DATABASE TECHNOLOGIES 2012

Proceedings of the
Twenty-Third Australasian Database Conference
(ADC 2012), Melbourne, Australia,
31 January – 3 February 2012

Rui Zhang and Yanchun Zhang, Eds.

Volume 124 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.
### Invited Papers

Beyond Heuristics: I/O-oriented Algorithms and Structures with Performance Guarantees ............ 3  
*Yufei Tao*

Security and Privacy for Database Systems ................................................. 5  
*Hua Wang*

Discovering Associations in High-Dimensional Data ....................................... 7  
*Geoffrey I. Webb*

### Contributed Papers

On Modeling Query Refinement by Capturing User Intent through Feedback ................. 11  
*Md. Saiful Islam, Chengfei Liu and Rui Zhou*

Optimised X-HYBRIDJOIN for Near-Real-Time Data Warehousing ........................... 21  
*M. Asif Naeem, Gillian Dobbie and Gerald Weber*

Scalable Motif Detection and Aggregation ...................................................... 31  
*Jens Dietrich and Catherine McCartin*

Optimal k-Constraint Coverage Queries on Spatial Objects ............................... 41  
*Chuanfei Xu, Yanqiu Wang, Yu Gu, Shukuan Lin and Ge Yu*

A Branch and Bound Method for Min-dist Location Selection Queries ....................... 51  
*Jianzhong Qi, Zhenghua Xu, Yuan Xue and Zeyi Wen*

Energy Efficiency for MapReduce Workloads: An In-depth Study .......................... 61  
*Boliang Feng, Jiabeng Lu, Yongluan Zhou and Nan Yang*

Sentiment Classification of Blog Posts using Topical Extracts ............................ 71  
*Zhixin Zhou, Xiuzhen Zhang and Phil Vines*
Feature-based Recommendation Framework on OLAP ............................... 81
  Yang Yang and Jinli Cao

Evaluating Semantic Browsers for Consuming Linked Data ........................... 89
  Fahad Alahmari, James A. Thom, Liam Magee and Wilson Wong

Discovering Social Media Experts by Integrating Social Networks and Contents ........ 99
  Zhao Zhang, Bin Zhao, Weining Qian and Aoying Zhou

Combining Content and Quality Indicators in Ranking Ambiguous Query Results On Flickr .... 109
  Hao Zhong and Ying Zhou

Indexing RFID Data using the VG-curve ............................................. 117
  Justin Terry, Bela Stantic and Abdul Sattar

TIP Spatial Index: Efficient Access to Digital Libraries in a Context-Aware Mobile System .... 127
  Wendy Osborn and Annika Hinze

Optimized XPath Evaluation for Schema-compressed XML Data ...................... 137
  Stefan Böttcher, Rita Hartel and Stefan Heindorf

Efficient String-Based XML Stream Prefiltering ..................................... 145
  Stefan Böttcher, Rita Hartel and Steffen Weber

Author Index ................................................................. 153
Preface

It is our pleasure to present to you the Proceedings of the 23rd Australasian Database Conference (ADC 2012), taking place in Melbourne, Australia. The Australasian Database Conference series is an annual forum for sharing the latest research and novel applications of database systems. ADC 2012 is held as part of the Australasian Computer Science Week (ACSW 2012).

As in the previous two years, ADC 2012 accepted all papers that the Program Committee considered as being of ADC quality without setting any predefined quota. The conference received 30 submissions and accepted 15 papers. The Program Committee that selected the papers consisted of 45 members, who were thorough and dedicated to the reviewing process.

The ADC 2012 best paper award goes to “Optimal k–Constraint Coverage Queries on Spatial Objects” by Chuanfei Xu, Yanqiu Wang, Yu Gu, Shukuan Lin and Ge Yu. We would also like to mention two runner-ups for best paper award: “Energy Efficiency for MapReduce Workloads: An In-depth Study” by Boliang Feng, Jiaheng Lu, Yongluan Zhou and Nan Yang, and “A Branch and Bound Method for Min-dist Location Selection Queries” by Jianzhong Qi, Zhenghua Xu, Yuan Xue and Zeyi Wen. The selection is based on the reviewers scoring on the papers.

In ADC 2012, we have three invited talks given by distinguished researchers from the database community in the Australasian area: Yufei Tao, Professor from Chinese University of Hong Kong, Hua Wang, Professor from The University of Southern Queensland and Geoff I. Webb, Professor from Monash University.

In addition, an ACM SIGSPATIAL Australia Chapter special workshop is collocated with ADC 2012, which has exclusively invited talks given by experts in the area of geographical/spatial data management and analysis. This is a unique chance for learning the state-of-the-art of this area. A poster competition follows the invited talks. ADC 2012 attendees can put up posters regarding their research.

We would like to thank all our colleagues who served on the Program Committee or acted as external reviewers. We would also like to thank all the authors who submitted papers, both accepted and rejected, and the attendees. This conference is held for you, and we hope that with these proceedings, you can have an overview of the vibrant research community and its activities. We encourage you to make submissions to the next ADC conference and contribute to this community.

Rui Zhang
University of Melbourne

Yanchun Zhang
Victoria University

ADC 2012 Programme Chairs
January 2012
Programme Committee and Review Panel

Program Chairs

Rui Zhang, University of Melbourne, Australia
Yanchun Zhang, Victoria University, Australia

Programme Committee

Mohammed Eunus Ali, Bangladesh University of Engineering and Technology, Bangladesh
Jinli Cao, La Trobe University, Australia
Yuenguo Chen, Renmin University, China
Bin Cui, Peking University, China
Shane Culpepper, RMIT University, Australia
Gill Dobbie, University of Auckland, New Zealand
Zhen He, La Trobe University, Australia
Annika Hinze, University of Waikato, New Zealand
Michael Houle, National Institute of Informatics, Japan
Helen Huang, University of Queensland, Australia
Yoshiharu Ishikawa, Nagoya University, Japan
Xue Li, University of Queensland, Australia
Chengfei Liu, Swinburne University, Australia
Sebastian Maneth, NICTA, Australia
Brahim Medjahed, University of Michigan, USA
Mohamed Mokbel, University of Minnesota, USA
Miyuki Nakano, Tokyo University, Japan
Helen Paik, University of New South Wales, Australia
Chaoyi Pang, CSIRO, Australia
Weining Qian, East China Normal University, China
Uwe Roehm, University of Sydney, Australia
Shazia Sadiq, University of Queensland, Australia
Falk Scholer, RMIT University, Australia
Jie Shao, University of Melbourne, Australia
Mohamed Sharaf, University of Queensland, Australia
HengTao Shen, University of Queensland, Australia
Jialie Shen, Singapore Management University, Singapore
Bela Stantic, Griffith University, Australia
Markus Stumptner, University of South Australia, Australia
Zahir Tari, RMIT University, Australia
James Thom, RMIT University, Australia
Junhu Wang, Griffith University, Australia
Wei Wang, University of New South Wales, Australia
Linnao Xu, IBM
Ge Yu, Northeastern University, China
Qi Yu, Rochester Institute of Technology, USA
Jenny Zhang, RMIT University, Australia
Qing Zhang, CSIRO, Australia
Shichao Zhang, University of Technology, Sydney, Australia
Ying Zhang, University of New South Wales, Australia
Yu Zheng, Microsoft Research Asia
Aoying Zhou, East China Normal University, Australia
Xiangmin Zhou, CSIRO, Australia
Xuan Zhou, Renmin University, China
Organising Committee

Members

Dr. Daryl D’Souza
Assoc. Prof. James Harland (Chair)
Dr. Falk Scholer
Dr. John Thangarajah
Assoc. Prof. James Thom
Dr. Jenny Zhang
Welcome from the Organising Committee

On behalf of the Australasian Computer Science Week 2012 (ACSW2012) Organising Committee, we welcome you to this year’s event hosted by RMIT University. RMIT is a global university of technology and design and Australia’s largest tertiary institution. The University enjoys an international reputation for excellence in practical education and outcome-oriented research. RMIT is a leader in technology, design, global business, communication, global communities, health solutions and urban sustainable futures. RMIT was ranked in the top 100 universities in the world for engineering and technology in the 2011 QS World University Rankings. RMIT has three campuses in Melbourne, Australia, and two in Vietnam, and offers programs through partners in Singapore, Hong Kong, mainland China, Malaysia, India and Europe. The University’s student population of 74,000 includes 30,000 international students, of whom more than 17,000 are taught offshore (almost 6,000 at RMIT Vietnam).

We welcome delegates from a number of different countries, including Australia, New Zealand, Austria, Canada, China, the Czech Republic, Denmark, Germany, Hong Kong, Japan, Luxembourg, Malaysia, South Korea, Sweden, the United Arab Emirates, the United Kingdom, and the United States of America.

We hope you will enjoy ACSW2012, and also to experience the city of Melbourne.,

Melbourne is amongst the world’s most liveable cities for its safe and multicultural environment as well as well-developed infrastructure. Melbournes skyline is a mix of cutting-edge designs and heritage architecture. The city is famous for its restaurants, fashion boutiques, café-filled laneways, bars, art galleries, and parks.

RMIT’s city campus, the venue of ACSW2012, is right in the heart of the Melbourne CBD, and can be easily accessed by train or tram.

ACSW2012 consists of the following conferences:

– Australasian Computer Science Conference (ACSC) (Chaired by Mark Reynolds and Bruce Thomas)
– Australasian Database Conference (ADC) (Chaired by Rui Zhang and Yanchun Zhang)
– Australasian Computer Education Conference (ACE) (Chaired by Michael de Raadt and Angela Carbone)
– Australasian Information Security Conference (AISC) (Chaired by Josef Pieprzyk and Clark Thomborson)
– Australasian User Interface Conference (AUIC) (Chaired by Haifeng Shen and Ross Smith)
– Computing: Australasian Theory Symposium (CATS) (Chaired by Julián Mestre)
– Australasian Symposium on Parallel and Distributed Computing (AusPDC) (Chaired by Jinjun Chen and Rajiv Ranjan)
– Australasian Workshop on Health Informatics and Knowledge Management (HIKM) (Chaired by Kerryn Butler-Henderson and Kathleen Gray)
– Asia-Pacific Conference on Conceptual Modelling (APCCM) (Chaired by Aditya Ghose and Flavio Ferrarotti)
– Australasian Computing Doctoral Consortium (ACDC) (Chaired by Falk Scholer and Helen Ashman)

ACSW is an event that requires a great deal of co-operation from a number of people, and this year has been no exception. We thank all who have worked for the success of ACSE 2012, including the Organising Committee, the Conference Chairs and Programme Committees, the RMIT School of Computer Science and IT, the RMIT Events Office, our sponsors, our keynote and invited speakers, and the attendees.

Special thanks go to Alex Potanin, the CORE Conference Coordinator, for his extensive expertise, knowledge and encouragement, and to organisers of previous ACSW meetings, who have provided us with a great deal of information and advice. We hope that ACSW2012 will be as successful as its predecessors.

Assoc. Prof. James Harland
School of Computer Science and Information Technology, RMIT University
ACSW2012 Chair
January, 2012
CORE welcomes all delegates to ACSW2012 in Melbourne. CORE, the peak body representing academic computer science in Australia and New Zealand, is responsible for the annual ACSW series of meetings, which are a unique opportunity for our community to network and to discuss research and topics of mutual interest. The original component conferences - ACSC, ADC, and CATS, which formed the basis of ACSW in the mid 1990s - now share this week with seven other events - ACE, AISC, AUIC, AusPDC, HIKM, ACDC, and APCCM, which build on the diversity of the Australasian computing community.

In 2012, we have again chosen to feature a small number of keynote speakers from across the discipline: Michael Kölling (ACE), Timo Ropinski (ACSC), and Manish Parashar (AusPDC). I thank them for their contributions to ACSW2012. I also thank invited speakers in some of the individual conferences, and the two CORE award winners Warwish Irwin (CORE Teaching Award) and Daniel Frampton (CORE PhD Award). The efforts of the conference chairs and their program committees have led to strong programs in all the conferences, thanks very much for all your efforts. Thanks are particularly due to James Harland and his colleagues for organising what promises to be a strong event.

The past year has been very turbulent for our disciplines. We tried to convince the ARC that refereed conference publications should be included in ERA2012 in evaluations – it was partially successful. We ran a small pilot which demonstrated that conference citations behave similarly to but not exactly the same as journal citations - so the latter can not be scaled to estimate the former. So they moved all of Field of Research Code 08 “Information and Computing Sciences” to peer review for ERA2012. The effect of this will be that most Universities will be evaluated at least at the two digit 08 level, as refereed conference papers count towards the 50 threshold for evaluation. CORE’s position is to return 08 to a citation measured discipline as soon as possible.

ACSW will feature a joint CORE and ACDICT discussion on Research Challenges in ICT, which I hope will identify a national research agenda as well as priority application areas to which our disciplines can contribute, and perhaps opportunity to find international multi-disciplinary successes which could work in our region.

Beyond research issues, in 2012 CORE will also need to focus on education issues, including in Schools. The likelihood that the future will have less computers is small, yet where are the numbers of students we need?

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 2011; in particular, I thank Alex Potanin, Alan Fekete, Aditya Ghose, Justin Zobel, and those of you who contribute to the discussions on the CORE mailing lists. There are three main lists: csprofs, cshods and members. You are all eligible for the members list if your department is a member. Please do sign up via http://lists.core.edu.au/mailman/listinfo - we try to keep the volume low but relevance high in the mailing lists.

Tom Gedeon
President, CORE
January, 2012
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.

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

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

2011. Volume 33. Host and Venue - Curtin University of Technology, Perth, WA.
2010. Volume 32. Host and Venue - Queensland University of Technology, Brisbane, QLD.
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.
1998. Volume 20. Hosts - University of Western Australia, Murdoch University, Edith Cowan University and Curtin University. Venue - Perth, WA.
1995. Volume 17. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Glenelg, SA.
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.
1987. Volume 9. Host and Venue - Deakin University, VIC.
1986. Volume 8. Host and Venue - Australian National University, Canberra, ACT.
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

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

Note that various name changes have occurred, which have been indicated in the Conference Acronyms sections in respective CRPIT volumes.
We wish to thank the following sponsors for their contribution towards this conference.

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

RMIT University,
www.rmit.edu.au/

Australian Computer Society,
www.acs.org.au

University of Melbourne,
www.unimelb.edu.au
INVITED PAPERS
Beyond Heuristics: I/O-oriented Algorithms and Structures with Performance Guarantees

Yufei Tao

1 Department of Computer Science and Engineering
Chinese University of Hong Kong
New Territories, Hong Kong
Email: taoyf@cse.cuhk.edu.hk

2 Division of Web Science
Korea Advanced Institute of Science and Technology
Daejeon, Korea
Email: taoyf@kaist.ac.kr

Abstract

System development often results in algorithms and access methods that are heuristic in nature. They have the merits of being easy to understand, simple to implement, and reasonably effective on many real datasets. Nonetheless, common criticism is that they typically come with no non-trivial performance guarantees. For years the database community has been striving to discover methods that are small and sweet. That is, such a method should be implementable in practice (i.e., small), and in the mean time, must retain attractive efficiency even in the worst case (i.e., sweet). This talk serves as an introduction to the current progress in the research of small and sweet algorithms and data structures. We will review solutions to several classic problems including spatial join, skyline retrieval, range searching, and so on. Special discussion will be dedicated to techniques generally useful for obtaining good worst-case I/O bounds.
Abstract
Database security is a discipline that seeks methods to protect data stored at DBMSs from intrusions, improper modifications, theft, and unauthorized disclosure of private information. This is realized through a set of security services, which meet the security requirements of both the system and the data sources. A number of different techniques and approaches has been developed to assure data confidentiality, integrity, and availability in DBMSs, however, despite such advances, the database security area faces several new challenges. This talk first presents the most relevant concepts underlying the notion of database security and summarizes the most well-known techniques such as key access control models, namely, the discretionary and mandatory access control models, and the role-based access control model. It then analyses optimal privacy-aware path approaches in Hippocratic Databases and a pairwise-systematic microaggregation for statistical disclosure control; additionally theoretical analysis of limiting disclosure of private information in relational database systems is discussed.

Keywords: Database, Security, Privacy, Access control.
Discovering Associations in High-Dimensional Data

Geoffrey I. Webb

Faculty of Information Technology
Monash University
Clayton, Vic, 3800, Australia
Email: Geoff.Webb@monash.edu

Abstract

Association discovery is one of the most studied tasks in the field of data mining (Agrawal et al. 1993). It involves identifying items that occur together in data, and has numerous applications in manufacturing, commerce, administration and science. However, far more attention has been paid to how to discover associations than to what associations should be discovered. In this talk Geoff will provide a highly subjective tour of the field. He will

• highlight shortcomings of the dominant frequent pattern paradigm;

• illustrate benefits of the alternative top-k paradigm; and


References


URL: http://dx.doi.org/10.1002/widm.28


Copyright ©2012, Australian Computer Society, Inc. This paper appeared at the 23rd Australasian Database Conference (ADC 2012), Melbourne, Australia, January-February 2012. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 124, Rui Zhang and Yanchun Zhang, Ed. Reproduction for academic, not-for-profit purposes permitted provided this text is included.
Contribution Papers
On Modeling Query Refinement by Capturing User Intent through Feedback

Md. Saiful Islam, Chengfei Liu and Rui Zhou
Faculty of Information and Communication Technologies
Swinburne University of Technology
VIC 3122, Australia
{mdsaifulislam,cliu,rzhou}@swin.edu.au

Abstract
SQL queries in relational data model implement the binary satisfaction of tuples. Tuples are generally filtered out from the result set if they miss the constraints expressed in the predicates of the given query. For naïve or inexperienced users posing precise queries in the first place is very difficult as they lack of knowledge of the underlying dataset. Therefore, imprecise queries are commonplace for them. In connection with it, users are interested to have explanation of the missing answers. Even for unexpected tuples present in the result set advanced users may also want to know why a particular piece of information is present in the result set. This paper presents a simple model for generating explanations for both unexpected and missing answers. Further, we show how these explanations can be used to capture the user intent via feedback specifically for refining initial imprecise queries. The presented framework can also be thought as a natural extension for the existing SQL queries where support of explanation of expected and unexpected results are required to enhance the usability of relational database management systems. Finally, we summarize future research directions and challenges that need to be addressed in this endeavour.

Keywords: Imprecise Query, Explanation, Point Domination Theory, User Feedback, Query Refinement.

1 Introduction
Relational Data Model is one of the widely used data model for storing and retrieving information. The underlying query engine accepts request from users through SQL by which the non-answers are filtered out from the answers. To filter the non-answers users express their constraints in the form of predicates. Predicates are generally grouped in conjunctive or disjunctive normal form or a combination of them. For naïve or inexperienced users, posing appropriate constraints in the predicates without having a complete knowledge of the underlying dataset is a tedious job. In most of the cases, users go for a number of trials before getting the ultimate or precise one. In the worst case, they reach to an unsatisfactory one. A system that can generate explanations for the unexpected and the expected information is advantageous in this case as the user can directly concentrate on the appropriate causes to alleviate their problems. Particularly, this could give them an opportunity to know why a particular piece of information is present in the result set and why the result set misses something that is expected, i.e., why not. At present, relational database systems does not support these kinds of explorative data analysis facilities to answer why and why not questions.

In cooperative database system a dialog is established between the user and the system to understand the intent of user (Nandi and Jagadish 2007, Sultana et al. 2009) by returning more information to the user in response to a query than just the query's answer set itself to help users in this regard (Motro 1994). Recently, explanation of query results and their provenance information such as why and how a particular piece of information arrived (Green et al. 2007, Cheney et al. 2009 and Glavic et al. 2009) and even missed (Huang et al. 2008, Herschel et al. 2010, Chapman et al. 2009 and Tran et al. 2010) are suggested to gain the confidence of users in this endeavour.

User feedback is extensively studied in information retrieval areas specifically for searching similar pages in the web and is generally considered as the basis for improving the sensitivity of the system (Moon et al. 2010). Feedback based query refinement is a newer concept in relational data model. Some researchers proposed keyword search in relational databases in this direction where feedback is an important step in ranking the resultant tuples by some non-linear ranking functions (Baid et al. 2010). In our model, feedback are collected in the form of what part of information in the result set a user doesn’t want and what part of non-answers the user expects to see. Taking into consideration this feedback we then try to capture the user intents. This kind of cooperative behavior of DB is discussed by Amer-Yahia et al. (2005) to bridge the gap between DB and IR techniques to increase the usability of databases (Jagadish et al. 2007). In our model, explanations of unexpected and expected information are analysed to refine the initial query. This kind of query refinement approach can greatly minimize the distances between the original and the refined queries.

Our main contributions are as follows:
- In our model, we provide a unified framework in which we can explain why result set includes something we don’t expect and why it misses something we expect. That is, our model is equally applicable to explain unexpected as well as missing answers.
- We show how the explanations can be used to capture user intent and exploited by our query refinement model. In other words, we show how incomplete user
feedback can be automatically made complete by our model.

- Finally, we also show how explanation based query refinement could be used as a helping hand to support imprecise queries in relational data model.

The remainder of the paper is organized as follows: Section 2 provides the preliminaries, Section 3 describes our explanation model, Section 4 presents how we can capture user intent, Section 5 describes the explanation based query refinement and future challenges, Section 6 presents related work and finally, Section 7 concludes the paper.

2 Preliminaries

This section describes briefly about the types of SQL queries that are targeted in this paper, their forms and transformations, taxonomy of database tuples with respect to the submitted query, provenance query and attributes, and finally presents the query refinement problem.

2.1 Types of SQL Queries

In our model, we consider simple queries in which each projected attribute is relation’s attribute and the selection condition is a conjunction of predicates or disjunction of predicates or an arbitrary combination of them. For simplicity and without any loss of generality, we assume that the user queries can be mapped to either Disjunctive Query (DQ) or Conjunctive Query (CQ). The definitions of these queries are given below:

Definition 1 (Disjunctive Query): A disjunctive query (DQ) is a query (Q), where the conditions are specified in disjunctive normal form as follows:

\[ DQ = C_1 \lor C_2 \lor \ldots \lor C_n \]

and

\[ C_i = C_{i1} \land C_{i2} \land \ldots \land C_{ik} \]

Definition 2 (Conjunctive Query): A conjunctive query (CQ) is a query (Q), where the conditions are specified in conjunctive normal form as follows:

\[ CQ = C_1 \land C_2 \land \ldots \land C_n \land C_1 = C_{i1} \lor C_{i2} \lor \ldots \lor C_{ik} \]

where \( C_i \)'s are selection or join predicates in both DQ and CQ.

Example 1: Consider a database consisting of two tables: Publication (PubID, Pages, Pubyear, Book, CitationCnt) and Book (ERAID, Acronym, Rank, Areanum, Area).

Now, a disjunctive SQL query (DQ) is given in Fig. 1 to retrieve publications from the above mentioned database.

SELECT PubID
FROM publication, book
WHERE (PubYear => 1990 AND CitationCnt => 80 AND Rank <= 2) OR (CitationCnt => 60 AND Book = 'vldb') AND (PubYear => 1990 OR Book = 'vldb') AND (Rank => 80 OR Book = 'vldb') AND (Book = acronym);

Figure 1: An example of disjunctive SQL query (DQ)

2.2 Conversion between CQ and DQ

The conversion between CNF and DNF is a well-known problem in the history of computational science. The prospect of our explanation model depends on the successful conversion between CQ and DQ. We assume that there would be no missing data in our database, at least virtually. For example, if there are missing values for which RDM maintains null, we assume comparing (<, \[=\), \[\leq\], \[\geq\], \[\neq\] ) anything with null– even another null values – results in false true state. This assumption simplifies SQL’s three-valued logic to two-valued logic. Therefore, we can treat our query Q as a Boolean expression. That is, evaluation of a tuple with respect to Q will be true or false. Hence, any DQ can be converted to the corresponding CQ by the distributive law from elementary algebra (distributing \( \lor \) over \( \land \)) and vice versa, e.g., CQ of DQ (Fig. 1) is given Fig.2. The equivalent CQ (Fig. 2) has seven disjunctions whereas the original DQ had two conjunctions. This suggests that the resulting CQ query could be large compared to the size of original DQ query. But in practice users are willing to issue fewer predicates in their queries. Therefore, we assume we can handle the complexity of the converted CQ and DQ in reasonable time. This is further motivated by users’ interest on the explanation of expected and unexpected tuples that are very few.

SELECT PubID
FROM Publication, Book
WHERE (CitationCnt >= 60) AND (Book = Acronym);

Figure 2: Equivalent CQ of the given DQ (Fig.1) obtained by applying the distributed law.

2.3 Taxonomy of Database Tuples

We treat user query (Q) in our model as a binary classifier by which the entire database tuples will be divided into two broad categories: Resultant Tuples (RT) and Non-Resultant Tuples (NRT). If submitted query is the imprecise or vague one (predicates are not specified correctly), the resultant tuples can further be partitioned into two groups: True Positives (TP) and False Positives (FP) or unexpected tuples. Similarly, non-resultant tuples are of two types: True Negatives (TN) and False Negatives (FN) or expected tuples or missing tuples. The taxonomy of database tuples regarding the user’s imprecise query is given below:

(a). Resultant Tuple (RT). A resultant tuple is a tuple in the result set for which at least one Ci in DQ evaluates to true. In other words, an RT tuple satisfies at least one Ci in DQ.

(i). Truly-positive Tuple (TP): A truly-positive tuple is a resultant tuple that satisfies at least one Ci in DQ with desirability ‘yes’ from user’s point of view.

(ii). Unexpected Tuple (FP): An unexpected tuple is a resultant tuple that satisfies at least one Ci in DQ with desirability ‘no’ from user’s point of view.

(b). Non-resultant Tuple (NRT): A non-resultant tuple is a tuple for which at least one Ci in CQ evaluates to false. In other words, an NRT tuple dissatisfies at least one Ci in CQ.

(i). Expected Tuple (FN): An expected tuple is a non-resultant tuple that fails to satisfy at least one Ci in CQ with desirability ‘yes’ from user’s point of view.
Fig. 1. We apply the eager approach (Tan, 2004) for the equivalent provenance query of the query given in Fig. 1. Query rewriting is largely based on the query rewrite rules creation of output tuple is propagated. Our provenance additional provenance attributes through which computing the provenance information of our resultant tuples. That is, provenance information is computed along with our query outputs. Table 1 shows the resultant tuples together with their corresponding provenance information.

In our model, we exploit provenance information to find the implicit predicates, eliminate the need of accessing the database again, classify the tuples and foster the decision process for queries refinement.

SELECT PubID, PubYear, CitationCnt, Rank FROM publication, book WHERE (PubYear >= 1990 AND CitationCnt >= 80 AND Rank <= 2 AND Book = Acronym) OR (CitationCnt >= 60 AND Book = ‘vldb’ AND Book = Acronym);

Figure 4: Equivalent provenance query $Q^*$ for the query given in Fig. 1

2.5 Query Refinement Problem

We refer to queries which require the conditions to be slightly adjusted as imprecise queries while those requiring no adjustment to as precise queries. Supporting imprecise queries over databases necessitates a system that collects feedback from the user to understand the submitted query intent and returns a new query to encounter both unexpected and expected tuples. Therefore, we define the query refinement problem as follows:

Given $Q$, $R^u$ and $R^e_N$ find a refined query $Q^*$ such that,

$$Q^* \equiv R^P \cup R^u \land R^e_N \neq R^e_N \cup R^u$$

But in reality, the distance between the optimum refined query $Q^*$ and the initial query $Q$ could be large. Therefore, our target is to find an approximate refined query $Q'$ which will minimize the number unexpected tuples and maximize the number of expected tuples as much as possible. But the approximate query $Q'$ will be highly similar to the initial query $Q$. That is,

$$Q' \equiv R^P \cup R^u \land R^e_N \neq R^e_N \cup R^u$$

and $sim(Q, Q') \gg sim(Q, Q^*)$

where $R^P \subseteq R^P$, $R^e_N \subseteq R^e_N$, $R^u \subseteq R^u$, $R^e_N \subseteq R^e_N$ and $sim(Q, Q^*)$ is the similarity score between $Q$ and $Q^*$. The similarity calculation of the refined query to the original query is given in section 5.

