kellerer, S. K. Singhal & R. Steinmetz, eds, 'Peer-to-Peer Computing', IEEE Computer Society, Los Alamitos, CA, USA, pp. 205–214.
- Monti, G. & Moro, G. (2009), Self-organization and local learning methods for improving the applicability and efficiency of data-centric sensor networks, in 'QShine/AAA-IDEA 2009, LNICST 22', Institute for Computer Science, Social-Informatics and Telecommunications Engineering, pp. 627–643.
- Moro, G. & Ouksel, A. M. (2003), G-Grid: A class of scalable and self-organizing data structures for multi-dimensional querying and content routing in p2p networks, in 'Proceedings of Agents and Peer-to-Peer Computing, Melbourne, Australia', Vol. 2872, pp. 123–137.
- Moro, G., Ouksel, A. M. & Sartori, C. (2002), Agents and peer-to-peer computing: A promising combination of paradigms, in 'AP2PC', pp. 1–14.
- Rand, W. M. (1971), 'Objective criteria for the evaluation of clustering methods', *Journal of the American Statistical Association* 66(336), 846–850.
- Ratnasamy, S., Francis, P., Handley, M., Karp, R. & Schenker, S. (2001), A scalable content-addressable network, in 'Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications', San Diego, California, United States, pp. 161 – 172.
- Silverman, B. W. (1986), *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London.
- Tasoulis, D. K. & Vrahatis, M. N. (2004), Unsupervised distributed clustering, in 'IASTED International Conference on Parallel and Distributed Computing and Networks', Innsbruck, Austria, pp. 347–351.
- Wolff, R. & Schuster, A. (2004), 'Association rule mining in peer-to-peer systems', *IEEE Transactions on Systems, Man, And Cybernetics—Part B: Cybernetics* 34(6), 2426–2438.
- Zaki, M. J. & Ho, C.-T., eds (2000), *Large-Scale Parallel Data Mining*, Vol. 1759 of *Lecture Notes in Computer Science*, Springer.



# Stock Risk Mining by News

Qi Pan      Hong Cheng      Di Wu      Jeffrey Xu Yu      Yiping Ke

Department of Systems Engineering and Engineering Management  
The Chinese University of Hong Kong  
Shatin, N.T., Hong Kong  
Email: {qpan, hcheng, dwu, yu, ypke}@se.cuhk.edu.hk

## Abstract

Due to the fast delivery of news articles by news providers on the Internet and/or via news datafeeds, it becomes an important research issue of predicting the risk of stocks by utilizing such textual information available in addition to the time series information. In the literature, the issue of predicting stock price up/down trend based on news articles has been studied. In this paper, we study a new problem which is to predict the risk of stocks by their corresponding news of companies. We discuss the unique challenges of volatility prediction, volatility ranking and volatility index construction. A new feature selection approach is proposed to select bursty volatility features. Such selected features can accurately represent/simulate volatility bursts. A volatility prediction method is then proposed based on random walk by considering both the direct impacts of bursty volatility features on the stocks and the propagated impacts through correlation between stocks. Finally, we construct a volatility index, called VN-index, which is a time series of predicted stock volatility. Moreover, stocks are ranked based on the predicted volatility values. Such information provides investors with knowledge on how widely a stock price is dispersed from the average, as an important indicator of stock risks in a stock market. We conducted extensive experimental studies using real datasets and report our findings in this paper.

## 1 Introduction

Modern risk management system has been strongly criticized in the recent financial tsunami, where the numerous different arguments converge to one single theme: the current system failed to accurately estimate the risks of financial instruments, which were considered *to be isolated, but in many cases seemingly challenge human understanding* [8]. In stock markets, risk means the uncertainty of future outcomes, and is the probability that an investment's actual return is different from the expected value. The risk of a financial instrument is commonly estimated by the stock price volatility, which measures the variation or dispersion or deviation of an asset's returns from the mean value.

Several models (e.g., ARMA [7], GARCH [1]) were proposed to predict the future volatility based on historical stock price (time series information). However, these models cannot fully capture the bursty behavior

of stock prices, especially when there is some breaking news hitting the market.

Several studies [17, 7, 4] have discussed the GARCH forecast errors and related the errors to the arrival of asset specific news articles, i.e., the existing models cannot interpret the change of external environment (news) and therefore could not react accordingly. In [17], a classification model is designed to detect interesting news articles that would help understand the behavior of stock price volatility. However, except for some empirical studies, none of these methods attempted to incorporate news information into risk analysis, or in other words, volatility prediction and ranking.

Although there have been many existing studies [21, 16, 17] which can predict the up/down trend of stock prices, volatility prediction and ranking from news is a new and challenging problem. We highlight the unique issues of volatility prediction, and discuss why the existing work for stock trend prediction can not be directly applied.

First, volatility carries different information from a trend. Figure 1(a) and (b) show the stock volatility and stock prices during 37 trading days (from Sept. 01, 2008 to Nov. 09, 2008), respectively. We can see that there is no obvious correlation between these two time series. Some volatility bursts occur at the turning points of stock price trends (e.g., point 13 and point 27) while others appear when there is no obvious change of stock price trend (e.g., point 21). This is because that volatility is computed as the standard deviation of stock price and therefore reflects market activities from a microscope perspective. As shown in the example, dramatic changes of daily stock prices can cause volatility bursts, but stable daily stock prices do not necessarily imply a stable volatility. A stock which has very stable daily prices may have a big fluctuation in intra-day prices, thus may produce a large volatility value.

Second, the class distribution of training text samples is very skewed if we use a text categorization approach based on news articles to predict stock volatility. Consider the daily ICBC stock prices in 37 days in Figure 1. There are 12 up trends and 25 down trends, with a ratio of 1:2 between up and down trends. On the other hand, there are only 4 volatility bursts out of 37, with a ratio of 1:8 between bursty and non-bursty volatility. Furthermore, there are 185 news articles associated with the up trend, and 363 news articles associated with the down trend, with a ratio of 1:2. On the contrary, there are only 51 news articles as positive samples, associated with bursty volatility, and 497 news articles as negative samples, associated with non-bursty volatility, with a ratio of about 1:10. We have observed similar skewed distributions in our large-scale experiments as well. The small number of positive samples related to the rare volatility bursts makes the problem of predicting volatility bursts chal-

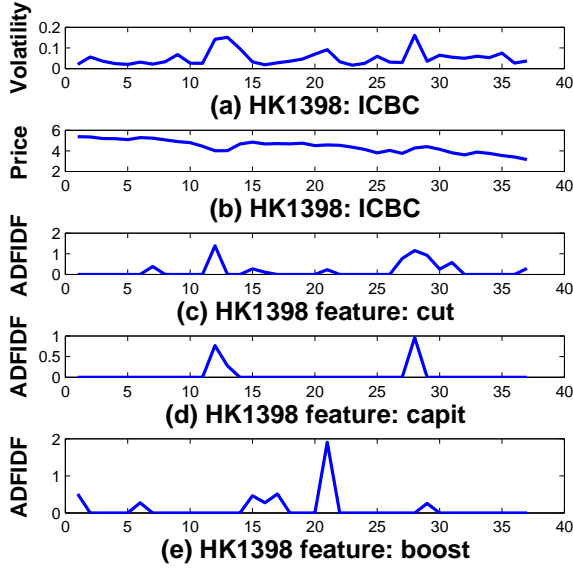


Figure 1: Volatility, Prices, and Features

lenging.

Third, volatility burst prediction shares some similarity with the prediction of the slopes of stock trends, as both problems focus on the magnitude of changes. Existing studies [21, 16, 17] can predict the up/down trend, but may not predict the slope of a trend accurately, because the available information is not sufficient. This evidence also suggests that the existing methods on trend prediction may not work on volatility prediction.