In refining initial query, we give emphasis to both the quality of query results as well as similarity metric. That is, we model refined query as the one that can return better quality results and at the same time will also be highly similar to the initial query.

Table 1: Set of resultant (RT) tuples of the query given in Fig. 4. The shaded region represents the provenance information

<table>
<thead>
<tr>
<th>PubID</th>
<th>PubYear</th>
<th>CitationCnt</th>
<th>Rank</th>
<th>Book</th>
<th>Acronym</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1993</td>
<td>90</td>
<td>1</td>
<td>sigmod</td>
<td>sigmod</td>
</tr>
<tr>
<td>P2</td>
<td>1990</td>
<td>148</td>
<td>1</td>
<td>sigmod</td>
<td>sigmod</td>
</tr>
<tr>
<td>P3</td>
<td>1996</td>
<td>81</td>
<td>1</td>
<td>sigmod</td>
<td>sigmod</td>
</tr>
<tr>
<td>P4</td>
<td>1994</td>
<td>82</td>
<td>1</td>
<td>vldb</td>
<td>vldb</td>
</tr>
<tr>
<td>P5</td>
<td>1983</td>
<td>72</td>
<td>1</td>
<td>vldb</td>
<td>vldb</td>
</tr>
<tr>
<td>P6</td>
<td>1980</td>
<td>66</td>
<td>1</td>
<td>vldb</td>
<td>vldb</td>
</tr>
<tr>
<td>P7</td>
<td>1987</td>
<td>117</td>
<td>1</td>
<td>vldb</td>
<td>vldb</td>
</tr>
</tbody>
</table>

Figure 3: Taxonomy of DB tuples wrt to user query

2.4 Provenance Query and Attributes

Provenance describes the lineage or pedigree of data and finds paramount of importance in scientific databases, data warehouses, and workflow management systems. Why provenance gives information about the source tuples that have direct contribution to the resultant tuples and where provenance tells about the location of source tuple values from where the output tuple values are copied from (Tan 2004, Cheney et al. 2009). Provenance Query ($Q'$) produces the same result as like as $Q$ but adds additional provenance attributes through which information about the input tuples that contributed to the creation of output tuple is propagated. Our provenance query is largely based on the query rewrite rules developed by Glavic and Alonso (2009). Fig. 4 shows the equivalent provenance query of the query given in Fig. 1. We apply the eager approach (Tan, 2004) for computing the provenance information of our resultant tuples.
3 Explanation Model

In our explanation model, we generate appropriate explanations for why the result set includes some information that is unexpected as well as why it misses something that is expected. For these, we consider the user query divides the entire DB into resultant (RT) and non-resultant (NRT) tuples (as given in Fig. 3). But because of user's imprecise specification of constraints in the predicates of the query RT also includes some unexpected tuples that we wish to exclude. Similarly, RT may also miss some expected tuples that we want to get back in the result set. In our explanation model we use clauses in DQ and CQ to explain expected and unexpected tuples, respectively. The advantage of using clauses to explain expected and unexpected tuples is that the generated explanations are exact and complete. Therefore, we know which causes we need to alleviate to encounter them in the refined queries.

3.1 Explanation of Unexpected Tuple

To model why a particular tuple (including the unexpected one) presents in the result set we rely on the Ci in DQ that are evaluated to true. This is because in DQ a Ci will be evaluated to true if each individual predicate Cij in Ci is satisfied. Therefore, we can say that our explanations are exact for unexpected tuples. The following defines the explanation of unexpected tuple (t’) more formally:

**Definition 3:** The explanation of an unexpected tuple, t’, consists of those conjunctions Ci in DQ that are evaluated to true. That is,

\[ \text{Expl}(t') = \{ C_i | C_i \in DQ \land t' \} \]

We assume for unexpected tuple, only few Ci in DQ will be evaluated to true. This is because, unexpected tuples lies on the boundary of user defined predicates. Therefore, to refine the initial query we can concentrate only on the explanation of the unexpected tuple, i.e., \( C_i \in \text{Expl}(t') \). It should be noted that to exclude an unexpected tuple t’ from the result set we need to modify at least one Cij in Ci appearing in Expl(t’) so that Ci \( \not\in DQ \land t' \).

**Example 2:** Consider the resultant tuples given in Table 1 and the CQ given in Fig 1. Now, user wants to know why result set misses tuple: (’P8’, 78, 1986, 1, ‘sigmod’, ‘vldb’) and seeks explanation for it. If we analyse tuple P8 we can see that it dissatisfies the fourth and fifth disjunctions of CQ. That is, the explanation of why P8 is not in the result set consists of the fourth and fifth conjunctions. Hence, we can say:

\[
P8 \text{ is not in the result set as (CitationCnt (78) < 80 or Book (’sigmod’) != ’vldb’) and (PubYear (1986) <1990 or Book (’sigmod’) != ’vldb’).}
\]

Now, if we would like to include it in the result set then we can set (CitationCnt>=78 and PubYear>=1986) in the first conjunction or Book in (’vlub, ‘sigmod’) in the second conjunction of DQ.

4 Capturing User Intent

The primary goal of any information retrieval (IR) system is to retrieve documents, which are relevant to the user query while retrieving as few non-relevant as possible. In IR, user feedback is one of the extensively studied and widely accepted techniques to refine the initial result set (Mishra and Koudas 2009, Bellahjame et al. 2011). Feedback is generally gathered to improve the precision and sensitivity of the retrieval system. We bring the idea of IR system into RDM to capture the user intent and refine the initial imprecise SQL query.

In our model, we propose the system to present the result set of the initial imprecise query to the user for her judgement. Then, she can select which tuples she considers unexpected as well as what other tuples from non-answers she expects to see in the result set. This could also complicate the collection of feedback as the user may feel reluctant to provide the complete set of unexpected and expected tuples. Therefore, we want our users to input only a subset of expected and unexpected tuples. We consider these tuples as the hints for the system. In our model, we then complement the feedback through point domination theory.

4.1 Point Domination Theory

Let \( G = \{G_1, G_2, ..., G_n\} \) be a set of the predicate preferences. We denote by \( t \geq G_i t’ \) and \( t \geq G_i t’ \) the statements ‘tuple t satisfies preference \( G_i \) better than t’ and ‘tuple t satisfies preference \( G_i \) as least as t’ respectively. If we plot the preference predicates \( G \) in an n-dimensional space, each tuple will be an n-dimensional point. Then we say a point t is as good as another point t’
\[ \forall i \in [1, n], t \succneq_{q_i} t', \text{ and point } t \text{ dominates point } t' \iff \forall i \in [1, n], t \succneq_{q_i} t' \text{ and } \exists k \in [1, n], t \succneq_{q_k} t' \text{ (Pareto-order: Voorneveld 2003).} \]

We denote by \( t \succneq t' \) and \( t' \succneq t \) the statements “tuple \( t \) is as good as tuple \( t' \)” and “tuple \( t \) dominates tuple \( t' \)” respectively.

To make the above clear, consider there are two predicates we wish to consider and place these two predicates in \( X \) and \( Y \) directions in a two-dimensional space. Suppose there are five points ‘a’, ‘b’, ‘p’, ‘c’ and ‘d’ in this space as shown in Fig. 5. Now, both ‘a’ and ‘b’ dominates ‘p’ as both ‘a’ and ‘b’ is as good as ‘p’ and satisfies at least one predicate better than ‘p’. Similarly, ‘p’ dominates both ‘c’ and ‘d’. If ‘p’ is our reference point, then ‘p’ divides the entire space into four regions as shown in Fig. 5. Now in general, any point from region \( R_1 \) is as good as or dominates ‘p’. Similarly, ‘p’ is as good as or dominates any point in region \( R_3 \).

**Figure 5: Point domination theory for discovering implicit ‘yes’ and ‘no’ tuples**

**Definition 5 (Explicit ‘yes’):** Tuples given explicitly by the user as expected are called explicit ‘yes’ tuples. Explicit ‘yes’ tuples are part of the non-answers (NRT tuples).

**Definition 6 (Explicit ‘no’):** Tuples given explicitly by the user as unexpected are called explicit ‘no’ tuples. Explicit ‘no’ tuples are part of the answers (RT tuples).

**Definition 7 (Implicit ‘yes’):** Any tuple from non-answers (NRT tuples) that is not dominated by explicit ‘yes’ tuples is called implicit ‘yes’ tuple. That is, implicit ‘yes’ tuples are as good as explicit ‘yes’ tuples.

**Definition 8 (Implicit ‘no’):** Any tuple from answers (RT tuples) that is dominated by explicit ‘no’ tuples is called implicit ‘no’ tuple. That is, implicit ‘no’ tuples are no better than explicit ‘no’ tuples.

**Example 4 (Implicit ‘no’):** Suppose that the user gives the following tuple as the unexpected tuple: (‘P5’, 1983, 72, 1, ‘vldb’, ‘vldb’).

The explanation of the above tuple is: CitationCnt (72) \( \geq 60 \) AND Book (‘vldb’) = ‘vldb’. Then according to the point domination theory, the following tuple is also unexpected as the above tuple dominates the following tuple in terms of both CitationCnt and Book attribute. Therefore, the following tuple is implicit ‘no’ tuple.


**Example 5 (Implicit ‘yes’):** Consider the user inputs the following tuple as the expected tuple: (‘P8’, 1986, 78, 1, ‘vldb’, ‘vldb’).

Now if we analyse the explanation of the above tuple we see that the above tuple is dominated by the following two tuples in terms of both PubYear and CitationCnt attributes. Then, according to the point domination theory, the following two tuples are implicit ‘yes’ tuples as they are expected too.


**Definition 9 (Explicit Predicates):** The predicates that are given explicitly by the user in the initial query are called explicit predicates.

**Example 6:** The explicit predicates for our example query (Fig. 1) are:

(i). \( \text{PubYear} \geq 1990 \)  
(ii). \( \text{CitationCnt} \geq 60 \)  
(iii). \( \text{Rank} \leq 2 \)  
(iv). Book = Acronym

The predicates on the left side are the explicit predicates for the first conjunction and the predicates on the right side are the explicit predicates for second conjunction of the DQ given in Fig. 1.

**Definition 10 (Implicit Predicates):** Implicit predicates are the predicates that are inferred from the (RT) resultant tuples and explicit feedback.

**Example 7:** The explanation of the first four tuples (as shown in Table 2) consists of the first conjunctions of DQ of Fig.1. That is, these four tuples satisfy the explicit predicates of the first conjunction. Assume user inputs the fourth tuple as the unexpected one. We can easily observe that for all tuples including the explicit ‘yes’: Book= ‘sigmod’ and Rank \( \leq 1 \). Hence, we infer the following predicates as the implicit predicates for the first conjunction.

(i). Book = ‘sigmod’  
(ii). Rank \( \leq 1 \)

### 4.2 Reduction of Implicit Feedback

After getting feedback from the user (both unexpected and expected tuples) we find the implicit ‘yes’ tuples. But these implicit ‘yes’ could be far from user’s expectation. Sometimes this implicit feedback could be difficult to encounter by the system and system response could be overwhelming for the users too. To alleviate this problem, we reduce them by evaluating the implicit predicates. That is, if the implicit ‘yes’ tuples fail to satisfy implicit predicates we delete them from the feedback list as shown in Fig. 6.

**Example 8:** Implicit ‘yes’ tuples (‘P10’, 1986, 79, 1, ‘sigmod’, ‘sigmod’) satisfies the implicit predicates for the first conjunction of DQ. So, we accept this tuple as user implicit feedback. But tuple (‘P9’, 1986, 1, ‘oopsla’, ‘oopsla’) is rejected as it fails to satisfy the implicit predicates.

As we see from example 5 and 8 that the reduction rate is quite promising (50% in this case). This suggests that implicit feedback should be treated carefully and must be corrected by analysing the implicit predicates. Implicit predicates could be advantageous in the case where users are willing to provide only the values in few dimensions (e.g., virtual tuples) rather than a complete set of tuples.
Table 2: Set of resultant (RT) tuples of the user query, Q. The last columns are the user feedback. Desirability with ‘no’ are the unexpected tuples.

<table>
<thead>
<tr>
<th>PubID</th>
<th>PubYear</th>
<th>CitationCnt</th>
<th>Rank</th>
<th>Book</th>
<th>Acronym</th>
<th>Desirability</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>1993</td>
<td>90</td>
<td>1</td>
<td>sigmod</td>
<td>Sigmod</td>
<td>yes</td>
</tr>
<tr>
<td>P2</td>
<td>1990</td>
<td>148</td>
<td>1</td>
<td>sigmod</td>
<td>Sigmod</td>
<td>yes</td>
</tr>
<tr>
<td>P3</td>
<td>1996</td>
<td>81</td>
<td>1</td>
<td>sigmod</td>
<td>Sigmod</td>
<td>yes</td>
</tr>
<tr>
<td>P4</td>
<td>1994</td>
<td>82</td>
<td>1</td>
<td>vldb</td>
<td>vldb</td>
<td>no</td>
</tr>
<tr>
<td>P5</td>
<td>1983</td>
<td>72</td>
<td>1</td>
<td>vldb</td>
<td>vldb</td>
<td>no</td>
</tr>
<tr>
<td>P6</td>
<td>1980</td>
<td>66</td>
<td>1</td>
<td>vldb</td>
<td>vldb</td>
<td>no</td>
</tr>
<tr>
<td>P7</td>
<td>1987</td>
<td>117</td>
<td>1</td>
<td>vldb</td>
<td>vldb</td>
<td>yes</td>
</tr>
</tbody>
</table>

Figure 6: Reduction of implicit ‘yes’ tuples

5 Query Refinement Model

One particular problem related to the imprecise query is how to refine it to encounter both unexpected and expected tuples. Our purpose is to exclude the unexpected as well as include the expected tuples as much as possible after getting feedback from the user. In our model, we don’t require our user to provide the complete set of unexpected and expected tuples. Only a subset of them can be sufficient. These tuples are then analysed to complement the feedback as we explain in section 4.

We consider One Clause At a Time (OCAT) in the given query and propose structural changes to minimize the number of unexpected and maximize the number of expected tuples. In our model, we assume that the following assumptions hold regarding the initial imprecise query:
- Users are more confident and less flexible about the equality (‘=’) operator.
- Users are more flexible about the inequality (>, ≥, <, ≤) operators.

5.1 Exclusion of Unexpected Tuples

Unexpected tuples are part of the resultant (RT) tuples. Though these tuples satisfy the user defined initial imprecise constraints, they generally stretch out near the boundaries of the predicates. In our query refinement model, to exclude these unexpected tuples from the result set we propose OCAT approach. In this technique, we group all resultant tuples based on their satisfying conjunctions (explanations). Then, we propose structural changes for each conjunction if they include any unexpected tuples. Otherwise, we leave them as they were before.

For each group D of resultant tuples, we tighten some predicates that appear in the conjunction to separate the unexpected tuples from the truly positive tuples. We rely on the information gain theory and Decision Tree (Mitchell 1997) to find the best separating attributes. To do so, we calculate the information gain of each attribute (A) that appears in the predicates. The formulas for calculating the information gain are given below:

\[ Info(D) = - \sum_{i} p_i \log p_i \]

where \( Info(D) \) is the expected information needed to classify a tuple in \( D \) and \( p_i \) is the probability that an arbitrary tuple in \( D \) belongs to class \( C_i \), estimated by \( |C_i,D|/|D| \). In our model, there are two categories of resultant tuples: \( C_1 \) (desirability ‘yes’) and \( C_2 \) (desirability ‘no’). Tuples with desirability ‘yes’ are the truly positive tuples and tuples with desirability ‘no’ are the unexpected tuples as given in Table 2. Information needed (after using \( A \) to split \( D \) into \( v \) partitions) to classify \( D \) is:

\[ Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j) \]

Information gained by branching on attribute \( A \) is:

\[ Gain(A) = Info(D) - Info_A(D) \]

We use the best-known and most widely-used C4.5 decision tree learner (Mitchell 1997) and WEKA implementation of it (Hall et al. 2009). All paths \( P \) from root to a leaf (‘yes’) node of the tree forms the CNF expressions that are ORed together to form a DNF expression. The DNF expression is then ANDed to the original query and finally simplified to get the refined query.

Table 3: Tuples identified by the second conjunction of DQ (Fig. 1)

<table>
<thead>
<tr>
<th>PubID</th>
<th>PubYear</th>
<th>CitationCnt</th>
<th>Desirability</th>
</tr>
</thead>
<tbody>
<tr>
<td>P4</td>
<td>1994</td>
<td>82</td>
<td>no</td>
</tr>
<tr>
<td>P5</td>
<td>1983</td>
<td>72</td>
<td>no</td>
</tr>
<tr>
<td>P6</td>
<td>1980</td>
<td>66</td>
<td>no</td>
</tr>
<tr>
<td>P7</td>
<td>1987</td>
<td>117</td>
<td>yes</td>
</tr>
</tbody>
</table>

Example 9: Consider the DQ given in Fig. 1 and the resultant tuples given in Table 2. The last four tuples (shown in Table 3) are identified by the second conjunction in DQ. The separation of the unexpected tuples (P4, P5 and P6) from truly positive tuples (P7) is done by the best separating attribute CitationCnt (among Pubyear and CitationCnt) as shown in Fig. 7.

Therefore, we propose the following structural changes for the second conjunction: CitationCnt ≥ 82 AND
Book = ‘vldb’ AND Book = Acronym and we have the following refined DQ to exclude the unexpected tuples in the result set:

\[
\text{SELECT PubID FROM publication, book WHERE (PubYear >= 1990 AND CitationCnt >= 80 AND Rank <= 2 AND Book = Acronym) OR (CitationCnt >= 82 AND Book = 'vldb' AND Book = Acronym);
}\]

**Figure 7: Separation of unexpected tuples from truly positive tuples**

To include these tuples in the result set, we can make the following changes in the first conjunction in DQ:

- CitationCnt >= 80 to CitationCnt >= 82 in the first conjunction.
- CitationCnt >= 80 to CitationCnt >= 78 in the first conjunction.

**Example 10:** Consider the CQ given in Fig. 2 and expected feedback {('P8', 1986, 78, 1, 'sigmod', 'sigmod'), ('P9', 1986, 79, 1, 'oopsla', 'oopsla'), ('P10', 1986, 79, 1, 'sigmod', 'sigmod')}. All these tuples fail to satisfy the following disjunctions:

(i). CitationCnt >= 80 or Book = 'vldb'
(ii). PubYear >= 1990 or Book = 'vldb'

To include these tuples in the result set, we can make following changes in the first conjunction in DQ:

(i). CitationCnt >= 78 instead of CitationCnt >= 80
(ii). PubYear >= 1986 instead of PubYear >= 1990

For the second conjunction in DQ, we can make the following change:

(i). Book in ('sigmod', 'oopsla') instead of Book = 'vldb'

So we have two options to encounter these expected tuples. As per our assumptions (assumption 2), we have the following refined DQ to include the expected tuples in the result set:

\[
\text{SELECT PubID FROM publication, book WHERE (PubYear >= 1986 AND CitationCnt >= 78 AND Rank <= 2 AND Book = Acronym) OR (CitationCnt >= 82 AND Book = 'vldb' AND Book = Acronym);
}\]

**Figure 8: Refined query of DQ given in Fig. 1**

The query given in Fig. 8 excludes the expected tuple by restricting the predicates CitationCnt >= 60 to CitationCnt > 82 in the second conjunction.

5.2 Inclusion of Expected Tuples

To include the expected tuples in the result set we usually need to relax some constraints for some predicates in the query. In our model, we summarize the explanation of all expected tuples and suggest structural changes for the user’s imprecise query. To do so, we group the expected tuples based on what changes we need to include them in the result set. Then, we propose structural changes to the closest conjunctions in DQ for each group.

**Example 11:** The sensitivity, specificity and balanced accuracy is defined as follows:

\[
\text{sensitivity} = \frac{n_p}{n_p + n_{fp}} \quad \text{and} \quad \text{specificity} = \frac{n_n}{n_n + n_{fn}}
\]

Sensitivity measures the proportion of actual positives tuples which are correctly retained by \(Q\). On the other hand, specificity measures the proportion of negatives (unexpected) tuples which are correctly excluded by \(Q\). An alternate measure that offers a trade-off between sensitivity and specificity is balanced accuracy. Balanced accuracy is defined as follows:

\[
\text{Accuracy} = \frac{\alpha \cdot n_p + \beta \cdot n_n}{n_p + n_{fp} + n_n + n_{fn}}
\]

The values of \(\alpha\) and \(\beta\) are generally set to 0.5 to treat both sensitivity and specificity with equal importance. But these values can be set to different values in the range [0, 1] to have different preferences for sensitivity and specificity. The imprecision metric defined by Tran and Chan (2010) covers only why-not (expected) questions and therefore does not serve our purposes.

**Table 4: Quality of the refined query (shown in Fig. 9) results with and without implicit feedback**

Now consider the implicit ‘yes’ tuples P4 and P9 in our example tuple set. Tuple P4 dominates tuple P3 and tuple P9 dominates P8 in terms of CitationCnt and Rank. Therefore, these tuples are retained in the result set by the
refined query given in Fig. 9. But if we add the implicit predicate (e.g., Book = ‘sigmod’) to the first conjunction of \(DQ\) in the refined query, tuples \(P4\) and \(P9\) are filtered out. Therefore, the corresponding metric is boosted as we see from Table 4. That is, implicit feedback greatly reduces the false positives rate and results in increased specificity.

5.3.2 Similarity of Refined Query to the Original Query

Any query refinement framework should emphasize on producing refined queries which are as similar as possible to the original input query. This has intuitive appeal since a refined query that is minimally modified from the original query is likely to retain as much of the intention of the original input query. Tran and Chan (2010) propose edit distance as the dissimilarity metric for this purpose. The edit operators considered in their work are as follows: modify the constant value of a selection predicate in the where-clause, add a selection predicate in the where-clause, add/remove a join predicate in the where-clause, and add/remove a relation in the from-clause. In our framework, we define imprecise queries that require the conditions to be slightly adjusted. That is, we allow only relaxation and/or restriction of query conditions in the refined queries. But there could be some predicates that need to be unaltered from users’ point of view as mentioned by Kießling and Köstler (2002).

We define the following distance metric between the initial imprecise query \(Q\) and the refined query \(Q'\) to take into account the user preferences:

\[
dis(Q, Q') = \frac{1}{m} \sum_{i=1}^{m} \left( \sum_{j=1}^{n} w_{ij} \cdot d(C_{ij}, C'_{ij}) \right)
\]

where \(w_{ij}\)'s are the predicate preferences of user and are set to between 0 and 1. If \(w_{ij}\) is set to 1, it means user prefers to maintain this predicate strongly (hard constraints as defined by Kießling and Köstler 2002). If it is 0, then the user does not care about the replacement of the corresponding predicates in the refined query. If it is between 0 and 1, then the user is flexible to slightly modify the corresponding constraints (soft constraints as defined by Kießling and Köstler 2002). The distance function \(d(C_{ij}, C'_{ij})\) is currently open in our framework.

As an example, we define them as follows depending on whether \(a_{ij}\) is numeric or categorical attribute:

(a). \(d(C_{ij}, C'_{ij}) = \frac{|v_{ij} - v'_{ij}|}{\max(v_{ij}, v'_{ij})}\) if \(a_{ij}\) is numeric.

(b). \(d(C_{ij}, C'_{ij}) = \begin{cases} 1 & \text{if } v_{ij} \neq v'_{ij} \\ 0 & \text{if } v_{ij} = v'_{ij} \end{cases}\) if \(a_{ij}\) is categorical.

The corresponding similarity metric is defined as follows:

\[
sim(Q, Q') = 1 - \dis(Q, Q')
\]

**Example 12:** The refined query given in Fig. 9 is 7.2% dissimilar to the original query given in Fig. 1. That is, the refined query is 92.8% similar to the original query. The individual weight for each predicate is set to 0.5 for the calculation of similarity score. If we consider the implicit predicates (e.g., Book = ‘sigmod’) to the first conjunction of \(DQ\) in the refined query as given in Example 7, then the similarity score is decreased as we see from Table 5. That is, the more new predicates we add the more dissimilar the refined query becomes to the original query.

<table>
<thead>
<tr>
<th>Implicit Predicates</th>
<th>(\dis(Q, Q'))</th>
<th>(\sim(Q, Q'))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not Considered</td>
<td>7.2%</td>
<td>92.8%</td>
</tr>
<tr>
<td>Considered</td>
<td>38.2%</td>
<td>61.8%</td>
</tr>
</tbody>
</table>

**Table 5:** Similarity score of the refined query (shown in Fig. 9) with and without implicit predicates

5.3.3 Combined Score on Quality and Similarity Metric

To emphasize on both quality of refined query results and similarity of the refined query to the original user submitted query we define a combined metric as follows:

\[
\text{Combined Score} = \delta \cdot \text{quality} + (1 - \delta) \cdot \text{sim}(Q, Q')
\]

The value of \(\delta\) can be set to 0.5 to treat both quality and similarity metric with equal importance. To have different preferences on quality and similarity metric \(\delta\) can be set to any value in the range \([0, 1]\). We can set \(\delta\) to 0 to ignore the quality metric and 1.0 to ignore the similarity score. The combined score for the refined query given in Fig. 9 together with implicit feedback considered or not is given in Table 6.

**Table 6:** Combined score together with implicit feedback and predicates considered or not

We set \(\delta\) to 0.5 for the calculation of the combined score. Table 6 manifests that the combined score provides a balance between quality and similarity metric for the refined query, though the relationship between the quality and similarity metric is reciprocal (if we add implicit predicates to the refined query, it improves the quality and as a side effect it makes the refined query more dissimilar to the original query).

5.4 Future Challenges

Though the explanation based query refinement model is promising in producing highly similar refined queries to the user submitted query, there are several challenges that need to be addressed before its successful application.

Firstly, there could be more than one ways to exclude an unexpected tuple if the corresponding query is \(DQ\). This is because if the predicates are ANDed together then to exclude an unexpected tuple we need to dissatisfy at least one of them as we explain it before in our explanation model (Section 3). Thus we need to select a subset of predicates from a conjunction in \(DQ\) to exclude a set of unexpected tuples which is an NP-hard problem. An approximation algorithm for selecting the best subset is needed that can run in polynomial time. By the best
subset we mean the set that incurs minimal changes in the original query. This is also true for CQ and expected tuples. This could be further complicated if we need to both include expected tuples as well as exclude the unexpected tuples through the same conjunction.

Secondly, a user defined quality metric aware query refinement algorithms need to be designed so that the parameters (sensitivity and specificity thresholds, predefined soft and hard constraints) can be specified beforehand. This is because there are few applications including medical databases where it is crucial to maintain the user defined quality metric.

Thirdly, the implicit feedback could be treated carefully, otherwise it could be explosive. In this paper, we show how we can reduce this feedback by identifying the implicit predicates. Besides of that, pre-calculated approximate functional dependencies and concept learning (Nambari et al. 2004) can also be coupled for this purpose.

Lastly, we consider only the adjustment of the constraints in our query refinement model. But insertion/deletion of conditions in the where-clause, add/drop relations in the from clause and even relaxation of join conditions (Tran and Chan 2010) could be considered to provide more flexibility in the explanation based query refinement model.

6 Related Work and Discussion
Huang et al. (2008) explain missing tuple by allowing modifications to the database such that the expected tuple appears in the query result wrt the modified database. But the number of returned alternatives can be huge and difficult for users to comprehend the appropriate explanation for the missing tuple. Herschel et al. (2010) extend the idea of (Huang et al. 2010) to take into account a set of missing tuples and select-project-join-union (SPJU) queries. The intuition of this model is to explain in terms of how to modify some of the untrusted data in order to produce the missing tuple. However, this model may not be applicable in applications where all the data stored are trusted. Champan and Jagadish [4] model explanation of missing tuple by identifying the operator(s) that filters it out from the result set.

Tran and Chan (2010) model explanation of expected tuples by refining the original query to include them in the result set. Authors exploit the idea of skyline queries to report the closest refined query wrt original one to minimize the distance between refined and original queries. It may happen that the number of returned refined queries is overwhelming for the naive user and frustrating as the user looks for exact explanations. Moreover, refined queries may introduce more false positives (unexpected tuples) in the result set. A more helpful explanation should be one that can model both why and why not questions. We propose query refinement exploiting the explanations generated by our framework to minimize the number of unexpected tuples as well as maximize the number of expected tuples.

Liu et al. (2010) collect false positives identified by the users as feedback to modify the initial rules in information extraction settings to exclude unexpected results. However, they don’t consider the expected results to refine the imprecise rules. In our model, we collect both false positives and false negatives to refine the initial imprecise query. We don’t require our user to provide the complete set of unexpected and expected tuples but only a subset of them. We automatically suggest other members based on point domination theory.

Meliou et al. (2010) propose causality as a unified framework to explain both resultant and non-resultant tuples. This approach generalizes and extends previously proposed definitions of provenance by leveraging the lineage of query answers (positive provenance) and non-answers (negative provenance) under the same framework. But they don’t separate unexpected tuples from answers (also expected tuples from non-answers) as we do in our model. We also show how one can modify the initial imprecise query exploiting the explanations of expected and unexpected tuples generated by our framework.

The need of supporting exploratory or imprecise queries in RDM and Web databases is studied with great importance in (Nambari et al. 2003, Kadlag et al. 2004, Nambari and Kambhampati 2005, Ma et al. 2006, Koudas et al. 2006, Mishra and Koudas 2009). This has particular application to the many/few answers problem often faced by database users. In (Mishra and Koudas 2009) to address this problem user feedback is incorporated to best capture user preferences whereas in (Koudas et al. 2006) a lattice based framework is proposed for minimum amount of query conditions relaxation. We show how one can capture user intent by collecting feedback (a sample set of unexpected and expected tuples) and modify the initial imprecise query exploiting the explanations generated by our model.

7 Conclusion and Future Work
This paper presents a unified framework to explain both the presence of unexpected and the absence of expected tuples. We show how to capture user intent exploiting the explanations generated by our model. For this, we require the user to provide a sample set of expected and unexpected tuples. Then, we show how one can complement it by automatically suggesting the complete set. Finally, we show how we can refine the initial imprecise query to exclude the unexpected tuples exploiting the explanations of them. Similarly, to include the missing information we relax certain predicates that appear in the explanation of expected tuples. In other words, both why and why not causes are exploited by our model to refine the initial imprecise query. We also present future challenges of explanation based query refined model.

Future work of our paper includes addressing the future challenges of explanation based query refinement model and the detailed evaluation of the proposed framework for demonstrating its performance and effectiveness in real world data set. We also plan to adapt our framework to solve many/few answers problem incorporating user feedback and attribute/predicate preferences.

8 References


Optimised X-HYBRIDJOIN for Near-Real-Time Data Warehousing

1M. Asif Naeem 2Gillian Dobbie 2Gerald Weber

Department of Computer Science, The University of Auckland, Private Bag 92019, 38 Princes St, Auckland, New Zealand.
Email: 1maae006@aucklanduni.ac.nz, 2{gill, gerald}@cs.auckland.ac.nz

Abstract

Stream-based join algorithms are needed in modern near-real-time data warehouses. A particular class of stream-based join algorithms, with MESHJOIN as a typical example, computes the join between a stream and a disk-based relation. Recently we have presented a new algorithm X-HYBRIDJOIN (Extended Hybrid Join) in that class. X-HYBRIDJOIN achieves better performance compared to earlier algorithms by pinning frequently accessed data from the disk-based relation in main memory. Apart from being held in main memory, X-HYBRIDJOIN treats this frequently accessed data no differently than other data from the disk-based relation. In this paper we investigate whether performance can be improved by treating the frequently accessed data differently. We present a new algorithm called Optimised X-HYBRIDJOIN, which consists of two phases. One phase, called the stream-probing phase, deals with the frequently accessed part of the disk-based relation. The other one is called the disk-probing phase and deals with the other part of the disk-based relation. In experiments we found that the performance of Optimised X-HYBRIDJOIN is significantly better than the performance of X-HYBRIDJOIN. We derive the cost model for our algorithm, which allows us to tune the components of Optimised X-HYBRIDJOIN. We performed an experimental study and we validate the cost model against the experimental results.

1 Introduction

Near-real-time data warehousing plays nowadays a prominent role in supporting overall business decision making. By extending data warehouses from static data repositories to active data repositories, businesses and other organizations can inform their users better and allow them to take effective and timely decisions.

In near-real-time data warehousing the changes occurring at source level are reflected in data warehouses without any delay. Extraction, Transformation, and Loading (ETL) tools are used to access and manipulate transactional data and then load them into the data warehouse. An important phase in the ETL process is a transformation where the source level changes are mapped into the data warehouse format. Common examples of transformations are unit conversion, removal of duplicate tuples, information enrichment, filtering of unnecessary data, sorting of tuples, and translation of source data key.

Let us consider an example for the transformation phase shown in Figure 1 that implements one of the above features, called enrichment. In the example we consider the source data with attributes product_id, qty, and date that are extracted from data sources. At the transformation layer, in addition to key replacement (from source key product_id to warehouse key s_key) there is some information added, namely sales price denoted by s_price to calculate the total amount, and the vendor information. In the figure these information with attributes name s_key, s_price, and vendor are extracted at run time from the master data and are used to enrich the source updates using a join operator.

In traditional data warehousing the source updates are buffered and the join is performed off-line. On the other hand, in near-real-time data warehousing this operation needs to be performed as soon as the data are received from the data sources. In implementing the online execution of join, one important challenge is the different character of both inputs. The stream input is fast and huge in volume while the disk input is slow. The challenge here is to amortise the disk access cost over the fast input stream.

A stream-based join algorithm called X-HYBRIDJOIN (Extended Hybrid Join) (Naeem et al. 2011) was proposed to deal with these challenges. In addition, the algorithm is designed to be particular efficient for Zipfian distributions as they are frequently found in practice. A frequently cited rule of thumb is the 80/20 rule (Anderson 2006). According to this rule 80% of sales in an e-commerce setting is based on 20% of the products and therefore, a small number of pages in master data are frequently used during the join operation. The algorithm used a buffer to load a specific portion of master data into memory. To eliminate the bottleneck in the stream the algorithm divides this buffer into two equal segments. One segment is non-swappable and holds a small number of the frequently accessed page(s) of master data permanently in memory while the other segment is swappable and exchanges its contents on each iteration of the algorithm. The main argument presented in the algorithm is that storing the frequently accessed part of the master data permanently in memory minimises the disk I/O cost and that eventually amortises the fast incoming stream of updates. An alternative approach would be to try to put the whole disk-based relation into memory. In some cases this alternative can be feasible. But still there are a number of scenarios where this alternative is not applicable e.g. if the join is to be performed on a single computer where the
queries (Chen et al. 2000) (Liu et al. 2004) (Avnur et al. 2000) (Babcock et al. 2003) (Chandrasekaran et al. 2002) (Dobra et al. 2002). Our focus is particularly on stream-based joins in stream-based databases. In the next sections, we further divide our literature review into two categories. In the first category we overview those join operators where all the inputs are in the form of a stream. In the second category we consider those join algorithms in which one input comes in the form of a stream while the other input comes from disk.

First Category: Symmetric Hash Join (SHJ) (Hong et al. 1991) (Wilscuit et al. 1991) has exploited the concepts of the traditional hash join algorithm by eliminating the delay for the input stream. SHJ maintains hash tables for both input streams in memory. Each new tuple from one stream is joined with the other stream stored in a hash table and the output for the joined tuple is generated. After generating the output the tuple is stored in its own hash table. The algorithm can generate the output as early as both matching tuples have arrived. However, it needs to store both inputs in memory.

XJoin (Tolga et al. 2000) is an extended form of SHJ that handles memory overflow by flushing the largest single partition on disk. XJoin presents a three stage strategy to switch its execution state between disk and memory. First priority is given to memory-resident tuples. During times where there are no incoming stream data, the algorithm executes the second, disk-to-memory phase and lastly deals with the tuples stored on disk (disk-to-disk) in the case when the inputs are terminated. In XJoin duplicate tuples are avoided by using a timestamp approach.

The double Pipelined Hash Join (DPHJ) (Ives et al. 1999) is also an extension of symmetric hash join based on two stages. In the first stage, which is similar to SHJ and XJoin, the algorithm joins the tuples which are in memory. In the second stage the algorithm marks the tuples which are not joined in memory and joins them on the disk. In DPHJ duplication of tuples is possible in the second phase when all tuples from both inputs have been read and the final clean-up join is executed. This algorithm is suitable for medium size data and does not perform well for large data. Hash-Merge Join (HMJ) (Mokbel et al. 2004), also one from the series of symmetric joins, is based on push technology and consists of two phases, hashing and merging.

All three approaches above do not consider the metadata about the stream. Therefore, they are unable to recognize data which is no-longer required and by dropping it the overhead can be reduced. In addition, the join approaches above focus on throughput optimisation while ignoring the other optimisation goals such as the characteristics of stream data which are equally important.

MJoin (Viglas et al. 2003), a generalised form of XJoin, extends the symmetric binary join operators to handle multiple inputs. MJoin uses a separate hash table for each input. On the arrival of a tuple from an input, it is stored in the corresponding hash table and is probed into the rest of the hash tables. It is not necessary to probe all hash tables for each arrival, the sequence of probing stops when a probed tuple does not match in a hash table. The methodology for choosing the correct sequence of probing is determined by performing the most selective probes first. The algorithm uses a coordinated flushing technique that involves flushing the same partition on disk for all inputs. All three stages from XJoin are included in MJoin. To identify the duplicate tuples MJoin uses two timestamps for each tuple, arrival time and departure time from memory.

In X-HYBRIDJOIN, introducing the non-swappable part in the disk buffer reduces the disk I/O cost. However, there are some unnecessary processing costs that negatively affect the performance of the algorithm. For example in each iteration the algorithm matches all tuples in the non-swappable part of the disk buffer with the hash table. It increases the unnecessary look-up cost for the algorithm. Similarly, the algorithm stores all stream tuples, whether they join with the swappable or non-swappable part of the disk buffer, in memory increasing the cost in terms of loading and unloading the stream tuples into memory. These overheads in terms of extra costs can be removed by improving the architecture of the algorithm.

After considering these observations, we propose an improved version of X-HYBRIDJOIN known as Optimised X-HYBRIDJOIN (Optimised Extended Hybrid Join). In Optimised X-HYBRIDJOIN we divide the algorithm in two phases and both phases can work independently. One phase deals with the swappable part while the other phase deals with the non-swappable part of the disk-based relation using appropriate data structures. In the proposed algorithm, due to choosing an appropriate architecture all unnecessary costs are eliminated and performance is improved significantly. To make our algorithm more efficient we also present the tuning of the algorithm based on a mathematical cost model.

The rest of the paper is structured as follows. The previous work related to the area is presented in Section 2. Section 3 describes our observations for the current algorithm. In Section 4 we present the proposed algorithm with its execution architecture, pseudo-code, cost model and tuning. The experimental study is described in Section 5 and finally Section 6 concludes the paper.

2 Previous work

Considerable work has been done on executing join queries (Chen et al. 2000) (Liu et al. 2004) (Avnur et al. 2000) (Babcock et al. 2003) (Chandrasekaran et al. 2002) (Dobra et al. 2002). Our focus is particularly on stream-based joins in stream-based databases. In the next sections, we further divide our literature review into two categories. In the first category we overview those join operators where all the inputs are in the form of a stream. In the second category we consider those join algorithms in which one input comes in the form of a stream while the other input comes from disk.

First Category: Symmetric Hash Join (SHJ) (Hong et al. 1991) (Wilscuit et al. 1991) has exploited the concepts of the traditional hash join algorithm by eliminating the delay for the input stream. SHJ maintains hash tables for both input streams in memory. Each new tuple from one stream is joined with the other stream stored in a hash table and the output for the joined tuple is generated. After generating the output the tuple is stored in its own hash table. The algorithm can generate the output as early as both matching tuples have arrived. However, it needs to store both inputs in memory.

XJoin (Tolga et al. 2000) is an extended form of SHJ that handles memory overflow by flushing the largest single partition on disk. XJoin presents a three stage strategy to switch its execution state between disk and memory. First priority is given to memory-resident tuples. During times where there are no incoming stream data, the algorithm executes the second, disk-to-memory phase and lastly deals with the tuples stored on disk (disk-to-disk) in the case when the inputs are terminated. In XJoin duplicate tuples are avoided by using a timestamp approach.

The double Pipelined Hash Join (DPHJ) (Ives et al. 1999) is also an extension of symmetric hash join based on two stages. In the first stage, which is similar to SHJ and XJoin, the algorithm joins the tuples which are in memory. In the second stage the algorithm marks the tuples which are not joined in memory and joins them on the disk. In DPHJ duplication of tuples is possible in the second phase when all tuples from both inputs have been read and the final clean-up join is executed. This algorithm is suitable for medium size data and does not perform well for large data. Hash-Merge Join (HMJ) (Mokbel et al. 2004), also one from the series of symmetric joins, is based on push technology and consists of two phases, hashing and merging.

All three approaches above do not consider the metadata about the stream. Therefore, they are unable to recognize data which is no-longer required and by dropping it the overhead can be reduced. In addition, the join approaches above focus on throughput optimisation while ignoring the other optimisation goals such as the characteristics of stream data which are equally important.

MJoin (Viglas et al. 2003), a generalised form of XJoin, extends the symmetric binary join operators to handle multiple inputs. MJoin uses a separate hash table for each input. On the arrival of a tuple from an input, it is stored in the corresponding hash table and is probed into the rest of the hash tables. It is not necessary to probe all hash tables for each arrival, the sequence of probing stops when a probed tuple does not match in a hash table. The methodology for choosing the correct sequence of probing is determined by performing the most selective probes first. The algorithm uses a coordinated flushing technique that involves flushing the same partition on disk for all inputs. All three stages from XJoin are included in MJoin. To identify the duplicate tuples MJoin uses two timestamps for each tuple, arrival time and departure time from memory.
Early Hash Join (Lawrence et al. 2005) is an improved version of XJoin with a different flushing strategy and an amplified technique to determine the duplicate tuples. EHJ uses a biased flushing strategy that supports flushing the partition with large input first. The technique used in EHJ to determine the duplicate tuples is based on cardinality. For one-to-one and one-to-many relationships the algorithm does not use any timestamp while for many-to-many relationships it requires an arrival timestamp only. In the approaches above, the algorithms expect all inputs in the form of streams, they are not adaptive in the context of near-real-time data warehousing where one input normally comes from disk.

**Second Category:** In the near-real-time data warehousing context, there is a need for joins between a stream of source updates and a disk-based master data relation. This scenario naturally arises for near-real-time data warehouses, if an incoming stream of user data has to be joined with master data.

The MESHJOIN (Mesh Join) algorithm (Polyzotis et al. 2007) (Polyzotis et al. 2008) has been introduced with the objective to amortise the slow disk access with as many stream tuples as possible. To perform the join, the algorithm needs a number of chunks of stream input at the same time. In each iteration, the algorithm loads a disk partition into memory and performs the join with all these stream chunks. The algorithm performs tuning for efficient memory distribution among the join components, but we identified in the past some issues around the access to the disk based relation. Also MESHJOIN cannot deal with intermittency of the stream efficiently.

R-MESHJOIN (reduced Mesh Join) (Naeem et al. 2010) is an enhanced form of MESHJOIN in which one issue related to suboptimal distribution of memory among the join components is resolved. However, R-MESHJOIN implements the same strategy as the MESHJOIN algorithm for accessing the disk-based relation.

A partition-based approach (Chakraborty et al. 2009) has been introduced to deal with intermittency in the stream. It uses a two-level hash table to attempt to join stream tuples as soon as they arrive, and uses a partition-based waiting area for the other stream tuples. The authors do not provide a cost model for their approach. In addition, the algorithm requires a clustered index or an equivalent sorting on the join attribute and it does not prevent starvation of stream tuples.

Another algorithm, HYBRIDJOIN (Hybrid Join) (Naeem et al. 2011) address the issue of accessing the disk-based relation. An effective strategy to access the disk-based relation is introduced in HYBRIDJOIN. Another advantage of HYBRIDJOIN is that it can deal with bursty streams, which is a limitation of both MESHJOIN and R-MESHJOIN. However, if we consider long-tail distributions, we find that the algorithm can be improved further.

The X-HYBRIDJOIN (Naeem et al. 2011) algorithm that we focus on in this paper is an extension of HYBRIDJOIN. This algorithm has been designed particularly to cope with Zipfian distributions. Although this is an adaptive algorithm and performs better than other similar approaches, there are some limitations at the architectural level that needs to be explored and improved further.

The motivation behind Optimised X-HYBRIDJOIN is to refine the existing approach in order to minimize the bottleneck in the stream of updates.

### Problem definition

In this section we first give an overview of X-HYBRIDJOIN and then identify the limitations that we observed in this algorithm. Our cost models are based on a non-uniform distribution on foreign keys in the stream data. In a real-world data warehousing scenario, uniform distributions are rarely encountered. Instead, frequencies often follow power laws, also known as Zipfian distributions. While power laws are natural surface properties of large data populations, the exponent that governs the power law can vary. Generally, smaller values of exponent give so-called short tails, bigger values of exponent give long tails. Long tails are interesting for scalable, very large data warehouses, since long tails are tipped to become more important in tomorrow’s economy, if consumer behavior diversifies (Anderson 2006).

Before going into further detail we first explain the major components of X-HYBRIDJOIN and the role of each component. Figure 2 presents an overview of X-HYBRIDJOIN showing a queue to store stream tuples, a two-part disk buffer, and the disk-based relation $R$. In actual the algorithm stores stream tuples in the hash table however, for simplicity we assume that these stored in the hash table. To make efficient use of relation $R$ by minimizing the disk access cost, the disk buffer is divided into two equal parts. One is called the non-swappable part, and stores a small but frequently accessed portion of relation $R$ in memory permanently. The other part of the disk buffer is swappable and for each iteration it loads the disk partition $p_i$ from relation $R$ into memory.

The key idea behind how X-HYBRIDJOIN works is that before the actual execution starts, the algorithm loads the frequently used part of relation $R$ into the non-swappable part of the disk buffer. After the actual execution starts, for each iteration the algorithm reads the oldest tuple from the queue and using this tuple as an index it loads the relevant disk partition into the disk buffer. Then the algorithm matches one-by-one all disk tuples available in both the swappable and non-swappable parts of the disk buffer with the stream tuples in the queue. If the matching is true, the algorithm generates that stream tuple as an output after deleting it from the queue. In the next iteration the algorithm again reads the oldest tuple from the queue, loads the relevant disk partition into the disk buffer and repeats the entire procedure.

Although in X-HYBRIDJOIN, introducing the new component called non-swappable part of disk buffer reduced the disk access cost, it also raised some issues related to processing cost. Firstly, for each iteration, the algorithm looks-up one-by-one all the disk tuples stored in both the swappable and the non-swappable parts of the disk buffer in the hash table. It increases the unnecessary look-up cost for the algorithm particularly when the corresponding stream tuple does not exist in the hash table. Secondly, the algorithm stores all stream tuples, whether they join with the swappable or the non-swappable part of the disk buffer, in memory. As a result, it introduces extra processing costs for algorithms in terms of loading these tuples into the hash table and removing them
from the hash table after processing. Contrarily, if we store only those stream tuples in memory that join with the swappable part of the disk buffer, we can accommodate more stream tuples in memory at the same time.

In summary, the problem that we consider in this paper is, how can we eliminate these unnecessary processing costs that occur in X-HYBRIDJOIN by improving its architecture and consequently the way the algorithm works.

4 Proposed solution

In this section, we propose a new algorithm called Optimised X-HYBRIDJOIN (Optimised Extended Hybrid Join) that overcomes the problems that we identified in X-HYBRIDJOIN. Optimised X-HYBRIDJOIN decomposes the algorithm into two hash join phases that can execute separately. One phase uses \( R \) as the probe input; the largest part of \( R \) will be stored in tertiary memory. This phase is called the disk-probing phase. The other join phase uses the stream as the probe input and it is called the stream-probing phase. This phase deals only with a small part of relation \( R \). For each incoming stream tuple, Optimised X-HYBRIDJOIN first uses the stream-probing phase to find a match for frequent requests quickly, and if no match is found, the stream tuple is forwarded to the disk-probing phase. The details of the proposed algorithm are presented in the following subsections.

4.1 Memory architecture

This section gives a high-level description of Optimised X-HYBRIDJOIN, while a detailed walkthrough of the algorithm can be found in Section 4.2. From the architectural point of view, the key concept in Optimised X-HYBRIDJOIN is to execute both the disk-probing phase and the stream-probing phase independently, using appropriate data structures. The reason for doing this is to eliminate unnecessary costs, as described later in this section. The memory architecture for Optimised X-HYBRIDJOIN is shown in Figure 3. The largest components of Optimised X-HYBRIDJOIN with respect to memory size are two hash tables, one storing stream tuples, denoted by \( H_S \), and the other storing tuples from the disk-based relation, denoted by \( H_R \). The other main components of Optimised X-HYBRIDJOIN are a disk buffer, a queue and a stream buffer. Disk-based relation \( R \) and stream \( S \) are the external input sources. Similar to X-HYBRIDJOIN, \( R \) is assumed to be sorted according to the frequency of access. The hash table \( H_R \) contains the frequently-accessed part of \( R \), which is stored permanently in memory.

Optimised X-HYBRIDJOIN alternates between the stream-probing and disk-probing phases. The hash table \( H_S \) is used to store only that part of the update stream which does not match tuples in \( H_R \). A stream-probing phase ends if \( H_S \) is completely filled or if the stream buffer is empty. Then the disk-probing phase becomes active. The length of the disk-probing phase is determined by the fact that only a small number of disk pages of \( R \) have to be loaded at one time in order to amortise the costly disk access. In the disk-probing phase of Optimised X-HYBRIDJOIN, the oldest tuple in the queue is used to determine the partition of \( R \) that is loaded for a single probe step. This is also the step where Optimised X-HYBRIDJOIN needs an index on table \( R \) in order to find the partition in \( R \) that matches the oldest stream tuple. After one probe step, a sufficient number of stream tuples are deleted from \( H_S \), so the algorithm switches back to the stream-probing phase. One phase of stream-probing with a subsequent phase of disk-probing constitutes one outer iteration of Optimised X-HYBRIDJOIN. The disk-probing phase could work on its own, without the stream-probing phase. Therefore, the stream-probing phase can be switched-off if it is not required and the memory needed for that phase would be reassigned.

The stream-probing phase is used to boost the performance of the algorithm by quickly matching the frequently-used master data. The disk buffer stores the swappable part of \( R \) and for each iteration it loads a particular partition of \( R \) into the memory. The other component queue is based on a doubly-linked-list, and is used to store the values for the join attribute. Each node in the queue also contains the addresses of its neighbour nodes. The reason for choosing this data structure is to allow random deletion from the queue. The stream buffer is included in the diagram for completeness, but is in reality always a tiny component and will not be considered in the cost model.

There are two key advantages of Optimised X-HYBRIDJOIN over X-HYBRIDJOIN. First, due to the independent processing of each phase the stream tuples can be looked-up directly in \( H_R \) without loading them into memory. This not only eliminates an unnecessary look-up cost, but also allows more of the stream to be accommodated in memory. In contrast to this, X-HYBRIDJOIN stores a major part of the stream, related to the non-swappable part, in memory and for each iteration, the algorithm looks-up all
the tuples of the non-swappable part in the hash table only-by-one. In the situation when the tuples do not match, the algorithm faces an additional look-up cost. Secondly, since Optimised X-HYBRIDJOIN does not store a large part of the stream in memory, it eliminates the costs of loading and unloading that part of the stream into the hash table, $H_S$. These additional features in Optimised X-HYBRIDJOIN help in reducing the overall processing cost for the algorithm and that eventually improves the performance.

### 4.2 Algorithm

After dividing the available memory among the join components, the algorithm starts its execution. The pseudo-code for Optimised X-HYBRIDJOIN is shown in Algorithm 1. The outer loop of the algorithm is an endless loop (line 2). The body of the outer loop has two main phases, the stream-probing phase and the disk-probing phase. Due to the endless loop, these two phases are executed alternately.

Lines 3 to 11 comprise the stream-probing phase. The stream-probing phase has to know the number of empty slots in $H_S$. This number is kept in variable $hS_{available}$. At the start of the algorithm, all the slots in $H_S$ are empty (line 1). The stream-probing phase has an inner loop that continues while stream tuples as well as empty slots in $H_S$ are available (line 3). In the loop, the algorithm reads one input stream tuple $t$ at a time (line 4). The algorithm looks up $t$ in $H_R$ (line 5). In the case of a match, the algorithm generates the join output without storing $t$ in $H_S$ (line 6). In the case where $t$ does not match, the algorithm loads $t$ into $H_S$, along with enqueuing its key attribute value in the queue (line 8). The counter of empty slots in $H_S$ then has to be decreased (line 9).

Lines 12 to 21 comprise the disk-probing phase. At the start of this phase, the algorithm reads the oldest key attribute value from the queue and loads a partition of $R$ into the disk buffer, using that key attribute value as an index (lines 12 and 13). In an inner loop, the algorithm looks up all tuples $r$ from the disk buffer in hash table $H_S$ one-by-one. In the case of a match, the algorithm generates the join output (line 16). Since $H_S$ is a multi-hash-map, there can be more than one match, the number of matches being $f$ (line 17). The algorithm removes all matching tuples from $H_S$ along with deleting the corresponding nodes from the queue (line 18). This creates empty slots in $H_S$ (line 19). In the next outer iteration the algorithm fills these empty slots if stream input is available.

### 4.3 Cost calculation

In this section we develop the cost model for our proposed Optimised X-HYBRIDJOIN. The main objective for developing our cost model is to interrelate the key parameters like the algorithm input size $w$, processing cost $c_{proc}$ for these $w$ tuples, the available memory $M$ and the service rate $\mu$. The other important application for our cost model is in the tuning process where the optimal size is determined for each component of the algorithm. The details about the tuning process are presented in Section 4.4. Normally, the main costs for an algorithm are described in terms of the distribution of memory to the components and processing time. We calculate both of these costs for our proposed Optimised X-HYBRIDJOIN. Equation 1 represents the total memory used by the algorithm except the stream buffer, and Equation 2 describes the processing cost for each iteration of the algorithm. The notations we used in our cost model are specified in Table 1.

#### 4.3.1 Memory cost

The optimal size for hash table $H_R$ can be different from the optimal size of the disk buffer. Therefore, we distinguish between $k$, the number of pages for the disk buffer, and $l$, the number of pages for $H_R$. The major portion of the total memory is assigned to the both hash tables while a much smaller portion comparatively is assigned to the disk buffer and the queue. The memory for each component can be calculated as given below.