In this paper, we concentrate on volatility prediction by utilizing both time series data (stock prices) and textual information (news articles). First, the textual information is transformed into time series by using the measure ADFIDF (Adjusted Document Frequency Inverse Document Frequency). Second, representative bursty volatility features are selected based on the co-occurrences of historical stock price volatility and news articles. Third, the feature weights, which measure the degree of importance of those features for each stock are learned. Then, based on the feature weights and the incoming news, the volatility of the corresponding stock is predicted. To improve the prediction accuracy on stocks which have very limited news reports, a random walk model is used to propagate the impacts from news among stocks based on their correlation. Finally, a volatility index is constructed as a time series of predicted volatility. Stocks can be further ranked based on the predicted volatility values.

### 1.1 Main Contributions

The main contributions of the paper are summarized as follows.

- We study a new problem of predicting stock risks based on the predicted volatility by utilizing both time series information (stock price) and textual information (news articles).
- We propose a new feature selection algorithm to select bursty volatility features which have co-occurring bursty patterns with the volatility bursts of stocks. A set of such selected bursty volatility features can accurately represent the stock volatility. Feature weights are learned from historical stock prices and news articles to mea-

sure the impact of bursty keywords on stock volatility.

- We further use random walk to propagate the impacts of news among stocks based on their correlation. The random walk approach can greatly improve the volatility prediction performance for those stocks with very limited news reports. The volatility prediction and ranking methods are built on top of the random walk model.
- We conducted extensive experimental studies using real datasets and demonstrated the superiority of our approach in comparison with existing approaches.

The rest of the paper is organized as follows. The definition of stock volatility and the problem formulation are introduced in Section 2. We study bursty volatility feature selection in Section 3, and stock volatility prediction in Section 4. Section 5 presents the experimental study. Section 6 reviews some related works and background information. Finally, Section 7 concludes the paper.

## 2 Problem Statement

**Definition 1** *Volatility is the standard deviation of the continuously compounded returns of a stock within a specific time horizon and is used to measure how widely prices are dispersed from the average as follows:*

$$\sigma = \sqrt{\sum_{i=1}^n [R_i - E(R_i)]^2 P_i} \quad (1)$$

where  $R_i$  is the possible rate of return,  $E(R_i)$  is the expected rate of return, and  $P_i$  is the probability of  $R_i$ .

**Problem Statement:** Given a set of stocks  $S = \{S_1, S_2, \dots\}$  where  $S_i$  is a time series, and a set of documents  $T = \{T_1, T_2, \dots\}$  available before or at time  $t$ , we focus on predicting stock volatility and ranking stocks based on the predicted volatility at the next time unit  $t + 1$ , based on the available textual information.

## 3 Bursty Volatility Features

To predict stock volatility, we could detect breaking events from available news articles that are indicators of volatility bursts. We observe that the emergence of a breaking event is usually accompanied with a burst of features (keywords). Some features suddenly appear widely in different news articles when the event emerges whereas their occurrences drop significantly when the event fades away. By monitoring the occurrence changes of the features in news articles, we can identify whether there is any new event occurring. Then the problem is how to select a small set of features which can represent all breaking events.

As we discussed, the number of volatility bursts in a stock  $S_k$  is considerably small in comparison with the total number of up/down trends occurring in the same stock. Even though the number of news articles that are related to the volatility bursts in the stock  $S_k$ , is also observed to be small, we believe that the features in those documents can potentially predict/rank volatility bursts effectively. The desirable properties of a feature are discussed below.

**Bursty Occurrences:** An effective feature needs to be a bursty feature rather than a stable feature over a time interval. It is most likely that such bursty features can effectively represent volatility bursts.



**High Indicative Ability:** An effective feature needs to have high ability to indicate volatility bursts, i.e., the bursts of a feature need to be a good indicator of the volatility bursts of the corresponding stock. Features whose high occurrences are always accompanied with volatility bursts are more preferable than those features whose high occurrences only cause volatility bursts occasionally.

**High Coverage and Low Redundancy:** A minimal set of selected effective features needs to cover the volatility bursts as much as possible. By coverage we mean that the set of selected effective features, as a whole, captures all volatility bursts. By redundancy we mean that some selected features may give similar information.

In the following, we discuss bursty feature measurement and how to select bursty volatility features.

### 3.1 ADFIDF Measure

As each stock is representing a company, if there are some important things related to the company, the news appears immediately. Generally, the wider the news is reported, the more important the news is. If there is no bursty news, the value which measures the feature burstiness should be around average. In the following, we discuss how to capture the wideness of a text feature.

Given a set of stocks  $S = \{S_1, S_2, \dots\}$ , where a stock  $S_k = [s_{k1}, s_{k2}, \dots, s_{kT}]$  is a sequence of stock prices in the time interval  $\mathcal{I}$ . In the same time interval  $\mathcal{I}$ , there exists a set of news/documents,  $T = \{T_1, T_2, \dots\}$ , where a document  $T_i \in T$  contains a set of features  $\{f_{ij}\}_{j=1}^m$ . We assume that it is known which stock  $S_k$  a document  $T_i$  is related to. The assumption is reasonable since most financial news providers do provide such information when distributing financial news articles. Then the features in the document  $T_i$  can also be identified to which stock they are related. We represent a feature  $f$  related to a stock  $S_k$  in the time interval  $\mathcal{I}$  as a time series,  $f(k) = [f^k(1), f^k(2), \dots, f^k(\mathcal{I})]$ , where  $f^k(t)$  is defined as follows.

$$f^k(t) = \frac{DF_{k,f}(t)}{N_k(t)} \times \log \left( \frac{N_k(T)}{DF_{k,f}(T)} \right) \quad (2)$$

where  $DF_{k,f}(t)$  is the number of related documents in  $T$  containing the feature  $f$  for the stock  $S_k$  at time  $t$ , and  $N_k(t)$  is the total number of documents in  $T$  related to the stock  $S_k$  at time  $t$ . Here  $t$  is a time unit defined by user, such as one month, one day, or one hour. In this paper,  $t$  is defined as one day in our evaluation. Therefore,  $f^k(t)$  reflects the wideness of the feature  $f$  for the stock  $S_k$  at time  $t$ . In the following, we call  $f^k(t)$  the ADFIDF (Adjusted Document Frequency Inverse Document Frequency) value of a feature  $f$  related to stock  $S_k$  at time  $t$ .

We assume that all  $f^k(t)$  values,  $\forall t \in \mathcal{I}$ , form a normal distribution. Then we can identify the bursty features as well as the bursty time interval where such bursty features occur. Note that although there are many methods, e.g. [15], related to the bursty feature identification, there is no conclusion which is the best. Actually, we could use other distribution, and the final identified features would be similar. A typical normal distribution for  $f^k(t)$ , for stock  $S_k \in S$ ,  $\forall t \in \mathcal{I}$ , is shown in Figure 2, where the  $x$ -axis is the  $f^k(t)$  value, and the  $y$ -axis is its density. The value  $f^k(t) = 0$  indicates that a feature  $f$  occurs when there are no explicit events related to stock  $S_k$  at time  $t$ . The mean value of  $f^k(t)$ , denoted as  $\bar{f}^k$ , corresponds to the maximum density value of  $f^k(t)$ . We say that

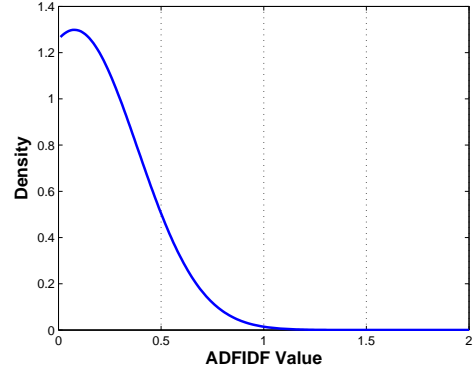


Figure 2: The  $f^k$  Normal Distribution

a feature  $f$  is a bursty feature related to stock  $S_k$ , if  $f^k \geq \delta$ , for a threshold  $\delta$ . In brief, the higher threshold, the better capability of filtering noise. However, if the threshold is set to be very high, it will miss many effective features; if the threshold is set to be very low, the noise will increase. We will test the threshold in Section 5.2.2. The bursty time interval, denoted as  $TB_f$ , for feature  $f$ , is a set of time intervals where  $f$  appears to be a bursty feature.