- Memory for disk buffer = $kv_p$
- Memory for $H_R$ = $l v_p$
- Memory for $H_S$ = $\alpha [M - (k + l)v_p]$
- Memory for the queue = $(1 - \alpha) [M - (k+l) v_p]$

By aggregating the above, the total memory used by Optimised X-HYBRIDJOIN can be calculated as shown in Equation 1.

$$M = (k+l) v_p + \alpha [M - (k + l)v_p] + (1 - \alpha) [M - (k+l) v_p]$$

(1)

Currently, the memory for the stream buffer is not included because it is small (0.05 MB is sufficient in all our experiments).

#### 4.3.2 Processing cost

In this section we calculate the processing cost for the algorithm. To make it simple we first calculate the processing cost for individual components and then

---

**Algorithm 1 Optimised X-HYBRIDJOIN**

**Input:** A disk based relation $R$ with an index on join attribute and a stream of updates $S$.

**Output:** $R \bowtie S$

**Parameters:** $w$ (where $w = w_S + w_N$) tuples of $S$ and $k$ pages of $R$.

**Method:**

1. $hS_{available} \leftarrow hS$
2. while (true) do
3. while (stream available AND $hS_{available} > 0$) do
4. READ a stream tuple $t$ from the stream buffer
5. if $t \in H_R$ then
6. OUTPUT $t \bowtie H_R$
7. else
8. ADD the stream tuple $t$ into $H_S$ while also placing its join attribute values into $Q$
9. $hS_{available} \leftarrow hS_{available} - 1$
10. end if
11. end while
12. READ the oldest join attribute value from $Q$
13. READ a partition of $R$ into the disk buffer using the oldest join attribute value from the queue for the index look-up.
14. for each tuple $r$ in the disk buffer do
15. if $r \in H_S$ then
16. OUTPUT $r \bowtie H_S$
17. $f \leftarrow$ number of matching tuples found in $H_S$
18. DELETE all matched tuples from $H_S$ along with the corresponding nodes from $Q$
19. $hS_{available} \leftarrow hS_{available} + f$
20. end if
21. end for
22. end while

---

**Table 1** Notations used in cost model

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>Number of pages for disk buffer</td>
</tr>
<tr>
<td>$l$</td>
<td>Number of pages for $H_R$</td>
</tr>
<tr>
<td>$w$</td>
<td>Algorithm input size</td>
</tr>
<tr>
<td>$M$</td>
<td>Total available memory</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Service rate</td>
</tr>
<tr>
<td>$c_{proc}$</td>
<td>Processing cost</td>
</tr>
<tr>
<td>$hS_{available}$</td>
<td>Number of empty slots in $H_S$</td>
</tr>
<tr>
<td>$R$</td>
<td>Disk based relation $R$</td>
</tr>
<tr>
<td>$S$</td>
<td>Stream of updates $S$</td>
</tr>
</tbody>
</table>
Table 1: Notations used in cost estimation of Optimised X-HYBRIDJOIN

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total allocated memory (bytes)</td>
<td>( M )</td>
</tr>
<tr>
<td>Service rate (processed tuples/sec)</td>
<td>( \mu )</td>
</tr>
<tr>
<td>Number of stream tuples processed in each iteration through ( H_R )</td>
<td>( w_N )</td>
</tr>
<tr>
<td>Number of stream tuples processed in each iteration through ( H_S )</td>
<td>( w_S )</td>
</tr>
<tr>
<td>Stream tuple size (bytes)</td>
<td>( v_S )</td>
</tr>
<tr>
<td>Disk page size (bytes)</td>
<td>( v_P )</td>
</tr>
<tr>
<td>Disk buffer size (bytes)</td>
<td>( k )</td>
</tr>
<tr>
<td>Disk buffer size (tuples)</td>
<td>( d = k \frac{R_R}{v_P} )</td>
</tr>
<tr>
<td>Size of ( H_R ) (pages)</td>
<td>( l )</td>
</tr>
<tr>
<td>Size of ( H_R ) (tuples)</td>
<td>( R_R = l \frac{v_P}{v_H} )</td>
</tr>
<tr>
<td>Size of ( H_S ) (tuples)</td>
<td>( h_S )</td>
</tr>
<tr>
<td>Disk relation size (tuples)</td>
<td>( R )</td>
</tr>
<tr>
<td>Memory weight for the hash table</td>
<td>( \alpha )</td>
</tr>
<tr>
<td>Memory weight for the queue</td>
<td>( 1 - \alpha )</td>
</tr>
<tr>
<td>Cost to read ( k ) number of disk pages into the disk buffer (nano secs)</td>
<td>( c_{I/O}(k \cdot v_P) )</td>
</tr>
<tr>
<td>Cost to look-up one tuple into the hash table (nano secs)</td>
<td>( c_H )</td>
</tr>
<tr>
<td>Cost to generate the output for one tuple (nano secs)</td>
<td>( c_O )</td>
</tr>
<tr>
<td>Cost to read one tuple from the hash table and the queue (nano secs)</td>
<td>( c_E )</td>
</tr>
<tr>
<td>Cost to read one stream tuple into the stream buffer (nano secs)</td>
<td>( c_S )</td>
</tr>
<tr>
<td>Cost to append one tuple in the hash table and the queue (nano secs)</td>
<td>( c_A )</td>
</tr>
<tr>
<td>Total cost for one loop iteration of the algorithm (secs)</td>
<td>( c_{loop} )</td>
</tr>
</tbody>
</table>

The hash table \( H_R \) is sorted according to the access relationships it needs to store multiple values in the hash table against one key value. However the hash table provided by Java API does not support this feature therefore, we used Multi-Hash-Map, provided by Apache Common Collections, to implement the hash table in our experiments. The detailed specifications of the data set that we used for analysis is shown in Table 2.

\begin{equation}
\mu = \frac{w_N + w_S}{c_{loop}} \tag{3}
\end{equation}

4.4 Tuning

Normally the stream-based join algorithms are executed online, where limited memory resources are available. Due to the fixed and small amount of available memory, each component in the join faces a trade-off with respect to memory distribution. Assigning more memory to one component means assigning less memory to some other components. On close observation it can be seen that the components like both hash tables require more memory compared to the other components, such as the disk buffer, the stream buffer and the queue.

The disk buffer and the hash table \( H_R \) are the key components for tuning, and the memory assigned to the other components depends on them. The reason for tuning the disk buffer is that the dominant I/O cost is directly connected to the disk buffer.

Tuning is not performed merely using a theoretical approach, rather the optimal tuning settings are approximated using an empirical approach. Finally the experimentally-obtained tuning results are compared with the results obtained using the cost model. Before proceeding further it is first necessary to describe the hardware and software specifications for our experiments.

4.4.1 Experimental arrangement

The details about the experimental setup that we used to implement the prototypes for all comparable algorithms are given below.

**Hardware specifications:** We accomplished our experiments on Pentium-IV machine with 3G main and 160G disk memory under Windows XP. We implemented the experiment in Java using the Eclipse IDE 3.2.1.1. We also used built-in plugins, provided by Apache, and nanoTime(), provided by Java API, to measure the memory and processing time respectively.

**Data specifications:** We analysed the performance of the algorithms using synthetic data. The relation \( R \) is stored on disk using a MySQL 5.0 database while the bursty stream is generated at run time using our own benchmark algorithm described in (Naeem et al. 2011) with an exponent value equal to 1. As in X-HYBRIDJOIN we also assume that the disk-based relation \( R \) is sorted according to the access frequency. To measure the I/O cost more accurately we set the fetch size for ResultSet equal to the disk buffer size.

Currently the Optimised X-HYBRIDJOIN supports join for one-to-one and one-to-many relationships. In order to implement the join for one-to-many relationships it needs to store multiple values in the hash table against one key value. However the hash table provided by Java API does not support this feature therefore, we used Multi-Hash-Map, provided by Apache Common Collections, to implement the hash table in our experiments. The detailed specifications of the data set that we used for analysis is shown in Table 2.

**Measurement strategy:** The performance or service rate of the join is measured by calculating the number of tuples processed in a unit second. In each experiment the algorithm runs for one hour and we start our measurements after 10 minutes and continue it for 30 minutes. For more accuracy we take

\[ c_{loop}(secs) = 10^{-9} [c_{I/O}(k \cdot v_P) + d \cdot c_H + w_S(c_O + c_E) + c_S + c_A] + w_N(c_H + c_O + c_S)] \tag{2} \]
three readings for each specification and then take their average as a final result. Where required we also calculate the confidence interval by considering 95% accuracy. Moreover, during the execution of the algorithm it is assumed that no other application is run in parallel.

4.4.2 Tuning using Empirical Approach

This section focuses on the tuning of key components, namely the disk buffer and the hash table $H_R$ using an empirical approach. The performance of the algorithm has been tested for a set of values for both components, rather than for every consecutive value. It has been assumed that the total allocated memory and the size of the disk-based relation are fixed. The sizes for the disk buffer and the hash table $H_R$ are varied in such a way that for each size of the disk buffer the performance is measured against a series of values for the size of $H_R$. The performance measurements for the grid of values for the sizes of the disk buffer denoted by $d$ and the size of $H_R$ denoted by $h_R$ are shown in Figure 4. The figure shows that, if the performance for each fixed value of $d$ is observed against all values of $h_R$, in the beginning the performance increases rapidly with an increase in $h_R$. However, after reaching a particular value of $h_R$, the performance starts decreasing with further increases in $h_R$. A plausible reason for this behavior is that initially, increasing $h_R$ increases the probability of matching the stream tuples with $H_R$ rapidly. After attaining the optimal value, further incrementing $h_R$ makes no significant difference to the stream-matching probability, due to the skew factor in stream distribution. On the other hand, the associated reduction in memory size for the hash table $H_S$ means that the performance begins to decrease. Similarly when the performance is analysed for each fixed value of $h_R$ against all the values of $d$, initially the performance increases, since the costly disk access is amortised for a larger number of stream tuples. After attaining a maximum, the performance decreases because of the increase in I/O cost for loading more of $R$ at one time in a non-selective way.

The figure shows that the optimal memory settings for both the disk buffer and the hash table $H_R$ can be determined by considering the intersection of the values of both components at which the algorithm individually performs at a maximum.

4.4.3 Tuning based on cost model

To validate our cost model against measurements, we tune our algorithm based on this cost model as presented in Section 4.3. According to Equation 3 the service rate depends on the values of $w_S$, $w_N$ and the cost $c_{loop}$. Therefore, to determine the settings at which the algorithm performs optimally it is first necessary to calculate the sizes of $w_S$ and $w_N$.

Mathematical model to calculate $w_N$: The main components that directly affect $w_N$ are the total size of $R$ (denoted by $R_t$) on the disk and the size of the hash table $H_R$ (denoted by $h_R$) that contains the frequently-used part of $R$ in the memory. If the stream of updates $S$ is formulated using Zipf’s law with the exponent value being equal to 1, then the matching probability $p_N$ for stream $S$ with $H_R$ can be determined using Equation 4.

$$
p_N = \frac{\sum_{x=1}^{h_R} \frac{1}{x} \left( \frac{1}{h_R} \right) \ln \left( \frac{h_R}{R_t} \right)}{\ln \left( \frac{R_t}{\phi_N} \right)} = \frac{\ln (h_R)}{\ln (R_t)} (4)
$$

Now using Equation 4 the constant factors of change can be determined in $p_N$ by changing the values of $h_R$ and $R_t$ individually. This assumes that $p_N$ decreases by a constant factor $\phi_N$ if the value of $R_t$ is doubled, and increases by a constant factor $\psi_N$ if the value of $h_R$ is doubled. Knowing these constant factors the value of $w_N$ can be calculated. Consider a hypothesis

$$
p_N = R^y h_R^z (5)
$$

where $y$ and $z$ are unknown constants whose values need to be determined.

By doubling $R_t$, the matching probability $p_N$ decreases by a constant factor $\phi_N$, Equation 5 becomes:

$$
\phi_N p_N = (2R)^y h_R^z
$$

Dividing the above equation by Equation 5 we get

$$
y = \psi_N
$$

Similarly by doubling $h_R$ the matching probability $p_N$ increases by a constant factor $\psi_N$ therefore, Equation 5 can be written as:

$$
\psi_N p_N = R^y (2h_R)^z
$$

By dividing the above equation by Equation 5 we get

$$
z = \log_2(\psi_N)
$$

After putting the values of constants $y$ and $z$ in Equation 5 we get:

$$
p_N = R^{\psi_N \log_2(\phi_N)} h_R^{\log_2(\psi_N)}
$$

Figure 4: Tuning of Optimised X-HYBRIDJOIN using measurement approach

<table>
<thead>
<tr>
<th>Parameter</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk-based data</td>
<td></td>
</tr>
<tr>
<td>Size of disk-based relation $R$</td>
<td>0.5 million to 8 million tuples</td>
</tr>
<tr>
<td>Size of each tuple</td>
<td>120 bytes</td>
</tr>
<tr>
<td>Stream data</td>
<td></td>
</tr>
<tr>
<td>Size of each tuple</td>
<td>20 bytes</td>
</tr>
<tr>
<td>Size of each node in the queue</td>
<td>12 bytes</td>
</tr>
<tr>
<td>Benchmark</td>
<td>Zipp’s law</td>
</tr>
<tr>
<td>Characteristics</td>
<td>Bursty and self-similar</td>
</tr>
</tbody>
</table>
If \( S \) is the total number of stream tuples that are processed (through both the stream-probing and disk-probing phases) in \( N \) iterations, then \( w_N \) can be calculated using Equation 6:

\[
w_N = \frac{(R^\log_2(h_N)) h_R^{-\log_2(\psi_N)}) S}{N}
\]

Mathematical model to calculate \( w_S \): The second phase of the Optimised X-HYBRIDJOIN algorithm, also called the disk-probing phase, deals with the rest of the disk-based master data \( R' \) (where \( R = R_L - h_R \)), which occurs less frequently in the stream input as compared to that part which exists permanently in memory. The algorithm reads \( R' \) in partitions while the size of each partition is equal to the size of the disk buffer \( d \). As mentioned earlier, the daily market transactions typically formulate the Zipfian distribution, which means that matching probability for every next partition in \( R' \) is less than the previous one. Therefore, the matching probability for each partition is calculated by taking the summation over the discrete Zipfian distribution separately and then aggregating all of them as shown below.

\[
\sum_{x=h_R+1}^{h_R+d} 1 + \sum_{x=h_R+1}^{h_R+2d} 1 + \sum_{x=h_R+2d+1}^{h_R+3d} 1 + \cdots + \sum_{x=h_R+(n-1)d+1}^{h_R+nd} 1
\]

We simplify this to:

\[
\sum_{x=h_R+1}^{h_R+nd} 1 \Rightarrow \sum_{x=h_R+1}^{R_L} 1
\]

From this the average matching probability \( \overline{p}_S \) can be obtained in the disk probe phase, which is needed for calculating \( w_S \). Let \( n \) be the total number of partitions in \( R' \), then the average matching probability \( \overline{p}_S \) can be determined by dividing the above summation by \( n \). In the denominator, a similar normalization term to that used in Equation 4 is used.

\[
\overline{p}_S = \frac{1}{n} \sum_{x=h_R+1}^{R_L} 1 = \ln(R_L) - \ln(h_R + 1)
\]

To determine the effects of \( d \), \( h_R \) and \( R_L \) on \( \overline{p}_S \), the same number of steps is required as in the case of \( w_N \). If \( d \) is doubled then \( n \) will be halved in Equation 7 and therefore, the value of \( \overline{p}_S \) increases with a constant factor of \( \theta_S \). Similarly, if \( h_R \) and \( R_L \) are doubled one-by-one in Equation 7, the value of \( \overline{p}_S \) decreases with a constant factor of \( \psi_S \) and \( \phi_S \) respectively. A similar hypothesis is considered here as in Equation 5.

\[
\overline{p}_S = d^\theta_S h_R^\psi_S R_L^\phi_S
\]

The values for the constants \( x \), \( y \) and \( z \) in this case will be \( x = \log_2(\theta_S) \), \( y = \log_2(\phi_S) \) and \( z = \log_2(\phi_S) \) respectively. Therefore by replacing the parameters with constants, Equation 8 will become.

\[
\overline{p}_S = d^{\log_2(\theta_S)} h_R^{\log_2(\phi_S)} R_L^{\log_2(\phi_S)}
\]

If \( h_S \) are the number of stream tuples stored in the hash table then the average value for \( w_S \) can be calculated using Equation 9.

\[
w_S(average) = d^{\log_2(\theta_S)} h_R^{\log_2(\phi_S)} R_L^{\log_2(\phi_S)} h_S
\]

Once the values of \( w_N \) and \( w_S \) have been determined, the algorithm can be tuned using Equation 3.

4.4.4 Comparisons of both Tuning Approaches

In this section to validate our cost model, we compare the tuning results obtained through measurements with the tuning results that we calculated using the cost model.

Disk buffer: In this experiment we perform tuning of the disk buffer using both the measurement and the mathematical approaches. The tuning results of each approach are shown in Figure 5(a). From the figure it can be observed that the results in both cases are very similar with a deviation of only 0.38%.

Hash table \( H_R \): We also made the tuning comparisons for hash table \( H_R \) using both approaches. The experimental results in this case are shown in Figure 5(b). From the figure, the results in both cases are again closely related with a deviation of only 0.33%. This proves the accuracy of our cost model.

5 Experimental study

In this section we present a series of experimental results to support the proposed join algorithm. We
conducted our experiments in two dimensions. In Section 5.1 we compare the performance of Optimised X-HYBRIDJOIN with algorithms that are directly related to it. In Section 5.2 we compare the costs predicted by the cost model for the algorithm with the measured costs.

5.1 Performance evaluation

In the near-real-time data warehousing context the total allocated memory and the size of disk-based relation are the common parameters that can vary and directly affect the performance of the algorithm. Therefore, in our experiments we compare all algorithms by varying both parameters one-by-one.

Performance comparisons when the size of \( R \) varies: In this experiment we compare the performance of Optimised X-HYBRIDJOIN with other join algorithms. In our experiments we assume that the size of disk-based relation \( R \) varies exponentially while the total allocated memory is fixed for all values of \( R \). The performance results are shown in Figure 6(a). From the figure it is clear that for all settings of \( R \) the performance in the case of Optimised X-HYBRIDJOIN is significantly better than that of the other algorithms.

Performance comparisons for different memory budgets: In our second experiment we test the performance of all algorithms using different memory budgets by keeping the size of \( R \) fixed (2 million tuples). Figure 6(b) presents the comparisons of all approaches. From the figure, for all memory budgets, Optimised X-HYBRIDJOIN again performs significantly better than all the other approaches. In both scenarios the reason for improvement in performance is the use of an efficient architecture in Optimised X-HYBRIDJOIN. On the contrary, in X-HYBRIDJOIN the data structures used for some components are ineffective causing some unnecessary costs in processing the stream tuples and eventually it effects the performance of the algorithm negatively.

5.2 Cost validation

In the second part of our experiments we validate the cost model for the algorithm by comparing the predicted cost with the measured cost. Figure 7 presents the comparisons of cost model predictions and measurements for different memory settings. It can be observed from the figure that for each memory setting the predicted cost is close to the measured cost. This proves the accuracy of the cost model.

6 Conclusions and future work

In this paper we present a significant optimisation for a recently introduced stream-based join called X-HYBRIDJOIN (Extended Hybrid Join). This algorithm is designed to efficiently process non-uniformly distributed data as found in real-world applications. In our investigation we discover that the algorithm has some architectural limitations affecting its performance. Data structures used for some components such as the non-swappable part of the disk buffer, are not optimal, causing additional look-up cost. In addition, the algorithm stores a major part of the stream in memory that matches with the non-swappable part of the disk buffer which is unnecessary and generates extra costs for loading and unloading stream tuples into memory. On the basis of these observations we propose an optimised version of the existing X-HYBRIDJOIN called Optimised X-HYBRIDJOIN (Optimised Extended Hybrid
In the proposed algorithm, processing of tuples that match the swappable and non-swappable parts of the master are executed independently using efficient data structures. The stream that matches with the non-swappable part does not need to be stored in memory. This has two advantages: (a) It eliminates the additional costs required for loading and unloading the stream tuples into memory. (b) More stream tuples that are related to the swappable part can be accommodated in memory. We calculate the mathematical costs for our algorithm and tune the algorithm based on both measurement and cost model. To compare the performance with related algorithms we implemented prototypes of all approaches. Our experiments show that Optimised X-HYBRIDJOIN performs significantly better than the related approaches. We also provide the source code for our implementations.

In the future we plan to generalise our algorithm for other kinds of distributions. This will be particularly useful for markets that do not follow the 80/20 Rule. Additionally, the generalisation of the algorithm will remove the need for the disk-based relation to be sorted.

Source URL: The source of our implementations can be downloaded from the given URL. https://www.cs.auckland.ac.nz/research/groups/serg/source/

References


Scalable Motif Detection and Aggregation

Jens Dietrich  Catherine McCartin
School of Engineering and Advanced Technology
Massey University
Palmerston North, New Zealand
Email: {j.b.dietrich,c.m.mccartin}@massey.ac.nz

Abstract

Motif search in graphs has become a popular field of research in recent years, mainly motivated by applications in bioinformatics. Existing work has focused on simple motifs: small sets of vertices directly connected by edges. However, there are applications that require a more general concept of motif, where vertices are only indirectly connected by paths. The size of the solution space is a major limiting factor when dealing with this kind of motif. We try to address this challenge through motif instance aggregation. It turns out that effective, parallel algorithms can be found to compute instances of generalised motifs in large graphs.

To expedite the process, we have developed GUERY, a tool that can be used to define motifs and find motif instances, in graphs represented using the popular JUNG graph library [10]. GUERY consists of two parts - a simple domain specific language that can be used to define motifs, and a solver. The main strengths of GUERY are 1. support for motif instance aggregation, 2. generation of query result streams, as opposed to (very large) static sets of matching instances, 3. support for effective parallelisation in the evaluation of queries.

The examples used for validation originate from problems encountered when analysing the dependency graphs of object-oriented programs for instances of architectural antipatterns.

1 Introduction

The detection of motifs has become a popular field of research in recent years, driven by applications in bioinformatics. For example, Milo and colleagues [9] have attempted to study the structural design principles of different networks from biochemistry, neuroscience, ecology, and engineering, by detecting network motifs: patterns of interconnections occurring in these networks at numbers that are significantly higher than those in randomized networks. The motifs considered in [9] are small sets of interconnected vertices. Interconnected means that there are edges in the network directly linking the vertices participating in the motif. In other domains, similar structures have been studied. For instance, motifs are closely related to design patterns [7] - a concept widely used in software engineering to describe and communicate design. Use cases, such as pattern-based program comprehension, have led to research in design pattern formalisation [17], and the construction of tools to detect those patterns in network models representing the design of the software that is being analysed. Another class of patterns that have been studied in software engineering are architectural antipatterns - patterns representing design flaws in programs that have a negative effect on quality attributes of the respective programs. The classical architectural antipattern is the existence of a circular dependency between modules [16]. Several other patterns have been studied, including diamond inheritance [14, 15] and subtype knowledge [13]. These patterns can be defined with respect to the dependency graph, representing the high level design of the artefacts of the software and their relationships. Design patterns and antipatterns are similar to the motifs considered in [9], except that they do not assert that the participating vertices are directly connected. Instead, indirect connections via paths are permitted.

In [6] we presented results of analysing the dependency graphs of object-oriented programs for instances of architectural antipatterns. This analysis required, in part, the enumeration of instances of such patterns, often in very large graphs. In response to challenges encountered in our previous work, we have developed GUERY1, a tool that can be used to define motifs and find motif instances, in graphs represented using the popular JUNG graph library [10]. GUERY consists of two parts - a simple domain specific language that can be used to define motifs, and a solver. The main strengths of GUERY are 1. support for motif instance aggregation, 2. generation of query result streams, as opposed to (very large) static sets of matching instances, 3. support for effective parallelisation in the evaluation of queries. We consider that the techniques employed in the development of GUERY may be widely applicable to motif detection in a variety of contexts.

The rest of this paper is organised as follows. In section 2 we review related work. We formally define some concepts needed later in section 3, and discuss a domain specific language that can be used to define motifs in section 4. Next we introduce motif aggregation in section 5. The algorithms we have developed for motif detection are covered in section 6, and some experiments we have conducted in order to validate these algorithms are presented in section 7. A discussion of the results presented concludes our contribution.

1http://sites.google.com/site/gueryframework/
2 Related Work

One existing graph analysis tool closely aligned to our purpose of detecting generalised motifs in graphs is CrocoPat [3]. CrocoPat manipulates relations of any arity. Its query and manipulation language, RML, is a full programming language with syntax elements such as conditionals and loops, based on first-order predicate calculus. The implementation is based on binary decision diagrams (BDDs), well-known as a compact representation of large relations. CrocoPat queries are purely relational, each query returns facts for a certain predicate symbol. Paths cannot be directly represented in CrocoPat. However, reasoning about paths in CrocoPat is supported through the higher-order transitive closure predicate TC, and, in fact, each of the graph motifs used for validation (Section 7) in this paper can be described by queries expressible in CrocoPat. Based on experimental evidence, CrocoPat is rather less efficient memory-wise than GUERY. More importantly, however, the underlying design of CrocoPat does not lend itself to parallelisation, nor do current implementations of CrocoPat support the generation of query result streams.

GRoQL (Graph Repository Query Language) is another graph query language built to extract information from TGraphs. TGraphs are typed, attributed, directed, and ordered graphs which conform to TGraph schemas. In turn, schemas are metamodels conforming to the metaschema of the graph modeling language grUML, a subset of UML class diagrams. Similar to SQL queries in relational databases, a typical GRoQL query specifies the range of some free variables, poses some conditions on these variables and describes the desired output. The conditions are specified using GRAL (Graph Specification Language), a predicate language based on Z. A central concept in GRoQL are regular path expressions. A regular path expression describes a path through a graph based on edge types and edge direction, and using basic regular expression syntax. A path expression can be used to test if pairs of nodes are connected via the specific path, to denote the set of vertices which can be reached from a starting vertex via the path, or to denote the set of vertices from which a particular vertex can be reached via the path. Similar functionality is incorporated into GUERY, but without the explicit regular expression syntax. In contrast to GUERY, currently available implementations of GRoQL do not support parallelisation in the evaluation of queries, nor do they support generation of query result streams.

SPARQL [12] supports a standardized query language and data access protocol for the semantic web. SPARQL is defined in terms of the W3C’s RDF data model and will work for any data source that can be mapped into RDF. SPARQL is an SQL-like language whose features include basic conjunctive patterns, value filters, optional patterns, and pattern disjunction. It works primarily via constraints on single vertices. SPARQL is considered a key semantic web technology, but it is insufficient for our purposes, since, in contrast to the tools introduced above, it does not support path constraints (transitive closure).

Several algorithms have been proposed for graph pattern matching [4], [20]. Our approach is more expressive as it supports the use of expressions on both vertex and edge labels (instead of just matching vertex labels), length constraints on individual connections (in [20], only static path length constraints that apply to all connections are considered), and aggregation clauses.

3 Motifs and Motif Instances

In this section we formally introduce motifs and related concepts.

Definition 1 (Path) Let \( G = (V,E) \) be a directed graph consisting of a set of vertices \( V \) and a set of directed edges (arcs) \( E \) (we use the term edge to denote a directed edge throughout this work.) A path is a finite sequence of edges \( (e_1, \ldots, e_n) \) such that for adjacent edges \( e_i \) and \( e_{i+1} \) the target vertex of \( e_i \) is the source vertex of \( e_{i+1} \). \( SEQ(E) \) is the set of all paths that can be constructed from given set of edges \( E \). For a given path \( p \), \( \text{start}(p) \) denotes the first vertex (source vertex of \( e_1 \)), \( \text{end}(p) \) the last vertex (target vertex of \( e_n \)) and \( \text{length}(p) \) the number of edges in the path.