It is worth noting that the commonly-used measure, TFIDF (term frequency inverse document frequency) [18], cannot be used since we need the features that witness a stock in a time interval rather than the importance of the features for a document. Therefore, we use a new measure ADFIDF. The idea of DFIDF is brought from [10], but ADFIDF is different from it. The original DFIDF is computed for the whole document set, so the DF and IDF values are global. However, ADFIDF is computed for a subset of documents containing all news related to a specific stock  $S_k$ . Moreover, the ADFIDF value is computed for a specific time unit  $t$  as in  $f^k(t)$ .

### 3.2 Bursty Volatility Features

We identify all bursty features based on ADFIDF. Then we introduce a *co-occurrence rate*, denoted as  $E(S_k, f)$ , to measure how a bursty feature  $f$  and the volatility bursts of a stock  $S_k$  occur at the same time. The larger  $E(S_k, f)$  is, the more important the feature  $f$  to the stock  $S_k$ . The idea of co-occurrence is, if a feature always bursts together with the stock volatility bursts in the same time interval, the feature is valuable for identifying volatility bursts.  $E(S_k, f)$  is defined below.

$$E(S_k, f) = \frac{V(S_k, TB_f)}{|TB_f|} \bigg/ \frac{V(S_k, \mathcal{I})}{|\mathcal{I}|} \quad (3)$$

where  $V(S_k, \mathcal{I})$  is the sum of the bursty volatility values regarding stock  $S_k$  in the time interval  $\mathcal{I}$ .

$$V(S_k, \mathcal{I}) = \sum_{t \in \mathcal{I}} V(S_k, t) \quad (4)$$

where  $V(S_k, t)$  refers to bursty volatility at time  $t$  computed using Eq.(1). Recall that  $TB_f$  is the set of bursty time intervals of the feature  $f$ , and  $\mathcal{I}$  is the entire time interval. In Eq.(3), the numerator computes the average volatility over the co-occurrence time intervals of the bursty feature  $f$  and volatility bursts. The normalization by the denominator makes the co-occurrence rates of features with respect to different stocks comparable.

**Algorithm 1** FeatureRank( $S_k, F_k, \gamma$ )

INPUT: stock  $S_k$ , bursty feature set  $F_k$ , decay factor  $\gamma$   
 OUTPUT: a list of pairs  $(f_j, E(S_k, f_j))$  for  $f_j \in F_k$   
 in descending order

```

1: compute  $TB_{f_i}$  for  $f_i \in F_k$ ;
2: for all  $f_i \in F_k$  do
3:   compute  $E(S_k, f_i)$  using Eq. (3);
4: end for
5:  $\mathcal{E} \leftarrow \emptyset$ ;
6: while  $F_k \neq \emptyset$  do
7:   sort  $F_k$  in decreasing order based on  $E(S_k, f_i)$ ;
8:   let  $f$  be the first feature in the sorted  $F_k$ ;
9:   remove  $f$  from  $F_k$ ;
10:  append the pair  $(f, E(S_k, f))$  into  $\mathcal{E}$ ;
11:  for all  $f_j \in F_k$  do
12:     $B = TB_{f_j} \cap TB_f$ ;
13:    if  $B \neq \emptyset$  then
14:       $V(S_k, t) \leftarrow \gamma \cdot V(S_k, t)$  for  $t \in B$ ;
15:      update  $E(S_k, f_j)$  based on Eq. (3);
16:    end if
17:  end for
18: end while
19: return  $\mathcal{E}$ ;

```

**3.3 Bursty Volatility Features Selection**

In the previous subsections, we have discussed ADFIDF for feature burstness measure and the co-occurrence rate  $E(S_k, f)$  for indicative ability measure for a feature  $f$ . We will discuss how to select a compact set of bursty volatility features to ensure high coverage and low redundancy.

Consider the example in Figure 1. There are three volatility bursts of the ICBC stock, denoted as  $S_k$ , shown in Figure 1(a). In addition there are three ADFIDF sequences of bursty features “cut” (denoted as  $f_x$ ), “capit” ( $f_y$ ) and “boost” ( $f_z$ ), shown in Figure 1(c)-(e), respectively. Here, the two ADFIDF sequences  $f_x^k$  and  $f_y^k$  have 2 similar bursts corresponding to 2 out of 3 volatility bursts of  $S_k$ , and the only burst in  $f_z^k$  corresponds to the remaining volatility burst in Figure 1(a). The three bursty volatility features,  $f_x$ ,  $f_y$ , and  $f_z$ , together cover the three volatility bursts in  $S_k$ . By “cover”, we mean that the features jointly represent the volatility information about  $S_k$ . Assume  $E(S_k, f_x) = E(S_k, f_y) \geq E(S_k, f_z)$  and the goal is to select top-2 bursty volatility features. If we select both  $f_x$  and  $f_y$ , then one of them is considered as redundant and the third volatility burst cannot be captured.

In order to select a set of representative bursty volatility features, we design an algorithm to rank all bursty volatility features such that the top- $k$  features, to be selected from the ranking list, will be more likely to accurately capture the corresponding volatility bursts of a stock. The algorithm FeatureRank is outlined in Algorithm 1. The main idea is to reduce  $E(S_k, f_y)$  if its burst time interval  $TB_{f_y}$  is overlapped with another  $TB_{f_x}$  for a higher ranked feature  $f_x$ , using a feature decay factor  $\gamma$ . As shown in Algorithm 1, it takes a stock  $S_k$ , a set of bursty features  $F_k$  related to  $S_k$ , and a feature decay factor  $\gamma$ . It computes the bursty time interval  $TB_{f_i}$  for every feature  $f_i \in F_k$  (line 1), and then computes  $E(S_k, f_i)$  using Eq.(3) (lines 2-4). Let  $\mathcal{E}$  keep a list of pairs  $(f_j, E(S_k, f_j))$  in descending order of  $E(S_k, f_j)$ . In a while loop (lines 6-18), in every iteration, it selects the top bursty volatility feature  $f$  from  $F_k$  and appends the pair  $(f, E(S_k, f))$  to  $\mathcal{E}$ . It then recomputes  $E(S_k, f_j)$  for all remaining  $f_j \in F_k$  using the decay factor  $\gamma$ , if there is an overlap between the burst time

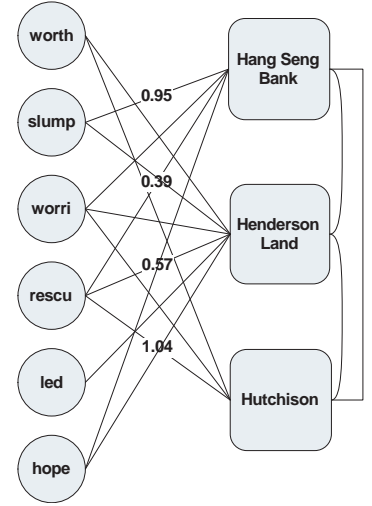


Figure 3: Volatility Prediction Based on Random Walk

interval  $TB_f$  of the selected feature  $f$  and  $TB_{f_j}$  of the feature  $f_j$ .

According to Algorithm 1, the top-2 bursty volatility features in Figure 1 would be either  $f_x$  (“cut”) or  $f_y$  (“capit”) plus  $f_z$  (“boost”).

**4 Volatility Prediction**

We have discussed how to select bursty volatility features in Section 3. Such bursty volatility features are selected based on how the burst features in documents co-occur with the volatility bursts in stocks. In this section, we discuss how such selected bursty volatility features are used to predict the stock volatility. The bursty volatility features can have both direct impacts on stock volatility and propagated impacts on stock volatility through stock-stock correlation, as volatility of a stock may affect and be affected by others.