We will consider paths of length 0 - these are paths consisting of a single vertex and no edge. This single vertex is both the start and the end vertex of the empty path. A regular path is a path where any edge occurs only once in the sequence defining the path. We use \( SEQ_{reg}(E) \) to denote the set of all regular paths constructible from \( E \). Even though \( E \) is finite, \( SEQ(E) \) is possibly infinite. Therefore, we will often restrict our investigation to regular paths.

Definition 2 (Motif) Let \( G = (V,E) \) be a directed graph consisting of a set of vertices \( V \) and a set of directed edges \( E \). A motif over \( G \) is a structure \( M = (VR,PR,s,t,CV,C_P) \) consisting of:

1. A set of vertex roles \( VR \).
2. A set of path roles \( PR \).
3. Two functions \( s: PR \rightarrow VR \) and \( t: PR \rightarrow VR \), associating path roles with vertex roles.
4. A set of vertex constraints \( CV \), consisting of \( n \)-ary predicates between vertex tuples, \( CV \subseteq \times_{i=1..n} V \), where \( n \) is the cardinality of \( VR \).
5. A set of path constraints \( C_P \), consisting of \( n \)-ary predicates between tuples of paths, \( CP \subseteq \times_{i=1..n} SEQ(E) \), where \( n \) is the cardinality of \( PR \).

Intuitively, constraints restrict the sets of possible vertex and path assignments. While vertex constraints are always defined with respect to vertex labels, we consider two types of path constraints: cardinality constraints that restrict the length of permitted paths, and constraints defined with respect to labels defined on the edges in the path.

Usually, the vertex roles in motifs are cohesive. By this, we mean that they are linked via path roles. This can be defined as follows:

Definition 3 (Motif graph and connected motifs) Let \( M = (VR,PR,s,t,CV,C_P) \) be a motif over a directed graph. The motif graph \( G_M = (V_M,E_M) \) is then defined as follows: \( V_M := VR \), \( E_M := PR \), for each \( pr \in PR \), \( \text{start}(pr) := s(pr) \) and \( \text{end}(pr) := t(pr) \). \( M \) is called connected iff \( G_M \) is weakly connected.

A binding associates vertex roles with vertices and path roles with sequences of edges. The \( s \) and \( t \) functions associate path and vertex roles: path roles connect source and target vertex roles. The conditions define a structural homomorphism between the motif and the target graph, as shown in figure 1.
Definition 4 (Binding) Let \( G = (V,E) \) be a directed graph and \( M = (V,R,P,r,s,t,C_V,C_P) \) a motif. A binding is a pair of functions \( \langle \text{inst}_V, \text{inst}_P \rangle \), where \( \text{inst}_V : V \to V \) and \( \text{inst}_P : P \to \mathcal{S}\mathcal{E}(E) \), such that \( \text{inst}_V(s(pr)) = \text{start}(\text{inst}_P(pr)) \) and \( \text{inst}_V(t(pr)) = \text{end}(\text{inst}_P(pr)) \).

A binding does not necessarily satisfy the sets of constraints that are part of the motif definition. If it does, we call it valid.

Definition 5 (Valid binding) Let \( G = (V,E) \) be a directed graph, \( M = (V,R,P,r,s,t,C_V,C_P) \) and \( \text{bind} = (\text{inst}_V,\text{inst}_P) \) a binding. \( \text{bind} \) is called valid if the following two conditions are fulfilled:

1. \( (\text{inst}_V(v_1),..,\text{inst}_V(v_n)) \in C_V \) for all vertex constraints \( C_V \)
2. \( (\text{inst}_P(p_1),..,\text{inst}_P(p_n)) \in C_P \) for all path constraints \( C_P \)

The question arises whether valid bindings are suitable representations for motif instances. The main problem is that path roles can be bound to arbitrary paths. Since the set of paths can be infinite even though the set of edges is finite, it is reasonable to restrict our consideration to bindings where path roles are bound to regular paths. We can call these bindings regular.

For many practical purposes, using regular bindings to represent motif instances is still not scalable. This is due to the fact that the set of possible paths is still too large. If \( |E| \) is the number of edges in the graph, there are at worst case scenario \( 2^{|E|} \) sets of edges that can form paths. As the order also matters, all permutations of these sets have to be considered. This implies that there are up to \( \sum_{i=1}^{|E|} \frac{|E|^i}{i!} \) many possible paths to consider. Using the Taylor series expansions of the exponential function, it can easily be shown that this converges to \( |E|e \). If the motif contains \( n \) path roles, then there are up to \( \sum_{i=1}^n \frac{|E|^i}{i!} \) different assignments of paths to path roles.

For instance, consider the simple graph shown in figure 2, and a simple motif that consists of one vertex role and a path role that connects vertices to themselves. The cardinality constraint for the path is \([0,\ast] \) - paths of arbitrary length are permitted.

The formula constrained above can be calculated as follows:

\[
\frac{3^2}{2} + \frac{3^3}{3!} + \frac{3^4}{4!} + \frac{3^5}{5!} = 16 \approx 3e = 16.309691...
\]

Figure 2: One node graph

A possible solution is to identify as equivalent bindings that define the same vertex bindings, but potentially different path bindings. For many applications this is sufficient: users want to compute one or more paths without referring to other vertices, but not necessarily the complete set. More precisely we define a motif instance as follows:

Definition 6 (Motif Instance) Let \( G = (V,E) \) be a directed graph, \( M = (V,R,P,r,s,t,C_V,C_P) \) a motif and \( B = \{\langle \text{inst}_V,\text{inst}_P \rangle \} \) the set of all valid bindings. Then the relationship \( \sim \subseteq B \times B \) is defined as follows:

\( \langle \text{inst}_V^1,\text{inst}_P^1 \rangle \sim \langle \text{inst}_V^2,\text{inst}_P^2 \rangle \) if \( \text{inst}_V^1 = \text{inst}_V^2 \). 

This definition supports one-line comments (line 1), and automatic naming of motifs (line 2). Following the select keyword, vertex roles are defined (line 3). Path roles are defined following the connected by keyword (line 4). Each path role is defined by a unique name, followed by the source and target vertex roles separated by "\( \ast \)" and the definition of cardinality constraints restricting the length of the instantiating paths. This is done by pairs of values defining the minimum and the maximum value, with "\( \ast \)" representing unbound. If no cardinality constraints are defined, the default \( [1,\ast] \) is used.

The motif has several constraints (line 5). The constraints are string literals, interpreted using the existing object-oriented MVEL expression language [1]. The MVEL expression engine can compile these constraints dynamically into optimised Java Byte code, making the evaluation of expressions very fast. The expressions refer to either edges or vertices and their labels. For instance, the expression "inherits.type=="extends" evaluates to true if the type label for each edge in a path instantiating the inherits path role equals "extends". Note that labels can be complex structures. For instance, it is possible to refer to the first two characters of the type label using the expression "inherits.type.substring(2)".

The motif defined in listing 2 represents a circular dependency between name spaces. Name spaces are used in object-oriented programming to group related classes together. Circular dependencies between name spaces are widely regarded as design flaws. This motif describes a circular dependency through a path that starts at a vertex within a name space (\( \text{in1} \)).
then goes through vertices in other name spaces (out1 and out2), returning to a vertex in the original name space (in2). Constraints are now defined on vertices with respect to their name space labels (line 4). The cardinality of the out and in path roles is set to 1, indicating that those paths are to be instantiated by a single edge. Those edges are exiting (out) and entering (in) the respective name space. Note that the cardinality constraint on the part of the path that connects the out vertices is [0, *]. This means that an empty path is possible, and therefore the motif can be instantiated by a triangular pattern consisting of three vertices v_{in1}, v_{in2} and v_{out}, with edges connecting v_{in1} to v_{out} and v_{out} to v_{in2}, respectively.

The third example (listing 3) represents a pattern where a client (directly) references a service specification, but also has a (possibly indirect) reference to a particular service implementation. This obstructs the dynamic swapping of service implementations that is an important feature in modern software design. In this motif, constraints are defined with respect to both vertex (line 3) and path roles (line 5). Not all path roles have length constraints, in this case the default [1, *] is used.

5 Motif Aggregation

In our previous work [6], we have conducted an empirical study to investigate just how common antipatterns that can be represented by motifs are, in dependency graphs extracted from real world software. It turns out that the number of instances is generally very large. This makes our approach, as outlined so far, difficult to use in this target domain: engineers analysing the quality of software are overwhelmed by the large number of instances of a particular antipattern. However, it is very often the case that many motif instances are very similar to other instances, and are perceived by users as non-significant variations. This raises the following questions:

1. How can similarity of motif instances be formally defined?
2. How can similarity of instances be included in pattern definitions?
3. Can algorithms to find motif instances take advantage of instance similarity?

For all three motifs discussed above, the introduction of instance similarity makes sense for a software engineer who wants to use these motifs. For the subtype knowledge motif (listing 1), an engineer is often only interested in which super types reference their own subtypes. Hence, different instances binding the super role to the same vertex can be considered similar. For the circular dependency motif (listing 2), an engineer is often only interested in finding the packages that have circular dependencies. Here, different instances binding the in1 role to vertices within the same name space can be considered similar. Finally, for the abstraction without decoupling motif (listing 3), instances can be considered similar if they have identical bindings for client and the service roles, but not necessarily for the impl role.

This concept of similarity can be expressed by aggregation functions. Functions are applied to motif instances, and instances are considered similar if those functions yield the same value. This similarity relation is an equivalence relation, known as the equivalence kernel of the respective function(s).

Definition 7 (Aggregation Function) Let G = (V, E) be a directed graph, M a motif and INST the set of instances of M in G. An aggregation function is a mapping j : INST → VALUE that maps instances to values in some domain.

For a given set of aggregation functions \{f_i : INST → VALUE\}, we define \(\simeq \subseteq INST \times INST\) as follows: \(\text{inst}_1 \simeq \text{inst}_2\) if and only if \(f_i(\text{inst}_1) = f_i(\text{inst}_2)\) for all \(f_i\).

GUERY/DSL supports the definition of simple aggregation functions in the group by clause. In the subtype knowledge motif (listing 1), there is one function that maps a motif instance to the vertex instantiating the super role (line 6). The circular dependency motif (listing 2) also uses one aggregation function (line 6). This function maps the instance to the name space of the vertex instantiating the in1 role. Finally, the abstraction without decoupling motif (listing 3) uses two aggregation functions (line 6) - instances are considered similar if both the client and the service role are mapped to the same vertices.
6 Motif Detection

6.1 Main Algorithm

In this section we discuss some of the algorithms used by QUERY to find motifs in graphs. The algorithms are applicable to connected motifs.

The main algorithm (Algorithm 2) is based on depth-first constraint resolution. Bindings associating vertex roles with vertices and path roles with paths are maintained in two maps, `vbindings` and `pbindings`, respectively. The maps also keep track of positions for individual key-value pairs, and support a `remove` function that can be used to remove associations for a given position. The `bind` function adds new associations to a map, while the `lookup` function is used to query maps for values associated with a given key.

The algorithm first chooses an arbitrary initial role, \(r_0\), with which analysis will start, and then uses the `schedule` function to arrange vertex constraints and path roles into an ordered list. Since the motif is connected, motif instances can be discovered by starting from bindings of an initial vertex role. Starting from this role, all other roles can be reached by either following the path roles directly, or by following them in reverse direction. This step can be seen as a query optimisation. It can be done statically, before the actual graph is analysed, and, therefore, does not depend on the size and the complexity of the actual graph. The main purpose is to avoid iterating over \(V \times V\) when instantiating path roles. Instead, we can always assume that, for each path role, either the source or the target vertex role is already instantiated.

The `schedule` function will also ensure that vertex constraints are evaluated immediately after bindings for the vertex roles referenced in the constraints have been added. Evaluating vertex constraints is usually much cheaper than graph traversal, and can be used to trim the search space. The complete algorithm for the `schedule` function is given in Algorithm 1. The comma operator denotes list concatenation: a new list is created by appending an element to the old list. The function \(\text{roles} : CV \rightarrow 2^{VR}\) maps vertex constraints to the set of vertex roles referenced in the respective constraint.

In the main algorithm, exploration starts using the initial vertex role. For each vertex in \(V\), the initial role is bound to this vertex (line 5), and the binding is stored in the `vbindings` map. The main analysis step is done in the recursive `resolve` procedure. The procedure uses a counter to keep track of the number of constraints that have been checked. If this number equals the number of rules, an instance consisting of vertex and path bindings has been found and can be reported (line 10). Being “reported” does not necessarily mean being added to a list of results. In many application scenarios, it is more desirable to notify applications that the algorithm has found another motif instance, but to leave it to the application to decide what to do with this information. This approach uses the observer pattern [7], and is very memory efficient. Instead of producing a set of instances, an application implementing this algorithm can generate query result streams.

In each step of the algorithm, a rule (either a vertex constraint or a path role) is resolved. If the next rule is a vertex constraint, it is instantiated using the current bindings, and the resulting expression is evaluated (line 17). The scheduling step ensures that the bindings for all vertex roles used in the constraint are available. If the next rule is a path role, the algorithm must distinguish three cases:

1. Both the start and the target vertex role of the path role are already bound.
2. Only the source vertex role is bound.
3. Only the target vertex role is bound.
Since we assume that the motif is connected, scheduling can ensure that it is not possible that both source and target roles are without binding. In the first case, if a path exists that connects source and target and satisfies the additional path constraints, it is used to instantiate the respective path role. Such a path can be discovered by using a standard shortest path algorithm. For the other two cases, all incoming (outgoing) paths satisfying the path constraints are computed, and their source (target) is bound to the respective path role. We do not explicitly specify the three graph traversal functions used (find_incoming_path, find_outgoing_path and find_outgoing_path), but usually breadth-first search is used here. The GUERY solver supports “pluggable” graph traversal strategies. The is discussed in more detail below in section 6.4.

6.2 Using Aggregation

If only instance classes, modulo aggregation functions, are to be computed, Algorithm 2 can be significantly optimised. The key idea is to extract the vertex roles used in the aggregation functions, and to try to resolve them first. This can be easily achieved by first selecting an initial vertex role from the set of roles used in aggregation functions, and then sorting the path roles in the scheduling function so that the motif graph is traversed in a way that visits those vertex roles first. The scheduling function also has to record the position in the rule list where all those vertex roles will have bindings. We refer to this number as the aggregation threshold. Once this is done, the back tracking section of the main algorithm can be rewritten to support back jumping: instead of reducing the position count by one, the position can be moved to the position just above the aggregation threshold, and all bindings below this point can be removed. This will make sure that the next instance computed will have a different binding for at least one of the vertex roles that occurs in the aggregation functions.

Note that this is no guarantee that only one instances per equivalence class will be computed. An example is the circular dependency motif (listing 2). Even if the in1 role is bound to different vertices, these vertices could still be in the same name space. However, the search space is significantly reduced. To consolidate results into classes, an extra reduce step may be necessary. A sufficient condition that no reduce step is needed is that all aggregation functions are injective. This is the case in the examples STK (listing 1) and AWD (listing 3) - in these motifs, the aggregation functions are instances of the identity function, mapping vertices to themselves.

6.3 Parallelisation

Parallelising Algorithm 2 is straightforward: search for different initial bindings can be executed in parallel. Using back jumping to optimise the search for classes of instances is still possible. The detailed algorithm is shown in Algorithm 3. The set of vertices is added to an agenda, and retrieved by threads as long as the agenda is not empty. Note that the query and element retrieval from the agenda (lines 14-15) must be synchronised (protected by a lock). Each thread has its own binding maps and position counter that are not shared with other threads.

Algorithm 3 For a given directed graph $G = (V, E)$ and a motif $M = (VR, PR, s, t, C_V, C_P)$, find all motif instances using $N$ threads

1. choose initial role $r_0 \in VR$
2. agenda = empty stack
3. rules := schedule(M, r_0)
4. count := 0
5. for all vertex in $V$ do
6. push(agenda, vertex)
7. while count < $N$ do
8. position := 0
9. vbinding = empty map
10. pbinding = empty map
11. thread = create_thread(job)
12. start(thread)
13. routine job
14. while |agenda| > 0 do
15. next_vertex := pop(agenda)
16. bind(vbinding, $r_0 \rightarrow$ vertex, 0)
17. resolve
18. end routine

6.4 Path Discovery

The main algorithm uses a function to find incoming and outgoing paths for a given vertex. The graph traversal strategy used to compute these paths can be configured, and is designed as a pluggable module in the implementation. By default, paths are computed through online computation using a simple breadth-first traversal strategy. We have also implemented a more advanced strategy that caches paths. Naive caching of reachability information is not scalable as the size of the memory required to store reachability information is $|V| \times |V|$. However, we have implemented a strategy that precomputes reachability between vertices and compresses the cache in two stages:

1. Compute the graph $G_{SCC}$ consisting of the strongly connected components and their relationships using Tarjan’s algorithm [18].
2. Compute reachability for the (acyclic) graph $G_{SCC}$ and store it using chain compression [8].

This strategy adds some initial overhead to the computation but can significantly improve the overall speed of the computation.

7 Validation

7.1 Implementation

We have implemented the algorithms discussed earlier as a Java class library “GUERY”. The library is based on JUNG2 [10], but uses special super classes for representing vertices and edges, respectively. The main purpose of these is to enable fast graph traversal, through vertices having direct references to incoming and outgoing edges. The correctness of the implementation is established through comprehensive test suites with high coverage.

We have used GUERY to validate thee aspects of the algorithms: the effectiveness of aggregation, the computation of partial result sets, and the benefits of using the parallel version of the algorithm with multi-core processors.

7.2 Aggregation

The data we have used for validation are dependency graphs extracted from compiled Java programs by
Algorithm 2 (Motif Detection) For a given directed graph $G = (V, E)$ and a motif $M = (VR, PR, s, t, CV, CP)$, find all motif instances

1: choose initial role $r_0 \in VR$
2: position := 0
3: rules := schedule($M, r_0$)
4: for all vertex in $V$ do
5:   bind($v$bindings, $r_0 \rightarrow$ vertex, 0)
6:   resolve
7: exit
8: routine resolve
9: if position = |rules| then
10: signal(“instance found”, $v$bindings, $p$bindings)
11: return
12: position := position + 1
13: rule = rules[position]
14: if rule $\in CV$ then
15:   instantiated_constr := instantiate(rule, $v$bindings)
16:   if check(instantiated_constr) then
17:   resolve
18: else if rule $\in PR$ then
19:   source := lookup($v$bindings, $s$(rule))
20:   target := lookup($v$bindings, $t$(rule))
21:   if exists(source) $\land$ exists(target) then
22:     path := find_first_path(source, target, $CP$)
23:     bind($p$bindings, $role$(rule) $\rightarrow$ path, position)
24:     resolve
25: else if exists(source) $\land$ $\neg$ exists(target) then
26:   paths := find_outgoing_paths(source, $CP$)
27:   for all path in paths do
28:     bind($v$bindings, $t$(rule) $\rightarrow$ end(path), position)
29:     bind($p$bindings, $role$(rule) $\rightarrow$ path, position)
30:     resolve
31: else if exists(target) $\land$ $\neg$ exists(source) then
32:   paths := find_incoming_paths(target, $CP$)
33:   for all path in paths do
34:     bind($v$bindings, $s$(rule) $\rightarrow$ start(path), position)
35:     bind($p$bindings, $role$(rule) $\rightarrow$ path, position)
36:     resolve
37: {backtrack}
38: remove($v$bindings, position)
39: remove($p$bindings, position)
40: position := position - 1
41: end routine
means of byte code analysis. The programs used are the ten largest programs (measured by the number of vertices) from the Qualitas corpus [19], a collection of programs often used in experimental software engineering. The motifs used in the experiments are the motifs discussed earlier: AWD (listing 3), CD (listing 2) and STK (listing 1). Table 1 shows the respective graphs, identified by the names of the programs, and the number of vertices and edges. Table 2 shows the number of instances, and equivalence classes of instances modulo aggregation functions, found in those programs. Using classes instead of instances significantly reduces the search space. The average number of instances per class is 18.60 for AWD, 9759.25 for CD and 6.94 for STK. The number is particularly high for CD, the motif that uses a non-injective aggregation function.

<table>
<thead>
<tr>
<th>graph (program)</th>
<th>vertices</th>
<th>edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>azureus-3.1.1.0.jar</td>
<td>6444</td>
<td>35392</td>
</tr>
<tr>
<td>jchempaint-2.0.12.jar</td>
<td>3342</td>
<td>17335</td>
</tr>
<tr>
<td>jrefactory-2.9.19.jar</td>
<td>2776</td>
<td>16007</td>
</tr>
<tr>
<td>rssowl-1.2.jar</td>
<td>2405</td>
<td>14249</td>
</tr>
<tr>
<td>jruby-1.0.1.jar</td>
<td>2093</td>
<td>11016</td>
</tr>
<tr>
<td>drjava-20050814.jar</td>
<td>2078</td>
<td>11916</td>
</tr>
<tr>
<td>weka-3.5.8.jar</td>
<td>2046</td>
<td>12230</td>
</tr>
<tr>
<td>sandmark-3.4.jar</td>
<td>1923</td>
<td>13574</td>
</tr>
<tr>
<td>jopen-4.9.jar</td>
<td>1779</td>
<td>9936</td>
</tr>
<tr>
<td>hibernate-3.3.1.jar</td>
<td>1700</td>
<td>10093</td>
</tr>
</tbody>
</table>

Table 1: Size of graphs in data set

7.3 Stream Processing
To validate the benefits of the observer based API, we have measured the time needed to compute the first 10 instances and classes for the three motifs AWD, CD and STK and the first three graphs from our data set. For this experiment, we used a PowerMac with a Intel Core 2 2.8 GHz Duo processor, the Java HotSpot(TM) 64-Bit Server VM (build 14.3-b01-101, mixed mode), and a solver configured to use two threads. The results are shown in table 3. The first 10 results are usually returned in under 1 s and can be presented to the user for analysis. The main advantage is that instances are produced faster than users can analyse them, and that users do not have to wait for the entire set of instances to be computed.

Note that the computation of the first 10 classes is slower than the computation of the first 10 instances. In order to compute classes, the solver must traverse deeper branches of the derivation tree, starting with the nodes to where the solver back jumped.

7.4 Parallelisation
To assess the effectiveness of the parallel algorithm, we computed all instances and variants for the largest graph in the data set, extracted from azureus-3.1.1.0.jar byte code. We used a computer with two Intel Xeon E5440@2.83GHz quad core processors, using the Java SE 1.6.0_16-b-1 Linux 64-bit virtual machine. We executed the queries (motif searches) using different numbers of threads. The respective performance data are shown in figures 3 (for AWD), 4 (for CD) and 5 (for STK), respectively. The algorithm scales very well when using multi core processors, but performance flattens out if the number of threads used goes beyond 8 and even slightly increases for larger numbers of threads as the effects of the overhead related to inter-thread synchronisation kick in.
1. the use of aggregation that can be used to sig-
map and reduce [5] style algorithm. Further research is necessary to explore this.

References

Optimal $k$-Constraint Coverage Queries on Spatial Objects

Chuanfei Xu  Yanqiu Wang  Yu Gu  Shukuan Lin  Ge Yu

College of Information Science and Engineering,
Northeastern University,
PO Box 135, Shenyang, China 110819
Email: {xuchuanfei, wangyanqiu, guyu, linshukuan, yuge}@ise.neu.edu.cn

Abstract
Given a set of spatial objects, our task is to assign all the objects to the minimum number of service sites and to find the regions for building these service sites. Each service site has a coverage region (i.e., an area of service) and a capacity (i.e., a maximum number of objects it can serve, called $k$-constraint). The service sites can provide service for objects located within the coverage regions. We propose a novel kind of spatial queries, called Optimal $k$-Constraint Coverage (OCC) queries. An OCC query returns some feasible regions such that setting up the minimum number of service sites within these regions will guarantee that all the spatial objects can be served. Furthermore, an optimal coverage scheme to assign the objects to these service sites is retrieved by this query as well. Due to the capacity constraints, objects located within the coverage region of a service site may not be assigned to one service site. Therefore, the cost of searching an optimal coverage over all possible coverage schemes becomes prohibitive. To answer OCC queries efficiently, we devise a general query framework, which provides two solutions to cope with OCC query processing. The naive solution only returns a local optimum without insuring the minimum number of service sites. To improve it, we devise Optimal Coverage Algorithm (Opt-C) to retrieve the optimal coverage scheme. The performance of the proposed methods is demonstrated by the extensive experiments with both synthetic and real datasets.

Keywords: Optimal Coverage; Minimum number; $k$-Constraint; Opt-C Algorithm

1 Introduction
Optimal location queries have gained much research attention (Zhang et al. 2006, Cabello et al. 2005, Xiao et al. 2011) in many real-world applications. Given a set of objects, an optimal location query returns the "best location" such that setting up a service site at this location guarantees the maximum number of objects by proximity. In some cases, the "best location" is denoted by a region (Wong et al. 2009, Zhou et al. 2011) in which any point can be an optimal location.

For existing optimal location query techniques, however, some significant improvements are required when we consider building multiple service sites. For example, we wish to build some service sites to provide assistance for people in the disaster area. To aid people on time, the distance between each person and his/her assigned service site must be not more than a given distance $r$. For each site, the medical supplies are prepared for at most $k$ people. To minimize the cost, we require a query which can return some regions such that setting up the minimum number of service sites would provide aid for all the persons.

The objective of the above problem is to investigate serving all the objects (people) with the minimum number of service sites. Compared to optimal location queries, this problem equals to maximizing the average number of served objects for each service site. In addition, the spatial assignement technique has also attracted much research attention (Leong et al. 2008, 2010) in several application domains. This optimized spatial assignment desires an optimal assignment between objects and a set of given service sites. It focuses on minimizing the cost between objects and fixed number of service sites while is unable to minimize the number of service sites.

For our problem, the coverage regions of service sites may have irregular shapes. For ease of presentation, we assume that the coverage region of any service site $p$ is a circular disk with the center at $p$ and the radius $r$. Moreover, $p$ can provide service for arbitrary $k$ objects within its coverage region. Figure 1(a) illustrates that two service sites $p_1$ and $p_2$ serve a set of spatial objects $o_1, o_2, o_3, o_4, o_5, o_6$. Suppose that the capacity of $p_1$ and $p_2$ ($p_1, k$ and $p_2, k$) are both set as 4. As shown in the figure, $o_1, o_2, o_3$ can be assigned to $p_1$ and $o_4, o_5, o_6$ can be assigned to $p_2$. Another available attempt to solve this problem is illustrated in Figure 1(b). In this solution, the service site $p_3$ serves the maximum number of objects, which is a result of the optimal location query. However, it is not the optimal coverage for all the objects with the lowest number of service sites since other two service sites $p_4$ and $p_5$ must be set up to serve $o_1$ and $o_5$, respectively.

Aiming to solve the above variant of optimal location query problem, we propose a novel kind of spatial queries called Optimal $k$-Constraint Coverage (OCC) queries which can retrieve a series of regions such that setting up the minimum number of service sites within the regions would guarantee to provide service for all the given objects, where each service site subjects to the capacity constraint (i.e., a maximum number of objects it can serve, called $k$-constraint). Given a set of spatial objects $O$, OCC queries are particularly...
useful for the optimal assignment resource that ensures to maximize the average number of objects covered by each service site. Hence, this type of queries are important to a wide range of applications, such as Location-Based Services, the best locations choice and so on. In these applications, the service sites can be set up in the whole space \( \mathcal{U} \) instead of some candidate points. In essence, there is a common problem such as “where are the best regions to build the minimum number of service sites” in these applications. Note that the results of OCC queries are the regions for building the service sites and an optimal coverage scheme to assign the spatial objects to these service sites located at any points within these regions.