To predict the stock volatility at time  $t$ , we use news articles which arrive before  $t$ . For example, to predict stock volatility on a particular day, we collect news articles which appear before 10:00AM on that day (market opening time) for prediction. The news articles which appear after 10:00AM will be used for next day prediction.

**4.1 Graph Construction**

We construct an edge-weighted node-labeled graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  where  $\mathcal{V} = \mathcal{V}_F \cup \mathcal{V}_S$  is a set of nodes,  $\mathcal{V}_F$  represents the set of bursty volatility features, and  $\mathcal{V}_S$  represents the set of stocks.  $\mathcal{E} = \mathcal{E}_{FS} \cup \mathcal{E}_{SS}$  is a set of edges, where  $\mathcal{E}_{FS}$  represents a set of edges from a node in  $\mathcal{V}_F$  to a node in  $\mathcal{V}_S$ , and  $\mathcal{E}_{SS}$  represents a set of edges from a node in  $\mathcal{V}_S$  to another node in  $\mathcal{V}_S$ . A node in  $\mathcal{V}$  is associated with a unique label, so we treat labels as node identifiers. The edge weight on an edge  $(v_f, v_s) \in \mathcal{E}_{FS}$  represents the impact of a bursty volatility feature  $v_f \in \mathcal{V}_F$  to a stock  $v_s \in \mathcal{V}_S$ . The higher the weight, the larger impact of the feature on the stock. The edge weight on an edge  $(v_{s1}, v_{s2}) \in \mathcal{E}_{SS}$  represents the degree of co-occurrences of volatility bursts between two stocks  $v_{s1}$  and  $v_{s2}$  in  $\mathcal{V}_S$ . The higher the weight, the more co-occurrences of the volatility bursts of two stocks.

Figure 3 shows a simple graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ . Here,  $\mathcal{V}_F$  contains 6 bursty volatility features, “worth”, “slump”, “warri”, “rescu”, “led”, and “hope”.  $\mathcal{V}_S$  contains 3 stocks, “Hang Seng Bank”, “Henderson Land”, and “Hutchison”. Table 1 shows the edge

Feature	Hang Seng Bank	Henderson Land	Hutchison
worth	0	0.88213	1.2434
slump	0.94723	0.61459	0
worri	0.98096	0.72786	0.33304
rescu	0.38712	0.56985	1.0362
led	0	0.61459	0
hope	0.63762	0.68553	0

Table 1: Impacts of Bursty Volatility Features

weights from a feature (a node in  $\mathcal{V}_F$ ) to a stock (a node in  $\mathcal{V}_S$ ) for Figure 3. The feature “rescu” is linked to all three stocks with different weights, which means that all these stocks are influenced by the feature “rescu”. Some features may only have impacts on a subset of stocks. For example, the feature “led” does not have any impacts on “Hang Seng Bank” or “Hutchison”, so there is no edge from “led” to “Hang Seng Bank” or “Hutchison”.

Based on the graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ , we perform random walk and calculate the volatility of a stock at time  $t+1$  based on the available bursty feature information at time  $t$ , as well as predicted volatility of correlated stocks, as in Eq.(5).

$$\mathbf{V}(S_k, t+1) = \alpha \sum_{(f_i, S_k) \in \mathcal{E}_{FS}} f_i^k(t) \cdot E(S_k, f_i) + (1-\alpha) \sum_{j \neq k} \rho(S_j, S_k) \cdot \mathbf{V}(S_j, t+1) \quad (5)$$

Here, the first part measures the accumulated direct impacts from bursty volatility features to a stock. This is the information we captured from news to stocks. Recall that  $f_i^k(t)$  is the ADFIDF value to indicate how the feature  $f_i$  is related to stock  $S_k$  at time  $t$  (Eq.(2)), and  $E(S_k, f_i)$  is the co-occurrence rate to measure how feature bursts of  $f_i$  and volatility bursts of  $S_k$  occur at the same time. The second part captures the propagated volatility bursts from correlated stocks  $S_j$  based on random walk, as stocks may affect each other in the stock market. This part can also improve volatility prediction for stocks which have very little related news. The correlation factor  $\rho(S_j, S_k)$  is computed as follows.

$$\rho(S_j, S_k) = \frac{\sum_{\tau=1}^t (V(S_j, \tau) - \overline{V(S_j)})(V(S_k, \tau) - \overline{V(S_k)})}{\sigma_{V(S_j)} \sigma_{V(S_k)}} \quad (6)$$

Here,  $V(S_k, \tau)$  is the bursty volatility at time  $\tau$  computed using Eq.(1).  $\overline{V(S_j)}$  and  $\overline{V(S_k)}$  are the mean volatility values of the two stocks  $S_j$  and  $S_k$ , respectively.  $\sigma_{V(S_j)}$  and  $\sigma_{V(S_k)}$  are the standard deviation of volatility for the two stocks in the time interval  $[1, t]$ .

## 4.2 Volatility Prediction

Based on the graph and random walk model, we discuss the procedure of volatility prediction. Volatility prediction involves two phases, namely a training phase and a testing phase. The training phase is done based on a set of documents (news articles)  $T$ , and a set of stocks  $S$ , obtained in the time interval  $[1, \mathcal{I}]$ . The testing phase is, given a set of new documents  $T'$ , on a time step  $t$ , to predict stocks volatility on the next time unit  $t+1$ .

The training phase is done as follows. First, for each stock  $S_k \in S$ , we compute the volatility over the time interval  $[1, \mathcal{I}]$ , denoted as  $V(S_k) = [\sigma_1, \sigma_2, \dots, \sigma_{\mathcal{I}}]$ , where  $\sigma_i$  is computed using Eq.(1). Then we determine a set of bursty features  $F_k = \{f_1, f_2, \dots\}$ , where  $f_i \in F_k$  corresponds to a time

series of ADFIDF  $f_i^k = [f_i^k(1), f_i^k(2), \dots, f_i^k(\mathcal{I})]$  in the time interval  $[1, \mathcal{I}]$ .  $f_i^k(t)$ ,  $t \in [1, \mathcal{I}]$ , is computed using Eq.(2). Second, we compute the co-occurrence rate  $E(S_k, f_i)$  for every  $f_i \in F_k$  using Eq.(3). Third, we obtain a list of pairs  $(f_i, E(S_k, f_i))$  using Algorithm 1 to rank the features with a decay factor  $\gamma$ . Finally, we compute the correlation  $\rho(S_j, S_k)$  between two stocks  $S_j$  and  $S_k$  using Eq. (6).

The testing phase is done as follows. Suppose that we obtain a set of new documents  $T'$ , at time  $t$ . First, we compute  $f_i^k(t)$  for every  $f_i$  in a document in  $T'$ , that is related to  $S_k$ . Second, we construct an edge-weighted graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$ .  $\mathcal{V} = \mathcal{V}_F \cup \mathcal{V}_S$ , where  $\mathcal{V}_F$  is the set of features that both appear in  $T'$  and are burst volatility features obtained in the training phase. The edge weight for an edge  $(f_j, S_k)$ , from a bursty volatility feature  $f_i$  to a stock  $S_k$  is assigned as  $E(S_k, f_i)$  which is computed in the training phase. The edge weight for an edge  $(S_j, S_k)$ , between two stock nodes, is assigned as  $\rho(S_j, S_k)$  computed in the training phase. An example is illustrated in Figure 3. Third, we compute  $\mathbf{V}(S_k, t+1)$ , for every  $S_k \in S$ , using Eq.(5) iteratively, until it converges based on random walk.

## 4.3 Volatility Index and Volatility Ranking

Based on the predicted stock volatility, we could perform two analytical tasks: volatility index construction and stock volatility ranking.