In Figure 1(c), any two service sites can be built at any locations of the two shadow regions to serve \( \{o_1, o_2, o_3\} \) and \( \{o_4, o_5, o_6\} \) respectively, where the capacity constraint of each service site is set as 4. Since the total number of the objects is 6, at least two sites are needed to serve \( \{o_1, o_2, o_3, o_4, o_5, o_6\} \). The shadow regions are the results of the OCC query over these objects. Besides, there are some other coverage schemes that all the objects are assigned to two service sites (e.g., \( \{o_1, o_2, o_3, o_4\} \) are assigned to the service site set up in shadow region A, and \( \{o_4, o_5\} \) are assigned to the service site set up in shadow region B). Our task is to return an optimal coverage scheme over all possible coverage schemes which denote assignments between objects and service sites. Along with the increasing number of served objects, the number of available schemes may become prohibitive. Therefore, the OCC query evaluation raises serious practicality concerns for the volume of objects in realistic settings.

This work focuses on OCC query processing, which has two main challenges. (i) How to retrieve the regions such that setting up service sites within these regions will guarantee to provide service for all given objects. (ii) How to choose an optimal coverage scheme that the spatial objects can be assigned to the minimum number of service sites while insuring the capacity constraints. Aiming at the above challenges, we propose a general query framework for answering the OCC queries simply and elegantly. Consider a set of spatial objects \( O=\{o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8\} \). We will illustrate the process of OCC query on these objects in the rest of the paper. In detail, our contributions are summarized as follows.

- We formally define a novel kind of spatial queries, called Optimal \( k \)-Constraint Coverage (OCC) queries, which can choose an optimal coverage scheme that aims to assign all the given objects to the minimum number of service sites, and retrieve some regions to build these service sites for serving all given objects while insuring the capacity constraints of service sites.
- To answer OCC queries efficiently, we first propose a local optimum algorithm to serve objects located in each coverage set with the lowest number of service sites.
- To improve the local optimum algorithm, we present another approach for choosing an optimal coverage to assign all objects to the minimum number of service sites, which guarantees that this scheme is global optimum. During this process, refinement methods are proposed to further improve the efficiency.
- We evaluate the performance of our methods through extensive experiments with real and synthetic datasets.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 defines the OCC query processing formally. Section 4 describes the query framework for answering OCC queries. Section 5 provides some efficient and effective methods for answering our queries in details. Section 6 gives the experimental results and Section 7 concludes this paper.

2 RELATED WORK

2.1 Spatial Queries

There is a large volume of previous works (Prasad et al. 1998, Chen et al. 2005, Beskales et al. 2008, Lian & Chen 2009, Ishikawa et al. 2009, Deoug et al. 2009, Nutanong et al. 2008) devoting to spatial query processing in last few years. Especially, this trend has led to the development of spatial database management for spatial queries. Joao Rocha Junior et al. (Rocha-Junior et al. 2010) address the top-\( k \) spatial preference query which returns a ranked set of the \( k \) best objects based on the scores of feature objects in their spatial neighborhood. They map the pairs of data and feature objects to the distance-score space, which enables to identify the minimum subset of pairs necessary to answer any ranked spatial preference query. They also improve the efficiency by avoiding examining the spatial neighborhood of the data objects during query execution. Zhenjie Zhang et al. (Zhang et al. 2008) address the continuous \( k \)-means problem, where a \( k \)-means query returns \( k \) points in space, while guaranteeing that the average squared distance between each point in \( \mathcal{P} \) and its nearest center is minimized. A novel algorithm is proposed to reduce the computation and communication costs, and a threshold is assigned to each moving object such that the object sends a location update only when it crosses the range boundary. Because these works do not consider the optimal locations (or regions) to build service sites, they are not suitable for our problem.
Furthermore, some other work focuses on finding a region such that building a new service site in this region guarantees the maximum number of objects by proximity. Sergio Cabello et al. (Cabello et al. 2005) address the RNN facility location problems which is actually the MaxBRNN problem when the optimization criteria is maximizing the number of potential customers for the new facility. This work is solved for Euclidean space. In this space, Raymond Chi-Wing Wong et al. (Wong et al. 2009) present the algorithm called Fichromatic reverse nearest neighbors problem, which focuses on finding an optimal region that maximizes the size of BRNNs and an efficient algorithm called Maxoverlap is proposed for that. They define two concepts called consistent region and maximal consistent region to efficiently find an optimal region that maximizes the size of customers by proximity; and it uses the region-to-point transformation to solve the MaxBRNN problem. Considering that a customer may use the service site which is any of his/her k nearest service site, (Zhou et al. 2011) extend the MaxBRNN to the MaxBrkNN which finds an optimal region that deploying a service site in this region attracts the maximum number of customers who would consider the site as one of their k nearest service sites.

Optimal location queries are a significant sort of spatial queries and several query approaches have been proposed for retrieving the optimal locations with respect to different semantics. Du Yang et al. (Du et al. 2005) define and investigate the optimal location problem. They focus on solving the problem in the Manhattan distance space. They retrieve objects of interest in some given order and then use a plane-sweep algorithm to identify an optimal location. Furthermore, Xia et al. (Xia et al. 2005) examine a related problem of finding the top-k most influential sites among a given set of service sites. Note that in the top-t influential sites problem, the search space is limited. This makes the top-t influential sites problem very different from the optimal location problem.

The solutions for MaxBRNN problem and optimal location queries cannot be applied to find optimal regions for multiple service sites and do not consider minimizing the number of required service sites.

### 2.2 Spatial Assignment

The spatial assignment problem also has attracted much research attention (Leong et al. 2008, 2010) for the optimal assignment domain. Leong Hou U (Leong et al. 2008) consider the capacity constrained assignment, where each service provider can serve at most k customers, when the total size of served customers is maximized and the total assignment cost is minimized. They propose efficient algorithms for optimal assignment that employ novel edge-pruning strategies, based on the spatial properties of the problem. Additionally, they develop approximate CCA solutions that provide a tradeoff between result accuracy and computation cost. They also (Leong et al. 2010) propose the continuous optimal assignment problem, whose objective is to construct an optimal assignment between mobile users and a set of servers and then constantly maintain it. The optimal assignment has the minimum average distance between the users and their assigned servers under constraint that each user is assigned to exactly one server and the maximum possible numbers of users are served. To solve this problem, this work first accelerates the initial assignment computation by exploiting the geometric properties of the problem, subsequently splits the problem into smaller, independent ones, and then solves them using an off-the-shelf optimal assignment algorithm. Given a set of service sites, however, the spatial assignment problem aims at an optimal assignment of objects. Thus, it is not suitable for choosing the optimal coverage problem proposed by this paper. To the best of our knowledge, this is the first paper that is able to return the regions in which the minimum number of service sites are built to serve all the given objects.

### 3 Problem Definition

In this section, we first summarize some preliminary works. Then we formally define the OCC query processing. Table 1 summarizes the commonly-used symbols in this paper.

#### 3.1 Preliminaries

We use \( \mathcal{O} = \{o_1, \ldots, o_l, \ldots, o_n\} \) to denote the set of spatial objects. Suppose that the capacity constraints of all service sites are fixed at \( k \). We adopt Euclidean distance for realizing distance between objects’ locations and service sites’ locations in the two-dimensional Euclidean space \( \mathcal{U} \), even though our techniques can also apply to higher dimensions and other distance metrics.

To provide service for these objects, all the objects need to be located within coverage regions of service sites. Furthermore, the number of objects that are assigned to each service site is not more than \( k \). We assume that the coverage region of every service site is a circular disk with the center at \( p_j \) and the same radius \( r \). If \( \text{dist}(o_i, p_j) \leq r \), an object \( o_i \) is located within the coverage region of the service site \( p_j \). We summarize the notion of all objects that can be located within the same coverage region as follows.

**Definition 1.** Given a set of objects \( \mathcal{O} \), a coverage set \( \mathcal{C}_s \) is the set of objects, which satisfies (i) \( \mathcal{C}_s \subseteq \mathcal{O} \) if \( \text{dist}(o_i, p_j) \leq r \) \( \forall o_i \in \mathcal{C}_s \), where \( p_j \) denotes a service site; (ii) there is not any other set \( \mathcal{C}'_s \) that includes \( \mathcal{C}_s \).

As mentioned earlier, some objects located within the coverage region of a service site can be in a coverage set. Next, we discuss which objects are able to be covered by one service site as follows.

**Lemma 1.** For the set of objects \( \mathcal{C} \), if \( \text{dist}(o_i, o_c) \leq \sqrt{3}r \), \( \forall o_i, o_c \in \mathcal{C} \), all the objects in \( \mathcal{C} \) can be covered by one service site.

---

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathcal{O} )</td>
<td>a set of objects</td>
</tr>
<tr>
<td>( \mathcal{C}_s )</td>
<td>a coverage set</td>
</tr>
<tr>
<td>( n )</td>
<td>cardinality of ( \mathcal{O} )</td>
</tr>
<tr>
<td>( I )</td>
<td>number of k-constraint coverage sets</td>
</tr>
<tr>
<td>( S )</td>
<td>number of coverage sets</td>
</tr>
<tr>
<td>( o_i )</td>
<td>an object in ( \mathcal{O} )</td>
</tr>
<tr>
<td>( k )</td>
<td>capacity constraint</td>
</tr>
<tr>
<td>( k-C_s )</td>
<td>a k-constraint coverage set</td>
</tr>
<tr>
<td>( \mathcal{U} )</td>
<td>two-dimensional Euclidean space</td>
</tr>
<tr>
<td>( p_j )</td>
<td>a service site in ( \mathcal{P} )</td>
</tr>
<tr>
<td>( CR(p_j) )</td>
<td>coverage region of ( p_j )</td>
</tr>
<tr>
<td>( r )</td>
<td>radius of the coverage regions</td>
</tr>
<tr>
<td>( \text{dist}() )</td>
<td>distance function</td>
</tr>
</tbody>
</table>
3.2 Problem Formulation

To set up each service site for a $k$-constraint coverage set $k-C_s$, we need to find the region such that any point in this region is a possible location at which the service site can be built. We next describe a region such that setting up a service site within this region can serve any object in $k-C_s$, which is summarized as follows.

Definition 3. Given a $k$-constraint coverage set $k-C_s$, its feasible region, denoted by $FR_{k}$, is defined as a region which is composed of all points such that setting up a service site $p$ at any of these points ensures $dist(o, p) \leq r$ for $k-C_s$.

The feasible region of a $k$-constraint coverage set can be given by the intersection among the circular disks with centers at all objects’ locations. Figure 3 illustrates the feasible region denoted as shadow region for $\{o_1, o_2, o_3, o_4\}$. Since the distance between any point in the intersection region and the location of any object is not more than $r$, building a service site in this region can cover all the objects in a coverage set.

To minimize the cost for building server sites, all spatial objects in $O$ should be served by the minimum number of server sites. In fact, we need to choose the minimum number of $k$-constraint coverage sets which can include each object in $O$. We formally define $OCC$ queries that are able to return an optimal coverage scheme that includes the minimum number of $k$-constraint coverage sets and the relevant feasible regions for building the minimum number of service sites to serve all the objects.

Definition 4. Given a set of $O$, an $OCC$ query, $OCC(O, r, k)$, retrieves a series of feasible regions to build the minimum number of service sites for the $k$-constraint coverage sets which include all the objects in $O$. It is formally defined as

$$OCC(O, r, k) = \{<FR_{k}, k-C_s> | \forall p_j \text{ located in } FR_k \text{ and } o_j \in k-C_s, \text{ (i) } \bigcup_{s=1}^{Min} (k-C_s) = O, \text{ (ii) } -\exists T, T < Min \bigcup_{s=1}^{r} (k-C_s) = O\},$$

where $Min$ denotes the minimum number of server sites, and $r$ is the radius of the coverage region of $p_j$.

As mentioned in the above definition, some regions to set up the minimum number of service sites are retrieved by the $OCC$ query. Due to (ii), an optimal coverage is returned by Eq. (2). Then we will describe the framework for the $OCC$ query processing.
4 The Query Framework

As mentioned previously, we have presented OCC queries which can find some feasible regions such that setting up the minimum number of service sites within these regions would provide service for all the given objects. Each feasible region is searched by the objects in a k-constraint coverage set. According to the definition of k-constraint coverage sets, any k objects (or lower than k objects) in a coverage set can make up a k-constraint coverage set. Thereby, the key of answering OCC queries is finding the minimum number of k-constraint coverage sets that are able to include all the given objects. For a coverage set $C_k$, hence, the number of k-constraint coverage sets approximates to $C_k^{|C_k|}$, where $|C_k|$ denotes the cardinality of $C_k$. The cost of evaluating k-constraint coverage set for all coverage sets is time-consuming so that the cost of OCC queries raises serious practicality concerns in realistic settings. Aiming at this problem, we devise a general query framework for answering OCC queries efficiently and elegantly.

As illustrated in Figure 4, the query framework first finds all coverage sets from the given set of objects. According to Lemma 1, all the given objects are partitioned into coverage sets in which objects can be covered by one service site. Next, this framework proposes two approaches to return the coverage schemes and feasible regions for setting up service sites. For a coverage set, the naive solution wishes to insure the maximum number of service sites whose capacities are full. Therefore, this solution approach only retrieves the local optimum that insures the minimum number of service sites for a coverage set.

Aiming at this problem, the other approach is able to return a global optimum that all objects are served by the minimum number of service sites with a high-performance method. To improve the efficiency, this approach refines intermediate results for OCC queries.

5 Optimal k-Constrain Coverage Query Processing

In the previous sections, we have formalized OCC queries and devised the query framework for answering OCC queries. In this section, we describe OCC query processing in details.

5.1 Naive Solution

Based on Lemma 1, we can find which objects are able to be covered by one service site from the given set of spatial objects $O$. According to the definition of coverage set, all the coverage sets are returned easily. We use R-tree to index the set of objects $O$ in the two-dimensional space $U$. Due to the capacity constraint of $k$, any service site provides service for at most $k$ objects. If some objects are within a k-constraint coverage set, we say that these objects can be assigned to the same service site. Thereby, we find the minimum number of coverage sets which include all objects in $O$ to answer the OCC query. The query framework gives two approaches to retrieve some k-constraint coverage sets which can cover the set of objects $O$.

The main idea for the naive solution approach is insuring the maximum number of service sites whose capacities are full in a coverage set. For instance, given a set of objects $O = \{o_1, o_2, o_3, o_4, o_5, o_6\}$ and $O = \bigcup_{i=1}^3 C_i$ where $C_1 = \{o_1, o_2, o_3, o_4\}$, $C_2 = \{o_2, o_5\}$, and $C_3 = \{o_4, o_6\}$. Assume that the capacity constraint of each service site is fixed at 3. For ease of presentation, the naive solution assigns objects with IDs of objects ordering (i.e., the ID of an object $o_i$ is $i$). $\{o_1, o_2, o_3\}$ are assigned to a service site, which insures that the capacity of this service site is full in $C_1$. As shown in Figure 5(a), three other service sites are required to serve $o_4$ and $o_5$. Hence, the naive solution approach returns four service sites to serve $\{o_1, o_2, o_3, o_4\}$ in the example. However, Figure 5(b) illustrates a better coverage scheme which needs three service sites to serve these objects, and the three k-constraint coverage sets are $\{o_1, o_6\}$, $\{o_2, o_5\}$ and $\{o_3, o_4\}$ respectively.

As mentioned previously, the naive solution only returns a local optimum that is an optimal coverage scheme for each coverage set. Using this solution, the objects can be served by the lowest number of service sites in each coverage set (e.g., in $C_1$ $o_1, o_2, o_3$ and $o_4$...
are served by two service sites. However, all the given objects are served by service sites whose number may be not the lowest. The key steps of the naive solution algorithm is illustrated in Algorithm 1.

### Algorithm 1: Naive Solution Algorithm

**Input**: \( \mathcal{O} = \bigcup_{s=1}^{\mathcal{O}} \mathcal{C}_s \) and the capacity constraint \( k \)

**Output**: \( \{k\mathcal{C}_1, \ldots, k\mathcal{C}_m\} \) and \( \{\mathcal{FR}, \ldots, \mathcal{FR}_s\} \)

1. \( s \leftarrow 1 \)
2. \( i \leftarrow 1 \)
3. For each coverage set \( \mathcal{C}_s \) do
   4. While \( |\mathcal{C}_i| \geq k \) do
      5. \( \mathcal{C}_i \leftarrow \mathcal{C}_i/k\mathcal{C}_i \)
      6. \( i \leftarrow i + 1 \)
8. Take the rest objects of \( \mathcal{C}_s \) into \( k\mathcal{C}_i \)
9. \( \mathcal{O} \leftarrow \mathcal{O}/\mathcal{C}_s \)
10. For each \( k\mathcal{C}_i \) do
11. For each object \( o \) in \( k\mathcal{C}_i \) do
   12. Generate the circular disk with the center at \( o \)
13. Evaluate the intersection \( \mathcal{FR}_i \)

This algorithm first searches some \( k \)-constraint coverage sets whose capacities are full and a \( k \)-constraint coverage set whose capacity may not be full for a coverage set (lines 4-7). In order to ensure that objects are assigned to one service site, the algorithm deletes the objects assigned to \( k \)-constraint coverage sets (line 9). Lastly, Naive Solution Algorithm evaluates all feasible regions for \( k \)-constraint coverage sets. The naive solution can only retrieve a local optimum that is an optimal coverage scheme for each coverage set. To choose an optimal scheme for all coverage sets, we propose an efficient algorithm called Opt-C Algorithm in the next subsection.

### 5.2 Optimal Coverage Scheme

In this subsection, we present refinement methods for improving the efficiency of OCC queries. Then we discuss an optimal coverage scheme algorithm called Opt-C Algorithm.

Using the naive algorithm to answer OCC queries, the number of service sites that are required to be built for serving all the objects is more than that of the optimal coverage. If we choose an optimal coverage with the exhaustive method, the time cost would become prohibitive. Aiming at this problem, we first propose a method to refine coverage sets before choosing an optimal coverage, which is summarized as follows.

**Lemma 2.** For OCC queries, if \( \bigcup^m \mathcal{C}_{s1} \subseteq \bigcup^m (k\mathcal{C}_{s2}) \wedge m \geq m' \), \( m \) coverage sets \( \{\mathcal{C}_1, \ldots, \mathcal{C}_m\} \) can be refined safely.

**Proof.** For number \( m \) of coverage sets, \( \forall o_0, o_i \subseteq \bigcup^m \mathcal{C}_{s1} \). According to the definition of coverage sets, \( o_i \) can be covered by number \( m \) of service sites that are set up for these coverage sets. For \( k \)-constraint coverage sets of these coverage sets, at least \( m \) service sites are required to serve all the objects in \( \bigcup^m \mathcal{C}_{s1} \). Since \( \bigcup^m \mathcal{C}_{s1} \subseteq \bigcup^m (k\mathcal{C}_{s2}) \), \( o_i \subseteq \bigcup^m (k\mathcal{C}_{s2}) \). Note that, these objects can also be served by number \( m' \) of service sites. Due to \( m \geq m' \), these objects can be served by a lower number of service sites. Therefore, for our queries, \( \{\mathcal{C}_1, \ldots, \mathcal{C}_m\} \) cannot generate any optimal coverage and they should be refined safely.

From the above refinement method, we can refine coverage sets which are unable to generate any optimal coverage schemes. This method reduces coverage schemes that are certainly not the optimal coverage schemes to improve the query efficiency. As shown in Figure 6, for the set of objects \( \{o_1, o_2, o_3, o_4, o_5, o_6\} \), the coverage sets are \( \{o_1, o_2, o_5, o_6\} \), \( \{o_2, o_3, o_5, o_6\} \) and \( \{o_1, o_4, o_6\} \). This figure illustrates a coverage scheme that objects \( \{o_1, o_2, o_3\} \) are assigned to a service site and objects \( \{o_3, o_4, o_6\} \) are assigned to another service site, where the capacity constraint of each service site is equal to 3. Thus, the scheme requires two \( 3 \)-constraint coverage set. According to Lemma 2, \( \{o_1, o_2, o_5\} \), \( \{o_2, o_3, o_5, o_6\} \) and \( \{o_3, o_4, o_6\} \) can be refined since any coverage scheme requires at least two service sites over this set of objects.

To further improve the efficiency of OCC queries, another refinement method is proposed to reduce the number of objects.

**Lemma 3.** For an OCC query, if some objects are only within a coverage set \( \mathcal{C}_s \) and the number of these objects is not more than the capacity constraint \( k \), then the set of these objects is certain to be included by an optimal coverage.

**Proof.** Since some objects are only within a coverage set \( \mathcal{C}_k \), we suppose that a service site \( p_i \) is set up to serve these objects. Due to \( |\mathcal{C}_s| \leq k \), all objects covered by \( p_i \) are within a \( k \)-constraint coverage set, denoted by \( k\mathcal{C}_k \). It follows that these objects are certain to be served by one service site. Therefore, the set of these objects is included by an optimal coverage.

As mentioned in the above lemma, some objects satisfying this lemma can be taken into a \( k \)-constraint coverage set which are included by an optimal coverage. Thus, the number of objects that have not been assigned to any service sites is reduced, which results in the number of \( k \)-constraint coverage sets decreases.
cause signed a service and o

Figure 7 illustrates the case in details. Objects scheme
Figure 8: An example of choosing an optimal coverage

Thus, Opt-C\[\text{Min}\] insures that some objects are within the

Next, to answer OCC queries, we present an effective algorithm for retrieving an optimal coverage. To find the global optimum, all the possible coverage scheme is needed to be considered. We use Opt-C\([n, Min]\) to represent an optimal coverage scheme, which represents number \(Min\) of service sites to serve number \(n\) of objects. Note that, an optimal coverage scheme is defined as

\[
\text{Opt-C}[n, Min] = \text{Opt-C}[n-m, Min-s]\bigcup\bigcup_{i=1}^{s}(k-C_j),
\]

where \(\bigcup\bigcup_{i=1}^{s}(k-C_j)\) denotes \(k\)-constraint coverage sets including some objects which is not included by the union of a lower number of coverage sets. Thus, Opt-C\([n, Min]\) insures that number \(n\) of objects are served by the minimum number of service sites. From Eq.(3), an optimal coverage scheme Opt-C\([n, Min]\) can be evaluated with Dynamic Programming method. For any objects, this method is able to serve them with the minimum number of \(k\)-constraint coverage sets. We propose Opt-C Algorithm representing the above processing, which is illustrated in Algorithm 2.

**Algorithm 2: Opt-C Algorithm**

| Input | \(\mathcal{O} = \bigcup_{i=1}^{S} C_s\) and the capacity constraint \(k\) |
| Output | \(\{k-C_1, \ldots, k-C_{Min}\}\) and \(\mathcal{F}\mathcal{R}_1, \ldots, \mathcal{F}\mathcal{R}_{Min}\) |

1. \(i \leftarrow 1\)
2. for each object \(o_i\) in \(\mathcal{O}\) do
   3. take \(o_i\) into a set \(\mathcal{O}'\)
   4. if \(\mathcal{O}' \subseteq \bigcup_{i=1}^{s}(k-C_j)\) then
      5. \(r \leftarrow 1\)
      6. if \(r > s\) then
         7. \(r \leftarrow s\)
         8. \(i \leftarrow i + 1\)
      9. \(\text{Min} \leftarrow r\)
   10. while \(s \leq \text{Min}\) do
      11. take \(\mathcal{O}'\) into result set
      12. \(s \leftarrow s + 1\)
   13. for each \(k-C_j\) do
      14. for each object \(o\) in \(k-C_j\) do
         15. generate the circular disk with the center at \(o\)
         16. evaluate the intersection \(\mathcal{F}\mathcal{R}_j\)

Before Opt-C Algorithm, we can use refinement methods to reduce intermediate results for OCC queries. As mentioned in Algorithm 2, this algorithm returns an optimal coverage which is the global optimum instead of the local optimum. At first, it initializes the set of objects and coverage sets. Then this algorithm insures that some objects are within the minimum number of \(k\)-constraint coverage sets (lines 4-8). Finally, it chooses an optimal coverage scheme and returns the results for an OCC query (lines 13-16).

Compared to the naive solution, Opt-C Algorithm chooses the minimum number of \(k\)-constraint coverage sets. Figure 8 illustrates an example for an OCC query. As mentioned in Section 1, we consider a set of objects \(\{o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8\}\). According to Lemma 1, these objects can be transformed into some coverage sets. \(C_1 = \{o_2, o_8, o_9\}\), \(C_2 = \{o_1, o_3, o_9\}\), \(C_3 = \{o_2, o_3, o_9\}\), \(C_4 = \{o_2, o_7, o_8\}\), \(C_5 = \{o_1, o_5, o_7\}\), \(C_6 = \{o_6, o_5, o_7\}\), \(C_7 = \{o_1, o_5, o_6\}\). Assume that the capacity of each service site is set as \(3\). The sets \(\{o_4, o_5, o_6\}\) and \(\{o_2, o_7, o_8\}\) are both \(3\)-constraint coverage sets. Since \(C_5 \cup C_6 \cup C_7 \subset \{o_4, o_5, o_6\} \bigcup \{o_2, o_7, o_8\}\), \(C_5, C_6\) and \(C_7\) can be refined
by Lemma 2. Next, if some objects are only within a coverage set (e.g., \( o_1 \) in \( C_2 \)), the 3-constraint coverage set \( \{o_1, o_2, o_6\} \) is certain to be included by an optimal coverage due to Lemma 3. Because these coverage sets (or \( k \)-constraint coverage sets) are refined by the refinement methods, the efficiency of \( OCC \) queries would be improved considerably.

As shown in Figure 8, an optimal coverage scheme represents that all the objects can be within the minimum number of 3-constraint coverage sets (e.g., \( O = \{o_1, o_3, o_9\} \cup \{o_2, o_7, o_8\} \cup \{o_4, o_5, o_6\} \)), where each 3-constraint coverage set can be served by a service sites. The feasible regions are retrieved by the intersections among the circular disks with the centers at the locations of objects.

6 Experiments

In this section, we present results from an extensive empirical study over several datasets to illustrate the performance of our proposed methods.

6.1 Experimental settings

All the experiments were conducted on a PC with a 2.6GHz Processor and 2GB main memory. We use two real world datasets “Los Angeles (LA)” and “Greece (Gr)”, which were also available in Topologically integrated geographic encoding and referencing (tiger) system. Furthermore, one synthetic dataset (SYN) was generated by Uniform distribution in the two-dimensional Euclidean space. The range of the space is standardized in 1000 \( \times \) 1000. We used R-tree to index these datasets respectively. In order to evaluate methods exactly, we randomly generated 20 queries and computed the expectation for each query evaluation. We describe the three datasets in detail.

**LA Dataset:** LA dataset, a two-dimensional real dataset, is composed of 60K geographical objects described by ranges of longitudes and latitudes. We used the centers to represent the locations of these objects.

**Gr Dataset:** Gr dataset contains 5, 922 cities and villages in Greece. For the cities and villages, we also used the spatial objects to represent the locations of the cities and villages.

**SYN Dataset:** SYN dataset is composed of 10K spatial objects. Each object is generated by Uniform distribution in the whole space.