A volatility index for stock  $S_k$  in the time interval  $\mathcal{I}$  is a time series of predicted volatility values:  $NI(S_k) = [\mathbf{V}(S_k, 1), \mathbf{V}(S_k, 2), \dots, \mathbf{V}(S_k, \mathcal{I})]$ . We call it VN-index since it is a volatility index constructed from news. If the predicted volatility is accurate, the correlation between VN-index and the real volatility sequence in the testing period should be large. The correlation is quantitatively measured using the Pearson correlation coefficient.

$$\rho(NI(S_k), V(S_k)) = \frac{\text{cov}(NI(S_k), V(S_k))}{\sigma_{NI(S_k)} \sigma_{V(S_k)}} \quad (7)$$

where  $NI(S_k)$  is the volatility index sequence,  $V(S_k)$  is the real volatility sequence, and cov means covariance. We will evaluate the quality of the constructed VN-index in Section 5.1.

Besides the volatility index construction, we can further rank stocks based on their predicted volatility values  $\mathbf{V}(S_k, \mathcal{I}+1)$  for each stock  $S_k \in S$ . The ranking quality will also be evaluated based on the ground truth from the real volatility value in Section 5.2.

## 5 Experimental Study

In this section, we evaluate our proposed volatility prediction approach through two groups of experiments: volatility index construction and volatility ranking.

We archive the minute-level intra-day stock prices and the news articles from the Hong Kong Exchange Market and Don Jones Factiva database<sup>1</sup> from Jan. 1, 2008 to Dec. 31, 2008, respectively. All 42 component stocks for Hang Seng Index (HSI) are selected, which are the most influential and most widely held public stocks in Hong Kong. At each day  $t$ , the daily realized volatility is computed by applying Eq.(1) on the 1-minute time series.

<sup>1</sup><http://www.factiva.com/>

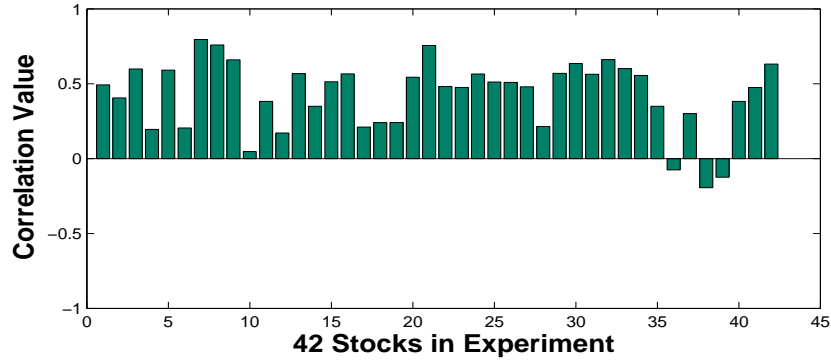


Figure 4: Prediction based on Bursty Features

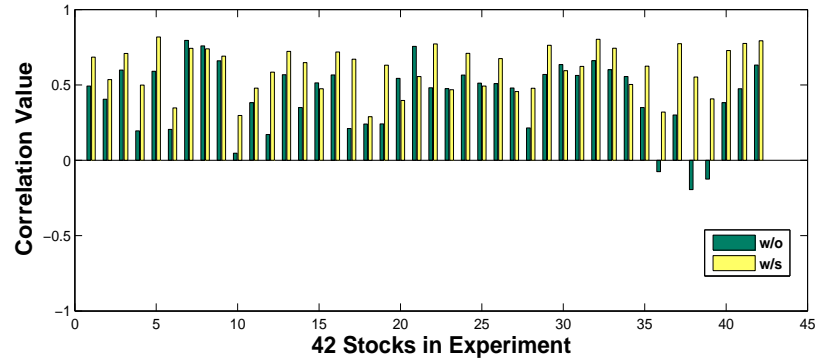


Figure 5: Prediction based on Random Walk

In total, over 150,000 news articles are collected. Each news article is related to a specific stock according to Factiva's classification system. Besides, we only tag news articles which appear before 10:00AM as the news article at that day for prediction, since most newspapers will release their news story before the market opens. The news articles which appear after 10:00AM will be labeled as the news articles of next day for prediction, e.g., the news release at 7:00PM will be used for next day prediction.

For the preprocessing of these news articles, all features are stemmed using the Porter stemmer. Features are words that appear in the news articles, with the exception of digits, web page address, email address and stop words. Features that appear more than 80% of the total news articles in a day are categorized as stop words. Features that appear less than 5% of the total news articles in a day are categorized as noisy features. Both the stop words and noisy features are removed. All data from Jan. 1, 2008 to Dec. 30, 2008 are used for training, and the data from Dec. 1, 2008 to Dec. 31, 2008, are used for evaluation.

We perform the experiments on a PC with a Pentium IV 3.4GHz CPU and 2GB RAM.

### 5.1 Volatility Index Construction

In this part, we construct the VN-index based on predicted volatility values and evaluate the quality. We focus on the following questions:

- (1) What are the effects of the proposed techniques (e.g., direct impacts from bursty volatility features versus propagated impacts based on random walk) in our algorithm? How much improvement can each of them contribute respectively?
- (2) What is the overall quality of the VN-index compared with the ground truth?

#### 5.1.1 Prediction based on Bursty Features

First, the VN-index is constructed purely based on the news information without taking the stock-stock correlation into consideration. That means the predicted volatility is computed by setting  $\alpha = 1$  in Eq.(5).

The result is shown in Figure 4. Each column in the figure represents a correlation value between the real stock volatility and the VN-index for a stock. The average correlation value is 0.4252, the maximum one is 0.7951, and the minimum one is -0.1944. Although the overall performance looks good (note that the average value of correlation between stocks is only 0.4094), the performance varies dramatically for different stocks.

We further analyze the result for those stocks whose correlation is very low, i.e., the predicted volatility is inaccurate. We find that for those stocks, their related features are much less than the average number. When a stock's related features are not sufficient to describe the stock price changes, the volatility prediction based on news is inaccurate.

#### 5.1.2 Prediction based on Random Walk

To improve the prediction for stocks which have very insufficient news reports, we exploit the stock-stock correlation through random walk to propagate the news impacts. In this experiment, the VN-index is constructed based on Eq.(5).

As shown in Figure 5, the left columns are the correlation results based on volatility prediction from news only, while the right columns are the correlation results based on both news direct impacts and propagated impacts from random walk. When random walk is added to the prediction model, the average correlation is 0.6021, which improves a lot from 0.4252. In addition, the correlation value for every stock is



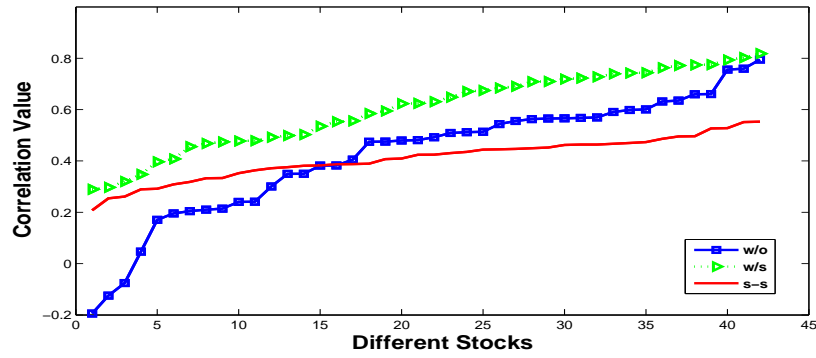


Figure 6: Comparison with Correlation between Stocks

positive.

### 5.1.3 Comparison with Stock-Stock Correlation

We further compare the correlation of the predicted volatility and the true volatility and the correlation between stocks. As shown in Figure 6, 's-s' means the correlation value between one stock and the other 41 stocks in the whole year of 2008. 'w/s' and 'w/o' represent the results with random walk and without random walk, respectively.

The mean value of correlation between stocks is 0.4094. For our approach without random walk, the average correlation between stock volatility and VN-index is 0.4252. When random walk is applied, the performance is even better. That means the co-movement of VN-index and stock volatility is better than the co-movement of volatility of different stocks in the stock market.