Alternative Techniques Considered. The aim of the experiments is to study the time cost and the number of required service sites for the algorithms to solve the optimal coverage problem under various settings. To the best of our knowledge, there is no reference method to solve this optimal problem. Therefore, we contrast Opt-C Algorithm returning the global optimum with the naive solution returning the local optimum.

6.2 Experimental Results

We first study the effect of data size on the performance of Opt-C Algorithm, Opt-C Algorithm with the refinement method (Opt-C Algorithm+Refinement) and the naive solution method under different datasets. We fix the radius \( r \) of each service site’s coverage region at 5, the capacity constraint \( k \) of each service site at 200, and vary the size of datasets. We observe the running time and the number of service sites that provide service for all the objects in the datasets.

Figure 9 shows the running time by the naive solution method, Opt-C algorithm and Opt-C Algorithm+Refinement algorithm in the three datasets. For any dataset, with the refinement method, Opt-C algorithm is more efficient than any other methods. The reason is that the refinement method improves the efficiency of answering \( OCC \) queries by reducing unnecessary \( k \)-constraint coverage sets.

We also observe that the effect of the capacity constraint \( k \) on the running time of these algorithms. We fix the radius \( r \) of each service site’s coverage re-
relation at 5. As shown in Figure 10, the running time of all the algorithms first increases and then reduces along with increasing $k$. This is because the number of $k$-constraint coverage sets for a coverage set $C$ approximates $C^2$ whose value first increases and then reduces along with increasing $k$. In addition, the running time of Opt-C Algorithm+Refinement is lower than the running time of the others.

Then we compare the number of service sites $|P|$ that provides service for all the objects in these datasets with different algorithms. We fix the radius $r$ of each service site’s coverage region at 5. As shown in Figure 11 obviously, more service sites are required with the size of datasets increasing. Furthermore, Opt-C Algorithm can cover these objects with less number of service sites than that of the naive solution for all datasets. It can be explained that the naive solution which chooses a local optimum that requires more number of service sites to serve these objects. Therefore, Opt-C Algorithm is more effective than the naive solution.

Next, we examine the effect of the radius $r$ of each service site’s coverage region on the running time of different algorithms under the three datasets. We fix the capacity of each service site at 200. As described in Figure 12, for all the datasets, Opt-C Algorithm+Refinement is more efficient than the other two algorithms. When $r$ raises from 1 to 20, the running time first decreases and then increases. The efficiency of these algorithms is lower with larger $r$ or smaller $r$, since larger $r$ results in more objects for each coverage set and smaller $r$ generates more coverage sets, which would reduce the efficiency.

We study the utilization rate of service sites under different capacity constraints with Opt-C Algorithm and the naive solution. The rate is equal to dividing $k$ by the average number of objects served by a service site. For any dataset, we vary the radius $r$ from 1 to 20. Figure 13 illustrates that the utilization rate of Opt-C Algorithm is much higher than that of the naive solution. The reason is that Opt-C Algorithm returns an optimal coverage scheme that has the minimum number of service sites. Therefore, the average number of objects served by a service site is higher.

Finally, we test the performance of our refinement methods under the three datasets. The refinement methods can refine massive $k$-constraint coverage sets before implementing Opt-C Algorithm. We fix the radius $r$ of each service site’s coverage region at 5 and 10 respectively, and vary $k$ from 100 to 500. As shown in Figure 14, our refinement methods reduce the number of $k$-constraint coverage sets for all the datasets effectively. Furthermore, the reducing rate is not lower than 30%.

7 Conclusions

In this paper, we have proposed a new type of spatial queries called Optimal $k$-Constraint Coverage (OCC) queries. Given a set of objects, these queries are able to return an optimal coverage scheme and a series of regions such that setting up the minimum number of service sites within these regions would guarantee all the objects to be served, where each service site has the capacity constraint and the coverage region in which the service site only can provide service for at most $k$ objects. To answer OCC queries efficiently, we have presented a general query framework. In the framework, we first have proposed the naive solution algorithm for choosing an optimal coverage in each coverage set, which retrieves the local optimum. To improve it, we have addressed another algorithm called Opt-C Algorithm that can choose an optimal coverage scheme. Using this algorithm, all the objects can be included by the lowest number of $k$-constraint coverage sets. Furthermore, two refinement methods have been developed to reduce massive $k$-constraint coverage sets before answering OCC queries. Finally, we have studied the performance of the proposed methods through the theoretical analy-
sis and extensive experiments with synthetic and real datasets.

8 Acknowledgments

The research is supported by the National Natural Science Foundation of China under Grant Nos. 60873009, 61003058 and the Fundamental Research Funds for the Central Universities No.N100604014.

References


Chen, L., Özsu, M. & Oria, V. (2005), Robust and fast similarity search for moving object trajectories, in ‘SIGMOD’.


A Branch and Bound Method for Min-dist Location Selection Queries

Jianzhong Qi\(^1\) Zhenghua Xu\(^2\) Yuan Xue\(^3\) Zeyi Wen\(^4\)

Department of Computing and Information Systems
University of Melbourne,
111 Barry St, Carlton, Victoria 3053, Australia

Email: \(^1\)jzqi, \(^2\)zhxu, \(^4\)zeylw}@csse.unimelb.edu.au
\(^3\)yuaxue@unimelb.edu.au

Abstract

Given a set of clients and a set of existing facilities, the min-dist location selection query returns a location from a given set of potential locations for establishing a new facility so that the average distance between a client and her nearest facility is minimized. This type of queries has a wide range of applications in marketing, decision support systems, urban development simulations and massively multiplayer online games. The computational cost of a naive algorithm, which sequentially scans all the potential locations, is too high to process this type of queries in real time. Motivated by this, we propose a branch and bound algorithm that exploits geometric properties of the data objects and early prunes unpromising potential locations from consideration to achieve a higher query processing efficiency. We conduct a detailed cost analysis and extensive experiments to validate the efficiency of the branch and bound algorithm. The results show that the proposed algorithm outperforms the naive algorithm constantly.

Keywords: Spatial databases, optimal location selection, min-dist metric

1 Introduction

Many businesses or organizations manage large numbers of facilities. For example, Walmart has warehouses, and Australia Post has branch offices. It is a common need for a business to add facilities. For example, Walmart might want to add a warehouse to reduce the distances between its stores and warehouses; Australia Post might want to add a new post office in an emerging suburb to reduce the traveling distances for their postmen. Usually, a set of potential locations is available, typically from a real estate agent (e.g., real estate websites provide hundreds of thousands of places for renting or buying).

This paper investigates the min-dist location selection query, which selects a location (for Walmart or Australia Post) that minimizes the average distance between a client (a chain store or an addressee) and her nearest facility (warehouse or post office) to reduce logistics cost or to improve the quality of service. We assume that the business has knowledge about its client distribution from surveys or past sales records.

The min-dist location selection query also has other applications as follows. In urban development simulation, very often urban planners need to simulate the above location selection procedure, so that the influence of establishing a new public facility or business center on the residents can be observed. In the multi-billion dollar computer game industry, massively multiplayer online games (MMOGs) like World of Warcraft have group quests for players to complete in teams, which mostly involve killing mobs (monsters or other non-player characters). As the quests often take players days or even weeks to complete, it is common for players to leave and rejoin the game during a quest. When a player rejoins the game, the subquest she was on may have been completed by her teammates and the team has moved on to another region to complete other subquests. It would be a waste of time for this player to rejoin the game from where she left. A very helpful utility for the game is selecting a starting point from a set of preset rejoin locations to minimize the average distance between a mob and its nearest player, so that players can focus on fighting mobs rather than walking.

The example in Figure 1 illustrates the query: \(\{c_1, c_2, \ldots, c_8\}\) is a set of clients (residents or mobs), \(\{f_1, f_2\}\) is a set of existing facilities (public facilities or teammates) and \(\{p_1, p_2\}\) is a set of potential locations (candidate locations for new facility establishment or rejoin). Now we need to select one from the potential locations to establish a new facility. Before adding a new facility, \(f_1\) is the nearest facility of \(c_1, c_2, c_3\) and \(c_6, f_2\) is the nearest facility of \(c_4, c_5, c_7\) and \(c_8\). If a new facility is established at \(p_1\), it will become the nearest facility for \(c_1, c_2\) and \(c_3\). If a new facility is established at \(p_2\), it will become the nearest facility of \(c_4\) and \(c_5\). As we can observe, \(p_2\) results in a smaller average distance between a client and the nearest facility, so it is selected as the answer.

Although an existing commercial software (ArcGIS 2011) can solve several kinds of simpler location optimization problems, none can solve the min-dist location selection problem, as we will detail in Section 2.

1.1 Contributions and Organization of the Paper

In this paper, we examine solutions to the min-dist location selection query and make the following contributions.

- We analyze the properties of the min-dist location selection query and propose pruning techniques to reduce the search space for processing the query.

Figure 1: An example for the query

\[c_1, c_2, \ldots, c_8\] is a set of clients (residents or mobs), \(\{f_1, f_2\}\) is a set of existing facilities (public facilities or teammates) and \(\{p_1, p_2\}\) is a set of potential locations (candidate locations for new facility establishment or rejoin). Now we need to select one from the potential locations to establish a new facility. Before adding a new facility, \(f_1\) is the nearest facility of \(c_1, c_2, c_3\) and \(c_6, f_2\) is the nearest facility of \(c_4, c_5, c_7\) and \(c_8\). If a new facility is established at \(p_1\), it will become the nearest facility for \(c_1, c_2\) and \(c_3\). If a new facility is established at \(p_2\), it will become the nearest facility of \(c_4\) and \(c_5\). As we can observe, \(p_2\) results in a smaller average distance between a client and the nearest facility, so it is selected as the answer.
- Based on the proposed pruning techniques, we propose a branch and bound method to efficiently process the query.
- We perform an analytical cost study and an extensive experimental study. The results confirm the effectiveness of the proposed pruning techniques and the efficiency of the proposed algorithm.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 formulates the query and presents a naive algorithm. Section 4 presents the branch and bound method. Section 5 analyzes the cost of the proposed method and Section 6 presents the experimental results. Finally, Section 7 concludes the paper.

2 Related Work

We review three categories of work below: work on the reverse nearest neighbor (RNN) query, work on min-dist location optimization problems, and work on min-inf location optimization problems.

RNN query: Korn & Muthukrishnan (2000) define the RNNs of an object \( o \) to be the objects whose respective nearest neighbor (NN) is \( o \). They propose an RNN-tree based solution to the query, where the RNN-tree is an R-tree (Guttman 1984) variant that indexes NN circles of the data objects rather than the data objects themselves. Here, the NN circle of an object \( o \) is a circle that centers at \( o \) with its radius being the distance between \( o \) and \( o \)'s nearest neighbor. Based on the NN circles, to find the RNN of an object \( o \) only requires checking which objects’ NN circles enclose \( o \). However, the RNN-tree based solution has two major drawbacks. One is that it requires the extra maintenance of an RNN-tree. The other is that it requires precomputing the NN circles. Therefore, this solution cannot handle objects with frequent updates. To solve the first problem, Yang & Lin (2001) propose to integrate the NN circle information into an R-tree that indexes the clients themselves, so that the resultant R-tree variant, the RdNN-tree, can be used to process RNN queries as well as other common types of queries on the clients, and thus avoid the maintenance of an extra RNN-tree. To solve the second problem, Stanoi & al. (2001) propose an approximation-refinement framework to compute the RNNs on the fly, so that no precomputation is needed.

There are studies on RNN query variants under different settings. For example, the reverse \( k \) nearest neighbor (RkNN) query finds objects whose \( k \) nearest neighbors include the query object. Wu et al. (2008) study the RkNN query on continuously moving objects, which correlates two sets of moving objects according to their closeness. The continuous join query on extended moving objects (Zhang et al. 2008, n.d.) also correlates multiple sets, but it focuses on intersecting objects with a time-constraint technique rather than closeness. While these approached work well for a single R(k)NN query, they are not tailored for computing RNNs for large amount of objects at the same time, which is one of the key difficulties in our study.

Max-inf problems: Max-inf location optimization problems aim at maximizing the influence of the locations, where the influence of a location is defined by the number of clients it attracts. Here, the concept of “attract” can have different meanings in different max-inf problems. Cabello et al. (2005) propose the MAXCOV facility location problem, which finds regions in the data space to maximize the numbers of RNNs for the points in these regions. Figure 2 gives an example, where the gray regions are the optimal regions. Points in these regions have 4 RNNs, while any point outside of these regions has at most 3 RNNs. Cabello et al. (2005) introduce the concept of nearest location circle (NLC) to solve the problem, where the NLC of a client \( c \) is a circle centered at \( c \) with its radius being the distance between \( c \) and \( c \)'s nearest existing facility. To find the solution for the MAXCOV criterion based problem is to find the regions that are enclosed by the largest number of NLCs, which requires complex computations. Wong et al. (2009) study this problem further and propose a more efficient method to compute the regions overlapped by the largest number of NLCs. Xia et al. (2005) propose the top-\( t \) most influential sites problem and a branch and bound algorithm to the problem. This problem finds the top-\( t \) most influential existing sites within a given region \( Q \). Du et al. (2005) find a point from a continuous candidate region that can maximize the influence value. They use \( L_i \) as the distance metric and have a strong assumption that all the roads are either horizontal or vertical. Cheema et al. (2011) find an influence zone for a query location \( p \), where the clients inside this zone form exactly the RANN query result of \( p \). A more recent study (Huang, Wen, Pathan, Taylor & Zhang 2011) ranks the candidate locations according to their influence values and another study (Huang, Wen, Qi, Zhang, Chen & He 2011) contributes in efficient algorithms to compute the top-\( k \) most influential candidate locations. Unlike the above problems, which relate the influence values to the cardinalities of RNN sets, Gao et al. (2009) propose to find the optimal location \( p \) outside a given region \( Q \) based on locations’ optimality, where the optimality of a location \( p \) is modeled by the amount of clients in \( Q \) whose distances to \( p \) is within a given threshold \( t \). Intuitively, the more clients \( p \) attracts, the greater its optimality. These studies differ from ours in optimization functions and other settings. Thus, their solutions do not apply.

Min-dist problems: Qi et al. (2012) first propose the min-dist location selection problem. Their study (Qi et al. 2012) is a concurrent study of ours. In that paper, solutions focus on how to efficiently identify the influenced clients for the potential locations, while this paper gives a solution on how to early prune unpromising potential locations from consideration.

Papadias et al. (2005) find the existing facility that has the smallest average distance to all the clients. They do not consider establishing a new facility. Zhang et al. (2006) propose the min-dist optimal-location problem. Given a client set \( C \), an existing facility set \( F \) and a region \( Q \), it finds points within \( Q \) so that if a new facility is established at any one of these points, the average distance of the clients to their respective nearest facilities is minimized. Figure 3 gives an example, where \( p_1 \) may be one of the points in the answer set and it is not the solution \( p_2 \) to our problem. Zhang et al. (2006) propose a method that first identifies a set \( L \) of candidate locations from \( Q \) and then divides \( L \) progressively until the answer set is found. Xiao et al. (2011) study the min-dist problem in road networks. They have a candidate edge set \( E \) for the new facility to be
established on. Their key insight is that the optimal location on a candidate edge must be one of the endpoints of the edge. Thus, only the endpoints of the edges in E need to be checked for the problem solution.

Zhang et al. (2006) and Xiao et al. (2011)’s studies have the same min-dist optimization function as ours, but our study has a set P, the potential locations given as candidates for selection. In many real applications, we can only choose from some candidate locations, e.g., McDonald’s may only open a new restaurant at a place for rent or sale rather than anywhere in a region or on a road. The main idea of Zhang et al.’s solution is the fast identification of a small set L of candidate locations from Q. However, the candidate locations in L could be any point from Q, which may not even contain a potential location from our potential location set P. Similarly, the endpoints of the edges in E (Xiao et al. 2011) are different from the points in P. This means that in the general case their approaches cannot provide a correct answer to our problem, and thus are not applicable.

Note that in computational geometry, given a set C of object locations (e.g., clients), the k-medoid query (Mouratidis et al. 2005) finds a set of medoids C’ ⊆ C with cardinality k that minimizes the average distance from each object c ∈ C to its closest medoid in C’. The k-median query is a variation, where we find k locations called medians, not necessarily in C, which minimize the average distance (from each object c ∈ C to its closest median). These two types of queries are actually using the min-dist metric. However, our problem is different from both of them. A fundamental difference is that these problems do not assume a set of existing facilities or a set of potential locations, but we do. If there is at least one existing facility or some potential locations to be chosen from in a specific location selection problem, k-medoid queries or k-median queries do not apply.

Related commercial software: As mentioned in the Introduction, an existing commercial software (ArcGIS 2011) can solve several kinds of simpler location optimization problems. The most related problem this software can solve is called the minimize impedance query, which finds locations for a set of new facilities to minimize the sum of distances between clients and their respective nearest facilities. However, this problem does not consider existing facilities. If we use this software to find a set of locations S_l for new facilities, there is no guarantee that S_l will contain all the points in the set of existing facilities F_l. Therefore, this software does not solve our problem.

3 Preliminaries

This section presents the definition of the min-dist location selection query and a naive algorithm to process the query. Frequently used symbols are summarized in Table 1.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>An entry in a R-tree node</td>
</tr>
<tr>
<td>o</td>
<td>Any point in the data space</td>
</tr>
<tr>
<td>dist(o1, o2)</td>
<td>The distance between two points o1 and o2</td>
</tr>
<tr>
<td>C, F, P</td>
<td>The set of clients, existing facilities and potential locations, respectively</td>
</tr>
<tr>
<td>n_c, n_f, n_p</td>
<td>Cardinality of C, F, and P, respectively</td>
</tr>
<tr>
<td>c, f, p</td>
<td>A client in C, an existing facility in F, and a potential location in P, respectively</td>
</tr>
</tbody>
</table>

Table 1: Frequently Used Symbols

3.1 Problem Definition

All data objects (clients, facilities and potential locations) are represented by points in an Euclidean space. We may refer to the data objects as data points or simply as points. Let dist(o1, o2) denote the distance between two points o1 and o2, and n_c be the number of clients. The min-dist location selection query is defined (Qi et al. 2012) as follows.

**Definition 1. Min-dist location selection query.**
Given a set of points C as clients, a set of points F as existing facilities and a set of points P as potential locations, the min-dist location selection query finds a potential location p_m ∈ P for a new facility to be established at, so that ∀p ∈ P:

\[
\sum_{c \in C} \{\min \{dist(c, o)|o \in F \cup p_m\}\} / n_c 
\leq \sum_{c \in C} \{\min \{dist(c, o)|o \in F \cup p\}\} / n_c.
\]

Since the denominator is the same on both sides of the inequality, the problem is equivalent to minimizing the sum (instead of the average) of the distances between the clients and their respective nearest facilities.

3.2 A Naive Algorithm

A straightforward algorithm to the min-dist location selection query is to sequentially check all potential locations. For every new potential location p, we compute the sum of the distances of all clients to their respective nearest facilities. The potential location with the smallest sum is the answer. We call this algorithm the sequential scan (SS) algorithm.

**Algorithm 1: SS(C, P)**

1. optLoc ← NULL; // optLoc is the optimal location;
2. for p ∈ P do
3. p.distSum ← 0;
4. for c ∈ C do
5. if dist(p, c) < c.dmn(c, F) then
6. p.distSum ← p.distSum + dist(p, c);
7. else
8. p.distSum ← p.distSum + c.dmn(c, F);
9. if optLoc = NULL or p.distSum < optLoc.distSum then
10. optLoc ← p;
11. return optLoc;

In SS, repeatedly finding the nearest facility to each client for every potential location is too expensive. Therefore, we precompute the distances of all the clients to their respective nearest facilities and store the distances. This precomputation involves a nested loop iterating through

\[1\text{Note that there may be ties in the average distances. To simplify our discussion, we always return the first potential location found that have the smallest average distance.}\]
every client and for every client iterating through every facility. KNN-join algorithms (e.g., Yu et al. (2010)) can do this more efficiently and maintain the results dynamically when clients and facilities are updated. The SS algorithm with precomputation is shown in Algorithm 1, where $c.dmn(c,F)$ denotes $c$’s precomputed distance to her closest existing facility and is stored with $c$’s record.

We see that even with precomputation SS is still very costly as it has to access the whole client dataset $\Omega^C$ times, where $n_p$ is the cardinality of $P$ and $C_p$ is the capacity of a block for $P$ (assuming we read $P$ in disk blocks). Therefore, the need for an efficient algorithm is obvious.

4 A Branch and Bound Method

In this section, we propose a brand and bound method that exploits data objects’ geometric properties to prune unpromising potential locations from consideration, so that the min-dist location selection query can be processed more efficiently. This method requires the query to be redefined in a form that enables the computation for the bounds. Next, we start with redefining the query.

4.1 Query Redefinition

We call the distance between a client $c$ and her nearest facility the nearest facility distance (NFD) of $c$. Let $dmn(o,S)$ denote the distance between a point $o$ and its nearest point in a set $S$. Then $dmn(c,F)$ and $dmn(c,F \cup p)$ denote the NFD of $c$ before and after a new facility is established on a potential location $p$, respectively. The min-dist location selection query is actually minimizing the sum of all the clients’ NFD.

If $o$ is a point not in the set $F$ and $dist(c,o) < dnn(c,F)$, then establishing a new facility at $o$ will reduce the NFD of $c$. In this case, we say that $c$ can get an NFD reduction from $o$. We define the influence set of $o$, denoted by $IS(o)$, as the set of clients that can get NFD reduction from $o$. Formally, $IS(o) = \{c| c \in C, \text{dist}(c,o) < dnn(c,F)\}$. The influence set of a potential location $p$ includes all the clients that will reduce their NFD if a new facility is established at $p$. For example, in Figure 1, $IS(p_1) = \{c_1,c_2,c_3\}$, and $IS(p_2) = \{c_4,c_5\}$.

If $IS(p) \neq \emptyset$ for a potential location $p$, then establishing a new facility at $p$ will reduce the sum of the clients’ NFD. We call the sum of the clients’ NFD reduced by $p$ the distance reduction of $p$, denoted by $dr(p)$. Formally, $dr(p) = \sum_{c \in IS(p)}(dmn(c,F) - dnn(c,F \cup p))$. Minimizing the sum of the clients’ NFD when adding a facility on $p$ is equivalent to maximizing $dr(p)$. Therefore, the min-dist location selection query can be redefined as follows.

Definition 2. Given a set of points $C$ as clients, a set of points $F$ as existing facilities and a set of points $P$ as potential locations, the min-dist location selection query finds a potential location $p_m \in P$, so that $\forall p \in P: dr(p) \leq dr(p_m)$.

4.2 The Branch and Bound Algorithm

The Branch and Bound (BB) method estimates the potential locations’ $dr$ values to achieve early pruning. It assumes that the datasets are indexed in spatial indexes. Specifically, it assumes an R-tree $R_P$ to index the potential location set $P$, and an R-tree variant $R_F^C$ to index the client set $C$ and store some other information for the computing the bounds (details are in Section 4.3), although the method can be easily adapted to any hierarchical spatial index. The branch and bound scheme is performed during a traversal on both trees.

Before explaining the BB method, we need to extend the concept of the influence set of a point to the influence set of a node in $R_P$. Let $N_P$ be a node in $R_P$. The influence set of the node $N_P$ is defined as $IS(N_P) = \{c| c \in C \text{ and } \exists o \in N_P.mbr: \text{dist}(c,o) < dnn(c,F)\}$. A client is in $IS(N_P)$ if there is a point (not necessarily a potential location) in the minimum bounding rectangle (MBR) of $N_P$ that can reduce the client’s NFD. Intuitively, $IS(N_P)$ defines the set of clients which might achieve distance reduction without knowing which potential locations are actually in $NP$; it is the union of $IS(o)$ for any possible point $o$ in the MBR of $N_P$. Figure 4 gives an example. Node $N_P$ indexes two potential locations $p_1$ and $p_2$. We can observe that $IS(p_1) = \{c_1,c_2,c_3\}$ and $IS(p_2) = \{c_4,c_5\}$. Also, there are three points $o_1,o_2$ and $o_3$ in the MBR of $N_P$. We have $dist(o_1,c_6) < dnn(c_6,F)$, $dist(o_2,c_7) < dnn(c_7,F)$ and $dist(o_3,c_8) < dnn(c_8,F)$. Therefore, $IS(N_P) = \{c_1,c_2,...,c_8\}$.

The idea of the BB method is as follows. We traverse $R_P$ in a depth-first order and simultaneously traverse the R-tree variant on $C$, $R_F^C$ (recall that this R-tree maintains some additional information for computing bounds). We use the tree structure to narrow down the clients we have to examine for identifying the influence sets of a potential location. As we visit a node $N_P$ of $R_P$ (suppose $N_P$ is in level $l$ of $R_P$), we identify a set of nodes from level $l$ of $R_F^C$ whose subtrees must cover all the clients in $IS(N_P)$; we call this set of nodes from $R_F^C$ the influence nodes (IN) of $N_P$ and denote it by $IN(N_P)$. Based on the MBR of $N_P$ and the aggregate information stored in the nodes of $IN(N_P)$, we can compute a lower bound and an upper bound for the distance reduction of all the potential locations contained in the subtree rooted at $N_P$. As we traverse down $R_P$, the bounds will become tighter. We record the largest lower bound so far during traversal, which serves as a pruning distance, denoted as $pd$. If at any time we encounter a node in $R_P$ with an upper bound of distance reduction smaller than $pd$, then that node can be discarded from the search, since our goal is to find the potential location with the largest distance reduction. When we reach the leaf level of $R_P$, we get the exact information of the potential locations and can compute their exact distance reductions.

If $pd$ is smaller than an exact distance reduction, then $pd$ gets updated to it. The search stops as we finish traversing $R_P$ and the potential location with the largest $dr$ value is the answer.

The derivation of the upper bound $maxdr$ and the lower bound $mindr$ are presented in Sections 4.3 and 4.4, respectively. The condition to identify $IS(N_P)$ of $N_P$, and the structure of $R_F^C$ are related to the upper bound, so we present them in Section 4.3.