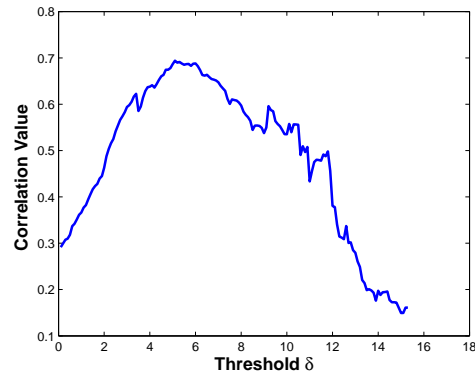
### 5.1.4 The Effect of Feature Selection

In this section, we evaluate the effectiveness of feature bursts and the impacts of the threshold  $\delta$  on the correlation value. As shown in Figure 7, the influence of the threshold for a single stock is large. When  $\delta = 0$ , all features are used. When  $\delta$  is large enough, no feature is selected. The  $y$ -axis is the correlation between the predicted volatility by news and the real daily volatility. When  $\delta = 5$ , the correlation achieves the maximum, while at both ends (i.e., when  $\delta$  is very small or large), the correlation is much lower. The experimental results show that, selecting too many features (i.e., when  $\delta$  is very small) actually decreases the correlation, as many selected features are not bursty features that are indicative of volatility bursts. In an extreme when  $\delta = 0$ , all possible features from the news articles are used. We can see that the correlation value is actually very low. Similarly, selecting too few features (i.e., when  $\delta$  is very large) also downgrades the correlation as some of the truly bursty features are not selected. In this paper, we set the threshold  $\delta$  for each stock as the value which provides the highest correlation value of bursty feature and stock price volatility in the training dataset.

## 5.2 Volatility Ranking

In this section, we evaluate the quality of ranking stocks based on their predicted volatility values. We focus on the following questions:

- (1) How does our approach compare with other approaches in volatility ranking?

Figure 7: The Influence of  $\delta$  for A Single Stock

- (2) What are the effects of the proposed techniques (e.g., bursty feature, random walk) in our algorithm? How much improvement can each of them contribute respectively?
- (3) If we combine our approach with traditional approach purely based on historical stock price data (e.g., GARCH), can we achieve any further improvement?

In this experiment, we use a much smaller training set for evaluation. Specifically, the data from Sept. 01, 2008 to Oct. 24, 2008 are used for training, and the data from Oct. 25 to Nov. 09, 2008 are used for evaluation.

### 5.2.1 Ranking Quality Comparison

We compare our proposed volatility ranking approach, denoted as VbN, for Volatility-by-News, with the following approaches.

- **Random Selection:** The volatility rank list is formed based on random selection. The accuracy is the statistical mean accuracy value for ranking.
- **Baseline Model:** The volatility rank list is formed based on average volatility on the training set.
- **GARCH:** We apply GARCH model to predict the volatility of next day and rank the stocks based on the predicted volatility. We use a five year daily stock data for training GARCH model, because if the time series is not long enough, the performance will be bad. The UCSD Garch toolbox is used in the experiments.

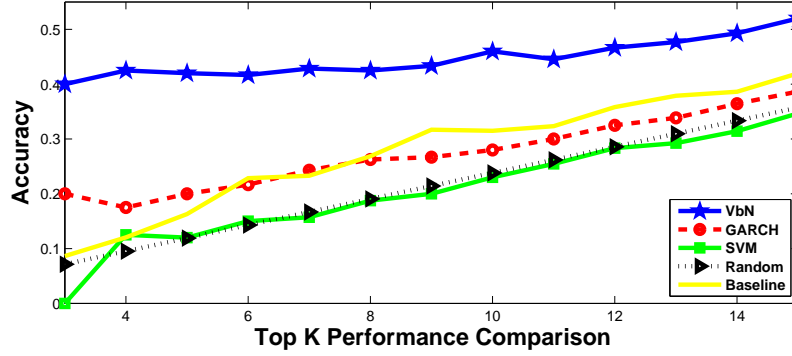


Figure 8: Volatility Ranking Comparison

Normalized Real Volatility	VbN	GARCH	SVM
CHALCO	Li & Fung	Esprit Hldgs	Esprit Hldgs
Li & Fung	CHALCO	Li & Fung	FIH
New World Dev	Esprit Hldgs	HK & China Gas	CITIC Pacific
Esprit Hldgs	FIH	New World Dev	CHALCO
COSCO Pacific	Henderson Land	Sino Land	Sinopec Corp
MTR Corporation	COSCO Pacific	Hutchison	CNOOC
Sino Land	Hutchison	China Shenhua	CLP Hldgs
China Mer Hldgs	HK & China Gas	Henderson Land	Hutchison
Cathay Pac Air	China Mer Hldgs	FIH	Hang Seng Bank
Hang Lung Prop	Sino Land	Cheung Kong	Li & Fung
China Unicom	New World Dev	HKEx	BOC Hong Kong
CITIC Pacific	Cathay Pac Air	Hang Seng Bank	Bank of E Asia
China Resources	Yue Yuen Ind	China Mer Hldgs	China Resources
Yue Yuen Ind	Hang Seng Bank	Cathay Pac Air	SHK Prop
Henderson Land	Cheung Kong	CHALCO	ICBC

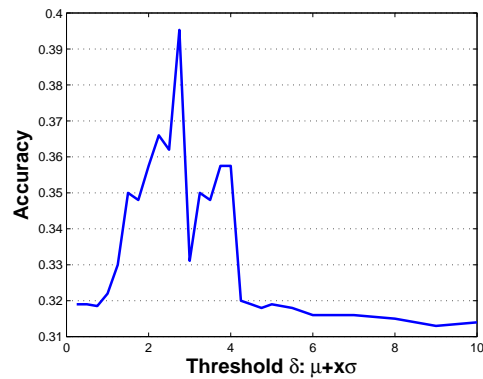
Table 2: Ranking Result Comparison

- **SVM**: we label news articles as positive and negative based on whether the volatility bursts occur after the news release, using a similar approach as in [17]. We use the most promising text classification model support vector machine (SVM) [12], to train the text classifier. Based on the classifier and the news features in the testing phase, the volatility of stocks is ranked.

In this experiment, since stocks have volatility in different scales, all the predicted and real volatility values are normalized by their mean value and are transform into relative volatility. The accuracy is measured by overlap-similarity [19],  $OS(\tau_1, \tau_2)$ , which indicates the degree of overlap between the top  $n$  volatility stocks of the two rankings  $\tau_1$  and  $\tau_2$ , where  $\tau_1$  is the ranking computed by a model, and  $\tau_2$  is the actual ranking from ground truth. The overlap of two stock sets  $A$  and  $B$  (each of size  $k$ ) is defined as  $\frac{|A \cap B|}{k}$ . As a case study, Table 2 shows a comparison among VbN, GARCH, and SVM against the ground truth, using top-15 stocks that have high volatility bursts in the next time unit. In Table 2, stocks are ranked in descending order of the corresponding volatility value. For results in Table 2, the overlap-similarity between VbN and the ground truth (normalized real volatility) is 0.67, larger than that between GARCH/SVM and the ground truth, which are 0.53 and 0.33, respectively.

We further test VbN in comparison with the other four methods by varying  $k=3-15$  in the top- $k$  ranking list. Figure 8 shows the mean value of accuracy comparison between different methods over the entire testing period. From Figure 8, when  $k$  is small ( $k=3$ ), the accuracy of VbN is 40% higher than SVM, 25% higher than Random Selection, and 20% higher than GARCH. When  $k$  increases, the accu-

racy of all methods increase, but VbN outperforms the other methods in all cases. When  $k=15$ , VbN achieves an accuracy of 50% which is around 15% higher than other approaches.

Figure 9: Volatility Ranking with Different  $\delta$ 

### 5.2.2 Volatility Ranking based on Bursty Features

In this experiment, we evaluate the effectiveness of bursty features and the impacts of the threshold  $\delta$  on volatility ranking. If  $\delta=0$ , all related features are included in the bursty feature set. On the other hand, if  $\delta$  is set to a large value, there may not be any bursty feature being selected. Figure 9 shows the results. The  $x$ -axis is in a range of  $\mu + x\sigma$ , where  $\mu$  is the mean of ADFIDE,  $\sigma$  is its deviation, and  $x$  is an integer in the range of  $[0, 10]$ .  $y$ -axis is the average accuracy of top- $k$  results for  $k=1-15$ . As shown

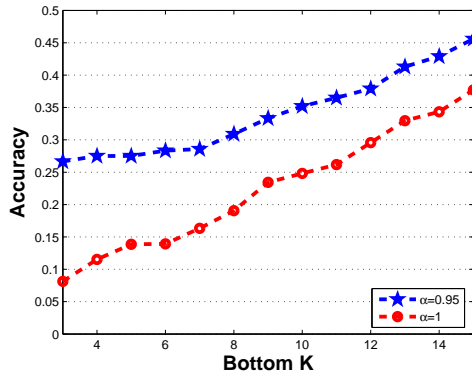


Figure 10: Volatility Ranking by Random Walk

in Figure 9, capturing bursty features is a very important factor for ranking stocks based on volatility. When  $\delta = \mu + 2.75\sigma$ , the ranking accuracy is the highest (39.5%), which is 7.7% higher than using all the bursty features (31.8%) and 8% higher than using no news information (31.5%). As the purpose of this experiment is to measure the effect of bursty features, we set  $\alpha = 1$  in Eq.(5).

### 5.2.3 Volatility Ranking by Random Walk

We evaluate the effectiveness of VbN based on random walk with  $\alpha = 0.95$  in Eq.(5). We observe that the ranking of all component stocks in Hang Seng Index is not noticeably affected by varying  $\alpha$ . It is because that the component stocks of Hang Seng Index are reported intensively. Therefore, the improvement based on propagated impacts by random walk is not obvious. In this experiment, we test another set of 42 stocks including 11 HSI-component stocks and 31 non HSI-component stocks that only receive few news articles from time to time. Figure 10 shows the mean value accuracy comparison for the bottom- $k$  out of the 42 stocks, when  $\alpha = 1$  (without random walk) and  $\alpha = 0.95$  (with random walk) over the entire testing period. As seen in Figure 10, when  $\alpha = 0.95$  (with random walk), its accuracy becomes 27.5% which is 20% higher than the accuracy when  $\alpha = 1$  (without random walk). When we increase the  $k$  value of the bottom- $k$  stocks from 1 to 15, the smallest accuracy margin is still as large as 10%, which indicates random walk is effective to improve the accuracy for ranking stocks that do not frequently receive news articles.

## 6 Related Works

The first systematic examination on the impacts of textual information on the financial markets was conducted in [13], which compared the movements of Dow Jones Industrial Average with general news during the period from 1966 to 1972. [5] formulated an *activity monitoring task* for predicting the stock price movements, which issued alarms based on the content of the news articles. [21] developed an online system for predicting the opening prices of five stock indices, where by combining the weights of the keywords from news articles and the historical closing prices of a particular index, some probabilistic rules were generated using the approach in [20]. [6] proposed a model for mining the impact of news stories on the stock prices, by using a  $t$ -test based split and merge segmentation algorithm for time series preprocessing and SVM [12] for impact classification. [17] discovered a relationship between the news and abnormal stock prices behavior. But it focused more on how to detect these

influential news using text categorization.

As aforementioned, the problem of volatility prediction is different from trend prediction. In this work, instead of studying the news articles, we attempt to capture the breaking events by finding a set of representative bursty features which can describe the bursts of stock price volatility.

So far, there have been many studies related the topic of bursty feature detection [15, 11, 10]. For example, [15] proposed an algorithm for constructing a hierarchical structure for the features in the text corpus by using an infinite-state automaton. Similar to these approaches, our proposed algorithm is based on the bursty features. Different from these approaches, we also require the bursty features to be concurrent and have a good coverage over the bursty periods of stock price volatility.

For the web graph, the traditional link analysis methods PageRank [2] and HITS [14] attempted to calculate the importance of a Web page based on the scores of the pages pointing to that page. The rank vector can be computed by repeatedly iterating over the web graph structure until a stable assignment of page importance is obtained. [3] used a bipartite graph to model the process of news generation and built a model to rank the news articles and the sources that generate them. [9] used a set of biased initial restart probability vectors in computing PageRank. They attempted to capture a more accurate importance score with respect to a particular topic. In order to consider indirect influence of features through correlated stocks, this work constructs a graph based on the correlation between stocks and takes the influence from bursty features as the initial starting probability in computing the final volatility rank.

## 7 Conclusion

In this paper, we studied a new research problem of predicting and ranking stock volatility based on news, where volatility is an important stock risk measure. We discussed the unique challenges of volatility prediction/ranking, and showed that the existing approaches on stock trend prediction cannot effectively solve our problem. We defined the bursty volatility features and proposed an algorithm to select a set of highly indicative bursty volatility features to represent volatility bursts. The main idea is to utilize features in news articles to strengthen the prediction and ranking of volatility. In addition, we proposed a random walk based approach that propagates the news impacts through correlated stocks. We conducted extensive performance study on volatility index construction and stock volatility ranking using real datasets and demonstrated the effectiveness of our proposed approach.

## 8 Acknowledgements

The work was supported by grants of the Research Grants Council of the Hong Kong SAR, China No. 419008 and 419109, and the Chinese University of Hong Kong Direct Grant No. 2050446.

## References

- [1] T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, April 1986.
- [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] G. M. D. Corso, A. Gullí, and F. Romani. Ranking a stream of news. In *Proc. of WWW'05*, pages 97–106, 2005.
  - [4] L. H. Ederington and J. H. Lee. The short-run dynamics of the price adjustment to new information. *The Journal of Financial and Quantitative Analysis*, 30(1):117–134, 1995.
  - [5] T. Fawcett and F. J. Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proc. of KDD'99*, pages 53–62, 1999.
  - [6] G. P. C. Fung, J. X. Yu, and H. Lu. The predicting power of textual information on financial markets. *IEEE Intelligent Informatics Bulletin*, 5(1):1–10, 2005.
  - [7] R. Gencay, M. Dacorogna, U. A. Muller, O. Pictet, and R. Olsen. *An Introduction to High-Frequency Finance*. Academic Press, 2001.
  - [8] Greenspan. Excerpts from greenspan speech on global turmoil. In *The Markets, reprinted in The New York Times*, November 6, 1998.
  - [9] T. H. Haveliwalla. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15:784–796, 2003.
  - [10] Q. He, K. Chang, and E.-P. Lim. Analyzing feature trajectories for event detection. In *Proc. of SIGIR'07*, pages 207–214, 2007.
  - [11] Q. He, K. Chang, E.-P. Lim, and J. Zhang. Bursty feature representation for clustering text streams. In *SDM'07*, 2007.
  - [12] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proc. of ECML'98*, pages 137–142, 1998.
  - [13] F. Klein and J. A. Prestbo. *News and the Market*. Chicago: Henry Regenry, 1974.
  - [14] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.
  - [15] J. M. Kleinberg. Bursty and hierarchical structure in streams. In *Proc. of KDD'02*, pages 91–101, 2002.
  - [16] V. Lavrenko, M. D. Schmill, D. Lawire, P. Ogivie, D. Jensen, and J. Allan. Mining of Concurrent Text and Time Series. In *Proc. of KDD'00 Workshop on Text Mining*, 2000.
  - [17] C. Robertson, S. Geva, and R. C. Wolff. Can the content of public news be used to forecast abnormal stock market behaviour? In *Proc. of ICDM'07*, pages 637–642, 2007.
  - [18] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.
  - [19] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006.
  - [20] B. Wüthrich. Probabilistic knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):691–698, 1995.
  - [21] B. Wuthrich, D. Permunetilleke, S. Leung, V. Cho, J. Zhang, and W. Lam. Daily prediction of major stock indices from textual www data. In *Proc. of KDD'98*, pages 364–368, 1998.



## Author Index

- Antoine, Elizabeth, 37
- Bellahsene, Zohra, 151  
Benatallah, Boualem, 141  
Bennett, Kerry, 3  
Bertino, Elisa, 93  
Bezdek, James C., 123  
Bjorklund, Truls A., 57  
Bouguettaya, Athman, iii
- Cao, Jinli, 133  
Cheema, Muhammad Aamir, 75  
Cheng, Hong, 179  
Chi, Yunxiang, 93  
Chrysanthis, Panos K., 103
- Deng, Ke, 47
- Flender, Christian, 67
- Grimsno, Nils, 57
- Harris, Michael C., 9  
Hoang, Dat Dac, 141
- Jiang, Jian, 161
- Kabir, Md Enamul, 93  
Kalinov, Pavel, 113  
Ke, Yiping, 179
- Labrinidis, Alexandros, 103  
Landberg, Anders H., 85  
Lau, Ricky, 161  
Lin, Xuemin, 75  
Liu, Jun, 151  
Liu, Xiaoyan, 161  
Lodi, Stefano, 171
- Moro, Gianluca, 171
- Nguyen, Khanh, 133
- Paik, Hye-young, 141  
Pan, Qi, 179  
Pardede, Eric, 85  
Pupunwiwat, Prapassara, 19
- Rahayu, J. Wenny, 85  
Ramamohanarao, Kotagiri, 5, 37, 123  
Roantree, Mark, 151
- Sartori, Claudio, 171  
Sattar, Abdul, 113  
Scholer, Falk, 9  
Shang, Shuo, 47  
Shao, Jie, 37  
Sharaf, Mohamed A., 103  
Shen, Heng Tao, iii  
Stantic, Bela, 19, 113
- Thom, James A., 9  
Toyama, Motomichi, 29
- Wang, Hua, 93  
Wang, Huaiqing, 161  
Wang, Liping Wang, 3  
Wu, Di, 179
- Xu, Guandong, 3  
Xu, Jeffrey, 179
- Yamada, Hiroyuki, 29
- Zhang, Rui, 37, 161  
Zhang, Shaoyi, 123  
Zhang, Wenjie, 75  
Zhang, Yanchun Zhang, 3  
Zhang, Ying, 75  
Zhang, Zhe, 161  
Zheng, Kai, 47

## 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 84 - Artificial Intelligence and Data Mining 2007**

Edited by Kok-Leong Ong, Deakin University, Australia, Wenyuan Li, University of Texas at Dallas, USA and Junbin Gao, Charles Sturt University, Australia. December, 2007. 978-1-920682-65-1.

Contains the proceedings of the 2nd International Workshop on Integrating AI and Data Mining (AIDM 2007), Gold Coast, Australia. December 2007.

**Volume 85 - Advances in Ontologies 2007**

Edited by Thomas Meyer, Meraka Institute, South Africa and Abhaya Nayak, Macquarie University, Australia. December, 2007. 978-1-920682-66-8.

Contains the proceedings of the 3rd Australasian Ontology Workshop (AOW 2007), Gold Coast, Queensland, Australia.

**Volume 86 - Safety Critical Systems and Software 2007**

Edited by Tony Cant, Defence Science and Technology Organisation, Australia. December, 2007. 978-1-920682-67-5.

Contains the proceedings of the 12th Australian Conference on Safety Critical Systems and Software, August 2007, 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.

Contains the proceedings of the 7th Australasian Data Mining Conference (AusDM 2008), Adelaide, Australia. December 2008.

**Volume 88 - Koli Calling 2007**

Edited by Raymond Lister University of Technology, Sydney and Simon University of Newcastle. November, 2007. 978-1-920682-69-9.

Contains the proceedings of the 7th Baltic Sea Conference on Computing Education Research.

**Volume 89 - Australian Video**

Edited by Heng Tao Shen and Michael Frater. October, 2008. 978-1-920682-70-5.

Contains the proceedings of the 1st Australian Video Conference.

**Volume 90 - Advances in Ontologies**

Edited by Thomas Meyer, Meraka Institute, South Africa and Mehmet Orgun, Macquarie University, Australia. September, 2008. 978-1-920682-71-2.

Contains the proceedings of the Knowledge Representation Ontology Workshop (KROW 2008), Sydney, September 2008.

**Volume 91 - Computer Science 2009**

Edited by Bernard Mans Macquarie University. January, 2009. 978-1-920682-72-9.

Contains the proceedings of the Thirty-Second Australasian Computer Science Conference (ACSC2009), Wellington, New Zealand, January 2009.

**Volume 92 - Database Technologies 2009**

Edited by Xuemin Lin, University of New South Wales and Athman Bouguettaya, CSIRO. January, 2009. 978-1-920682-73-6.

Contains the proceedings of the Twentieth Australasian Database Conference (ADC2009), Wellington, New Zealand, January 2009.

**Volume 93 - User Interfaces 2009**

Edited by Paul Calder Flinders University and Gerald Weber University of Auckland. January, 2009. 978-1-920682-74-3.

Contains the proceedings of the Tenth Australasian User Interface Conference (AUIC2009), Wellington, New Zealand, January 2009.

**Volume 94 - Theory of Computing 2009**

Edited by Prabhu Manem, University of Ballarat and Rod Downey, Victoria University of Wellington. January, 2009. 978-1-920682-75-0.

Contains the proceedings of the Fifteenth Computing: The Australasian Theory Symposium (CATS2009), Wellington, New Zealand, January 2009.

**Volume 95 - Computing Education 2009**

Edited by Margaret Hamilton, RMIT University and Tony Clear, Auckland University of Technology. January, 2009. 978-1-920682-76-7.

Contains the proceedings of the Eleventh Australasian Computing Education Conference (ACE2009), Wellington, New Zealand, January 2009.

**Volume 96 - Conceptual Modelling 2009**

Edited by Markus Kirchberg, Institute for Infocomm Research, A\*STAR, Singapore and Sebastian Link, Victoria University of Wellington, New Zealand. January, 2009. 978-1-920682-77-4.

Contains the proceedings of the Fifth Asia-Pacific Conference on Conceptual Modelling (APCCM2008), Wollongong, NSW, Australia, January 2008.

**Volume 97 - Health Data and Knowledge Management 2009**

Edited by James R. Warren, University of Auckland. January, 2009. 978-1-920682-78-1.

Contains the proceedings of the Third Australasian Workshop on Health Data and Knowledge Management (HDKM 2009), Wellington, New Zealand, January 2009.

**Volume 98 - Information Security 2009**

Edited by Ljiljana Brankovic, University of Newcastle and Willy Susilo, University of Wollongong. January, 2009. 978-1-920682-79-8.

Contains the proceedings of the Australasian Information Security Conference (AISC 2009), Wellington, New Zealand, January 2009.

**Volume 99 - Grid Computing and e-Research 2009**

Edited by Paul Roe and Wayne Kelly, QUT. January, 2009. 978-1-920682-80-4.

Contains the proceedings of the Australasian Workshop on Grid Computing and e-Research (AusGrid 2009), Wellington, New Zealand, January 2009.

**Volume 100 - Safety Critical Systems and Software 2007**

Edited by Tony Cant, Defence Science and Technology Organisation, Australia. December, 2008. 978-1-920682-81-1.

Contains the proceedings of the 13th Australian Conference on Safety Critical Systems and Software, Canberra Australia.

**Volume 101 - Data Mining and Analytics 2009**

Edited by Paul J. Kennedy, University of Technology, Sydney, Kok-Leong Ong, Deakin University and Peter Christen, The Australian National University. November, 2009. 978-1-920682-82-8.

Contains the proceedings of the 8th Australasian Data Mining Conference (AusDM 2009), Melbourne Australia.