The recursive part of the BB algorithm is summarized in Algorithm 2. We explain the algorithm together with the example in Figure 5, where the nodes within a dotted rectangle represent the IN of a node in $R_P$. Initially, $N_P$ is set to the root node of $R_P$, whose MBR is set to $\text{NULL}$. For each node $N_P$ being accessed, we con-
struct IN($e_p$) for each of its entry $e_p$ using the child nodes of the nodes in IN($N_p$) (lines 1 to 3). In Figure 5, we have IN($e_1$) = $\{N^b_1\}$, IN($e_2$) = $\{N^b_2, N^b_3\}$ and IN($e_3$) = $\{\}$.
These actually means IN($N_1$) = $\{N^b_1\}$, IN($N_2$) = $\{N^b_2, N^b_3\}$ and IN($N_3$) = $\{\}$. Then the child nodes of $N_1$, $N_2$ and $N_3$ will use the child nodes of these INs to construct their own INs. For example, IN($N_{11}$) is constructed with the child nodes of $N^b_1$, and the resultant IN($N_{11}$) is $\{N^b_{11}, N^b_{12}\}$. Similarly, IN($N_{12}$) and IN($N_{13}$) are constructed with the child nodes of $N^b_2$ and $N^b_3$, which results in IN($N_{12}$) = $\{N^b_{12}, N^b_{13}\}$ and IN($N_{13}$) = $\{N^b_{12}, N^b_{13}, N^b_{14}\}$. This ensures that the level of

\[ \text{ALGORITHM 2: BB($N_p$, IN, pd, optLoc)} \]

1. if $N_p$ is a non-leaf node then
   2. for $e_p \in N_p$ do
      3. Construct IN($e_p$), compute $\text{maxdr}$ and $\text{mindr}$;
      4. if $\text{maxdr} > pd$ then
         5. if $\text{mindr} > pd$ then
            6. $pd \leftarrow \text{mindr}$;
            7. BB($e_p$.childnode, IN($e_p$), pd, optLoc);
   8. else if $N_p$ is a leaf node but nodes in IN are non-leaf nodes then
      9. for $N^b_c \in \text{IN}$ do
         10. $\text{IN}' \leftarrow \{\}$;
         11. for $e^b_c \in N^b_c, e^b_c$ satisfies IN conditions of $N_p$ do
            12. $\text{IN}' \leftarrow \text{IN}' \cup e^b_c$.childnode;
            13. BB($N_p$, $\text{IN}'$, pd, optLoc);
      14. else // $N_p$ and nodes in IN are all leaf nodes
         15. for $N^b_c \in \text{IN}$ do
            16. for $e_p \in N^b_c$ do
               17. for $e^b_c \in N^b_c, \text{dist}(e_p, e^b_c) < e^b_c$.dnn($c, F$) do
                  18. $e_p$.dr \leftarrow $e^b_c$.dnn($c, F$) - $\text{dist}(e_p, e^b_c)$;
               19. if optLoc = NULL or $e_p$.dr > optLoc.dr then
                  20. optLoc \leftarrow $e_p$;
            21. if optLoc.dr > pd then
               22. pd \leftarrow optLoc.dr;

\[ \text{Figure 5: Example of algorithm BB} \]

IN($N_p$) and that of $N_p$ are the same. The bounds $\text{maxdr}$ and $\text{mindr}$ of an entry $e_p$ are computed using the aggregated attributes stored in the parent entries of IN($e_p$). The child node of $e_p$ will be pruned if $\text{maxdr} \leq pd$ (line 4). If $\text{mindr} > pd$, pd will be updated to $\text{mindr}$ (lines 3 to 6). Then the child nodes of the unpruned entries of $N_p$ are traversed (line 7). Note that the heights of $R_P$ and $R^b_P$ may be different. Thus, there are different procedures for the condition where the traversal reaches the leaf level of only one tree. (i) If the traversal reaches a non-leaf node $N_p$ of $R_P$ and IN($N_p$) are leaf nodes, we construct the INs of the child nodes of $N_p$ using nodes of IN($N_p$) since there is no child node for the nodes in IN($N_p$) (lines 1 to 7). (ii) If the traversal reaches a leaf node $N_p$ of $R_P$, and IN($N_p$) are non-leaf nodes, we construct a subset of IN($N_p$) with the entries of each node $N^b_c \in \text{IN}($$N_p$), denote it as IN($N^b_c$), and perform algorithm BB on $N_P$ and IN($N^b_c$) (lines 8 to 13). Doing this recursively guarantees that all clients contained in the subtrees of the nodes in IN($N_p$) will be accessed, which means IS($N_p$) is fully accessed. The advantage of this method compared to the construction of INs for the data entries directly is its reduction of node accesses in $R^b_P$. Since entries of a same node tend to have similar INs, if we access the nodes in IN($N_p$) directly for each data entry of $N_p$, many nodes (and their descendant nodes) will be accessed repeatedly and the number of node accesses will increase. When the traversal reaches the leaf nodes of both trees, for each data entry $e_p$ of a node $N_p$ of $R_P$, all entries of each node $N^b_c \in \text{IN}($$N_p$) are accessed to update $e_p$.dr (lines 14 to 18); optLoc and pd are updated accordingly (lines 19 to 22). When the traversal ends, IN has been accessed for all potential locations and optLoc is found.

### 4.3 An Upper Bound

We derive an upper bound $\text{maxdr}$ for the distance reduction of all potential locations contained in the subtree rooted at a node $N_p$. To simplify our discussion, we use sub($N$) to denote the set of data entries contained in the subtree rooted at the node $N$. Then $\text{sub}(N)$ denotes the cardinality of sub($N$). For example, sub($N_p$) denotes the set $\{p|p$ is a potential location indexed in the subtree root at node $N_p}\$.

The following discussion holds for any R-tree based index on the set of clients, we use $\mathcal{N}_C$ (instead of $\mathcal{N}_C^b$ used in the above subsection) to denote a node in such an index. Recall the definition of $\text{dr}$ of a potential location $p$, $\text{dr}(p) = \sum_{c \in \text{IS}(p)} \text{dnn}(c, F) - \text{dist}(c, F \cup p)$. We derive $\text{maxdr}$ in a similar way. Since we are using a set of nodes to compute $\text{maxdr}$ for a node $N_p$ of $R_P$, $\text{maxdr}(N_p) = \sum_{c \in \text{IS}(\mathcal{N}_C)} (\text{sub}(\mathcal{N}_C) \cdot (g_1(\mathcal{N}_C) - g_2(\mathcal{N}_C, N_p)))$, where $S$ is a set of client R-tree nodes, $g_1$ is a metric related to a node $\mathcal{N}_C$, and $g_2$ is a metric related to $\mathcal{N}_C$ and $N_p$. We use $\text{sub}(\mathcal{N}_C)$ because, in an ideal condition, every client $c \in \text{sub}($$\mathcal{N}_C$) is in IS($p$) of some potential location $p \in N_p$. To derive a reasonable upper bound, we try to find small $S$ and $g_1(\mathcal{N}_C)$ and a large $g_2(\mathcal{N}_C, N_p)$. We use the metrics, maxDist and minDist, as $g_1$ and $g_2$, respectively. IN($N_p$) is used as $S$, which will be established based on maxDist and minDist. The definitions of maxDist and minDist are as follows.

**Definition 3. The largest NFD value (maxDist) for all clients that are in the subtree rooted at a node $\mathcal{N}_C$**

This metric is proposed by Yang & Lin (2001) as $\text{maxdnn}$ and it is defined in a bottom-up fashion. For the leaf nodes, maxDist($\mathcal{N}_C$) is defined as the largest NFD value for all the clients indexed in $\mathcal{N}_C$, i.e., maxDist($\mathcal{N}_C$) = $\max(\text{dnn}(c, F)|c \in \mathcal{N}_C} \}$, while for the non-leaf nodes, maxDist($\mathcal{N}_C$) is defined as the largest maxDist value for all the child nodes of $\mathcal{N}_C$, i.e,
\( \text{maxDist}(N_c) = \max\{\text{maxDist}(c_{i}.\text{childnode}) | c_i \in N_c \} \).

Figure 6 gives an example, where \( dnn(c_3, F) \), \( dnn(c_4, F) \) \text{and} \( dnn(c_4, F) \) are the largest NFD values of the clients indexed in the leaf nodes \( N_1 \), \( N_2 \) and \( N_3 \), respectively. The three NFD values are picked as the respective \( \text{maxDist} \) values of the three nodes. Meanwhile, the largest value among \( \text{maxDist}(N_1) \), \( \text{maxDist}(N_2) \) \text{and} \( \text{maxDist}(N_3) \) is picked as the \( \text{maxDist} \) of the parent node \( N_4 \) of these three nodes. Therefore, we get \( \text{maxDist}(N_4) = \text{maxDist}(N_3) = dnn(c_4, F) \), which is effectively the largest NFD value for all the clients in the subtree rooted at \( N_4 \).

\( \text{maxDist}(N_i), i = 1, 2, 3 \)

Figure 6: Example of \( \text{maxDist} \)

**Definition 4.** The smallest distance (minDist) between two objects.

This metric is proposed by Roussopoulos et al. (1995). We use \( \text{minDist}(c, N_p) \) to denote the smallest distance between a point \( c \) and the MBR of a node \( N_p \). If \( c \) is within the MBR of \( N_p \), then \( \text{minDist}(c, N_p) = 0 \). Otherwise, \( \text{minDist}(c, N_p) \) is the distance between \( c \) and its nearest point on the MBR of \( N_p \). Similarly, we use \( \text{minDist}(N_c, N_p) \) to denote the smallest distance between the MBR of \( N_c \) and the MBR of \( N_p \). If these two MBRs overlap, then \( \text{minDist}(N_c, N_p) = 0 \). Otherwise, \( \text{minDist}(N_c, N_p) = \min\{\text{dist}(o_1, o_2) | o_1, o_2 \text{ are points on the MBRs of } N_c \text{ and } N_p, \text{ respectively}\} \). (Figure 7)

\( \text{minDist}(c_1, N_p) \)

Figure 7: Example of \( \text{minDist} \)

The following theorem guarantees that the subtrees of the nodes in \( \{N_c | \text{minDist}(N_c, N_p) < \text{maxDist}(N_c)\} \) cover all the clients in \( \text{IS}(N_p) \). This set defines \( \text{IN}(N_p) \).

**Theorem 1.** Given two nodes \( N_c \) and \( N_p \), \( \text{sub}(N_c) \cap \text{IS}(N_p) = \emptyset \) if \( \text{minDist}(N_c, N_p) \geq \text{maxDist}(N_c) \).

**Proof.** According to the definitions of \( \text{minDist} \) and \( \text{maxDist} \), we have:

1. \( \forall c \in \text{sub}(N_c) \forall p \in \text{sub}(N_p) : \text{dist}(c, p) \geq \text{minDist}(N_c, N_p) \);
2. \( \forall c \in \text{sub}(N_c) : \text{maxDist}(N_c) \geq dnn(c, F) \).

Hence, if \( \text{minDist}(N_c, N_p) \geq \text{maxDist}(N_c) \), we can obtain:

\[ \forall c \in \text{sub}(N_c) \forall p \in \text{sub}(N_p) : \text{dist}(c, p) \geq dnn(c, F). \]

Therefore, \( \text{sub}(N_c) \cap \text{IS}(N_p) = \emptyset \).

**Theorem 2.** The following expression defines an upper bound for the \( dr \) values of all data points indexed in the subtree rooted at a node \( N_p \) of \( R_p \):

\[ \sum_{c \in \text{IN}(N_p)} \{ |\text{sub}(N_c)| : \text{maxDist}(N_c) - \text{minDist}(N_c, N_p) \} \]

**Proof.** An implicit statement about the definition of \( dr(p) \) (cf. Section 4.1) for a potential location \( p \) is that \( dnn(c, F) > \text{dist}(c, p) \). However, this may not be true if \( \text{IN}(N_p) \) is used instead of \( \text{IS}(p) \) when \( c \in \text{sub}(N_c) \) and \( N_c \in \text{IN}(N_p) \) but \( c \notin \text{IS}(p) \). Let us define a set \( S_p(N_c) \) which contains the clients in \( \text{sub}(N_c) \) who are also in \( \text{IS}(p) \). Formally,

\[ S_p(N_c) = \{ c | dnn(c, F) - \text{dist}(c, p) > 0, c \in \text{sub}(N_c) \text{, } N_c \in \text{IN}(N_p) \} \]

We can see that \( S_p(N_c) \subseteq \text{sub}(N_c) \). Take advantage of the following relationships.

1. \( \forall p \in \text{sub}(N_p) : \{ c | c \in \text{sub}(N_c), N_c \in \text{IN}(N_p) \} \supseteq \text{IS}(N_p) \supseteq \text{IS}(p) \);
2. \( \forall c \in \text{sub}(N_c) : \text{maxDist}(N_c) \geq dnn(c, F) \);
3. \( \forall c \in \text{sub}(N_c) \forall p \in \text{sub}(N_p) : \text{minDist}(N_c, N_p) \leq \text{dist}(c, p) \).

For any potential location \( p \in \text{sub}(N_p) \), we obtain:

\[ dr(p) = \sum_{c \in \text{IS}(p)} [\text{dnn}(c, F) - \text{dist}(c, p)] = \sum_{c \in \text{IN}(N_p) \cup \text{sub}(N_c)} \{ |\text{sub}(N_c)| : \text{maxDist}(N_c) - \text{minDist}(N_c, N_p) \} \]

Thus, the upper bound holds.
To compute $maxdr$ in the process of traversal, each entry $e_2^p$ of $R_P^c$ stores $\text{sub}(N^c_P)$ and $\text{maxFDist}(N^c_P)$, denoted as $cNum$ and $\text{maxFDist}$, for its child node $N^c_P$. For the data entries, $cNum = 1$ and $\text{maxFDist} = \text{dnn}(e_2^c, F)$. Recursively from the leaf nodes to the root node, the values of the two new attributes can be computed for each entry of the non-leaf nodes based on their definitions. When there is an update in $R_P^c$, the structure can be maintained efficiently in a similar recursive manner. Tree $R_P^c$ can also be used in processing conventional queries on an R-tree efficiently, since adding two attributes will not significantly increase the height of the tree. This will be validated in our cost analysis and the experiments.

4.4 A Lower Bound

The way we derive the lower bound $\text{mindr}$ is similar to that of deriving $\text{maxdr}$. We define $\text{mindr}(N_P)$ in the form of $\text{max} \{g_1(N_C) - g_2(N_C,N_P)\} | N_C \in S \}$, where $S$ is a set of client R-tree nodes, $g_1$ is a metric related to a node $N_C$, and $g_2$ is a metric related to $N_C$ and $N_P$. We use the maximum value instead of the sum value because it is possible for the $IS(p)$ of a potential location $p$ to be indexed in only one subtree rooted at a node of $R_P^c$. We use $\text{maxFDist}$ as $g_1$, and $\text{minExistDNN}$ as $g_2$. $\text{IN}(N_P)$ is used as $S$. Metric $\text{minExistDNN}$ is defined based on $\text{maxFDist}$. These two metrics are proposed by Roussopoulos et al. (1995) and Xia et al. (2005), respectively. We present their definitions before presenting the definition of $\text{mindr}$.

Definition 5. The minimum upper bound of the distance between a point and its nearest data point $o$ in another MBR ($\text{minMaxDist}$).

$\text{minMaxDist}(c, N_P)$ denotes the minimum upper bound of the distance from a client $c$ to her nearest potential location in $\text{sub}(N_P)$. It is the distance between $c$ and the second nearest corner of $N_P$. $\text{minMaxDist}$ since there must be a potential location $p$ on the side joining the nearest and the second nearest corners, and the distance between $c$ and $p$ must be equal to or less than $\text{minMaxDist}(c, N_P)$. Figure 9

![Figure 9: Example of minMaxDist](image)

Definition 6. The minimum upper bound of the distance between a point in the MBR of some node $N_1$ to its nearest data point $o$ contained in the MBR of another node $N_2$ ($\text{minExistDNN}$).

We use $\text{minExistDNN}(N_C, N_P)$ to denote the minimum upper bound of the distance between a point $o$ within the MBR of $N_C$, $N_C.mbr$, and its nearest potential location $p \in \text{sub}(N_P)$. Formally,

$\text{minExistDNN}(N_C, N_P) = \max \{\text{minMaxDist}(o, N_P) | o \in N_C.mbr\}.$

Xia et al. (2005) propose a method to efficiently compute $\text{minExistDNN}(N_C, N_P)$. As shown in Figure 10, if we draw the four perpendicular bisectors of $N_C.mbr$’s edges and diagonals, they intersect $N_C.mbr$ at eight points. These points are called the intersection points (Xia et al. 2005) of $N_C$. Xia et al. prove that for any point $o \in N_C.mbr$, there is a corner point or an intersection point $o’$ of $N_C.mbr$, such that $\text{minMaxDist}(o’, N_P) \geq \text{minMaxDist}(o, N_P)$. As a result, the computation of $\text{minExistDNN}(N_C, N_P)$ requires checking at most twelve points.

Now we have the following theorem to define and prove the lower bound $\text{mindr}$.

Theorem 3. The following expression defines a lower bound for the $dr$ value of all data points indexed in the subtree rooted at a node $N_P$ of $R_P$.

$$\text{max} \{\text{maxFDist}(N_C) − \text{minExistDNN}(N_C, N_P) | N_C \in \text{IN}(N_P)\}.$$

Proof. The definition of $\text{minExistDNN}$ implies:

$$\forall c \in \text{sub}(N_C) \exists p \in \text{sub}(N_P) : \text{minExistDNN}(N_C, N_P) \geq \text{dist}(c, p) \Rightarrow \forall c \in \text{sub}(N_C) \exists p \in \text{sub}(N_P) : -\text{minExistDNN}(N_C, N_P) \leq -\text{dist}(c, p).$$

Also, $\exists c \in \text{sub}(N_C) : \text{maxFDist}(N_C) = \text{dnn}(c, F)$

Hence, $\exists c \in \text{sub}(N_C) \exists p \in \text{sub}(N_P) : \text{maxFDist}(N_C) - \text{minExistDNN}(N_C, N_P) \leq \text{dnn}(c, F) - \text{dist}(c, p) \leq \text{dr}(p)$.

Therefore, the lower bound holds.

4.5 Discussion

Let us revisit the BB method. Its core idea is that a depth-first traversal is performed on $R_P$ while a global lower bound $pd$ is used to prune the subtrees. The pruning distance $pd$ is updated once $\text{mindr}$ of some node or $dr$ of some potential location is found to be larger than it. The pruning power relies on the fast increase of $pd$. The reason why we do not use a best-first traversal is that $\text{mindr}$ is rather small. Thus the main reason for $pd$ to increase is the update of newly found larger $dr$ value. If we use a best-first traversal, $dr$ value will not be found until the traversal reaches the leaf nodes of both trees, which means almost all non-leaf nodes may end up staying in the active page list waiting to be pruned. The space requirement for this process is too high. Hence, we opt to use the depth-first traversal.

5 Cost Analysis

In this section, we analyze the I/O cost and CPU cost of the BB method and compare them with those of the SS method.

We first introduce the notation and equations used in the analysis. We assume an R-tree node has the size of
a disk block. Let $C_m$ be the maximum number of entries in a disk block (i.e., $C_m = \text{block size} / \text{size of a data entry}$) and $C_r$ be the effective capacity of an R-tree node, i.e., the average number of entries in an R-tree node. Then the average height of an R-tree, $h$, is computed as $\left\lceil \log_{C_r}(n) \right\rceil$, where $n$ is the cardinality of the dataset (we denote the cardinalities of $C, F, \text{ and } P$ by $n_C, n_F, n_P$, respectively). The expected number of nodes in an R-tree is the total number of nodes in all tree levels (leaf nodes being level 1 and the root node being level $h$), which is $\sum_{i=1}^{h} \frac{n}{C_r^{i-1}}$.

**I/O cost:** For the SS method, the data points are retrieved in blocks from the disk, and the I/O cost is $I/O = \frac{n_P}{C_m} \frac{N_C}{C_m} = \frac{n_P N_C}{C_m^2}$. For the BB method, the I/O cost depends on the number of R-tree nodes accessed. In the method, $R_P$ is traversed in a depth-first order and for every node $N_P$ of $R_P$, we need to retrieve the nodes in the client R-tree that satisfies certain conditions with $N_P$. In the worst case, every node of $R_P$ is traversed, and for every node of $R_P$, the whole client R-tree is traversed. Therefore, the worst-case I/O cost is $I/O = \frac{n_P}{C_m} \frac{n_C}{C_m} = \frac{n_P n_C}{C_m^2}$. While this worst-case I/O cost is worse than the I/O cost of the SS method because $C_r < C_m$, in practice, many nodes of the R-trees are pruned during the traversal. We quantify the percentages of the pruned nodes in $R_P$ and $R_C$ as the pruning power, and denote them by $w_P$ and $w_C$, respectively. Then we have the average I/O cost of the BB method, $I/O_b = (1 - w_P)(1 - w_C) \frac{n_P n_C}{C_m^2}$. The superiority of the BB method over the SS method lies in $w_P, w_C$. In our performance study, we will show that the pruning techniques used in the BB method are effective and $I/O_b$ is constantly much less than $I/O_s$.

**CPU cost:** The CPU cost can be considered as the product of the CPU cost per disk block (R-tree node) multiplied by the number of disk blocks (R-tree nodes) accessed. The I/O cost analysis provides the number of nodes accessed. The CPU cost per disk block (R-tree node), typically involves distance metric computations. For every pair of disk blocks accessed, the SS method computes $dist_c(c, p)$ for every pair of client $c$ and potential location $p$. So there are $C_m^2$ distance metric computations. For every pair of R-tree nodes ($N_P, N_C$), the BB method only computes the values of several distance metrics to determine whether $N_C$ should be put in $IN(N_P)$ for further process. Thus, the BB method has a much smaller number of distance metric computations to process a pair of R-tree nodes than that of the SS method to process a pair of disk blocks. We have also shown that on average, the BB method has a much smaller I/O cost than that of the SS method. Therefore, the CPU cost of the BB method is much smaller than that of the SS method.

### 6 A Performance Study

In this section, we report the results of our performance study. Experimental setting is presented in Section 6.1. To evaluate the performance of the proposed method under different environments, we use the R-tree (Guttman 1984) and its proposed cardinalities of $C, F$ and $P$, which are 15206, 3008 and 3009, respectively, to 1000K. Three types of datasets are used: (i) Uniform datasets, where data points are generated randomly according to a uniform distribution; (ii) Gaussian datasets, where data points follow the Gaussian distribution; (iii) Zipfian datasets, where data points follow the Zipfian distribution. The parameters used in the experiments on synthetic datasets are summarized in Table 2, where values in bold denote default values.

We adopt two groups of real datasets provided by Digital Chart of the World (RtreePortal 2011), which contain the points of populated places and cultural landmarks in the US and in North America. We name them as the US group and the NA group, respectively. For each group of datasets, the populated places are used as the client set $C$. The cultural landmark dataset is divided into two datasets. Half of the cultural landmarks are chosen randomly to form the existing facility set $F$, and the remaining are used as the potential location set $P$. For the US group, the cardinalities of $C, F, P$ are 15206, 3008 and 3009, respectively, while those for the NA group are 24493, 4601 and 4602. We use the R-tree (Guttman 1984) and its proposed variant as the underlying access methods.

### 6.1 Experimental Setup

All experiments were conducted on a personal computer with 3GB RAM and 2.66GHz Intel(R) Core(TM)2 Quad CPU. The disk page size is 4K bytes, and no buffer is used. We measure the running time and the number of I/Os.

We conduct experiments on synthetic and real datasets. Synthetic datasets are generated with a space domain of $1000 \times 1000$. The dataset cardinalities range from 0.1K to 1000K. Three types of datasets are used: (i) Uniform datasets, where data points are generated randomly according to a uniform distribution; (ii) Gaussian datasets, where data points follow the Gaussian distribution; (iii) Zipfian datasets, where data points follow the Zipfian distribution. The parameters used in the experiments on synthetic datasets are summarized in Table 2, where values in bold denote default values.

We adopt two groups of real datasets provided by Digital Chart of the World (RtreePortal 2011), which contain the points of populated places and cultural landmarks in the US and in North America. We name them as the US group and the NA group, respectively. For each group of datasets, the populated places are used as the client set $C$. The cultural landmark dataset is divided into two datasets. Half of the cultural landmarks are chosen randomly to form the existing facility set $F$, and the remaining are used as the potential location set $P$. For the US group, the cardinalities of $C, F, P$ are 15206, 3008 and 3009, respectively, while those for the NA group are 24493, 4601 and 4602. We use the R-tree (Guttman 1984) and its proposed variant as the underlying access methods.

### 6.2 Experiments on Uniform Datasets

This subsection focuses on the effect of dataset cardinalities. We vary the sizes of $C, F$ and $P$ independently.

#### 6.2.1 Varying the Number of Clients

The results for the experiments that vary the number of clients are shown in Figure 11. From this figure, we can see that the BB method outperforms the SS method by almost ten times in terms of both the running time and the number of I/Os. This is because of the pruning techniques used by the BB method to reduce the search space for the
query answer, and this result confirms our cost analysis, where the average cost of the BB method is shown to be much smaller than that of the SS method.

We also see that even with a small set of clients (10K), it takes the SS method seconds to process the query. Considering the capability of human perception, 0.1 seconds may be a preferable choice for processing a query (Morse 1996). Then the SS method is far inferior and is unable to produce the query answer in time, especially for the urban development simulations and the MMOG applications. As for the BB method, it computes the query answer within seconds. With some upgrades in hardware, it is still realistic for the BB method to produce the query answer in time.

6.2.2 Varying the Number of Existing Facilities

Figure 12: The effect of existing facility set cardinality

The results of the experiments varying the number of existing facilities are shown in Figure 12. Again, in this figure, the BB method shows much better performance than the SS method in terms of both the running time and the number of I/Os because of the pruning techniques used to reduce the search space.

Another observation is that an increase in the number of facilities yields a drop in both the running time and the number of I/Os for the BB method. The reason is that on average the more the facilities, the shorter the nearest facility distances for the clients. In other words, \( d_{mn}(c, F) \) decreases with the increase of the number of existing facilities. As a result, \( \text{maxFDist}(NC) \) decreases and the pruning power of the BB method to prune nodes in \( R_F \) is enhanced. Therefore, the number of I/Os and running time are reduced. SS is not affected due to its lack of pruning capability and it does not access the set of \( F \) (it accesses \( F \) for \( d_{mn}(c, F) \) computation, which is assumed to be pre-computed).

6.2.3 Varying the Number of Potential Locations

Figure 13: The effect of potential location set cardinality

Results of the experiments that vary the number of potential locations are shown in Figure 13. The BB method still shows high efficiency in these experiments.

We observe that, generally, the growth in the number of potential locations has the similar effect on the running time and the number of I/Os as increasing the number of clients. We also notice that, as the number of potential locations increases, the running time and the number of I/Os of the BB method increase much slower than those of the SS method do (please note the logarithmic scale). This is because when the number of potential locations becomes larger, the height of \( R_F \) increases and every time a non-leaf node in \( R_F \) is pruned, more potential locations are pruned. When the number of potential locations \( n_p \) becomes very large (i.e. \( n_p > 10K \)), the proposed pruning techniques function even better and the advantage of the BB method becomes significant.

6.3 Experiments on Gaussian Datasets

Figure 14: The effect of \( \sigma^2 \) in Gaussian distribution

In the following experiments, we vary the distribution of the datasets.

Figure 14 shows the results of experiments conducted on the Gaussian datasets where we vary the value of \( \sigma^2 \). For the Gaussian datasets, varying \( \sigma^2 \) means varying the degree of the inclination for the data points to cluster at the central area of the distribution. Increasing \( \sigma^2 \) leads to less dense data points at the center. We observe that, compared with varying dataset cardinalities, varying \( \sigma^2 \) does not affect much of the algorithm performance. The BB method still outperforms the SS method in terms of both the running time and the number of I/Os, which confirms the results of our cost analysis.

6.4 Experiments on Zipfian Datasets

Figure 15: The effect of \( \alpha \) in Zipfian distribution

We vary the value of \( \alpha \) in the experiments conducted on the Zipfian datasets and the results are shown in Figure 15. Like the Gaussian datasets, we notice that the value of \( \alpha \) does not affect much of the algorithm performance. We also notice that the resultant running time and number of I/Os are similar to those of the experiments conducted on the Gaussian datasets. We further compare these results with those of the experiments conducted on the uniform datasets with the same dataset cardinalities,
and find that the differences among them are small, too. Thus, we can conclude that the effect of different distributions on the proposed method is trivial.

### 6.5 Experiments on Real Datasets

![Figure 16: Performance comparison on real datasets](image_url)

The experimental results on real datasets are shown in Figure 16. The comparative performance of the methods is similar to that of experiments conducted on the synthetic datasets. The BB method still outperforms the SS methods significantly for both US and NA datasets.

Overall, we see that the BB method outperforms the SS methods constantly because of the pruning techniques used to reduce the search space for the query answer. When the dataset cardinalities become large, the advantage of the BB method becomes more significant. These results agree with our cost analysis.

### 7 Conclusions

We conducted a comprehensive study on processing the min-dist location selection query. We first analyzed the basic properties of this query type and presented a naive algorithm (SS) to process the query. However, the SS algorithm is inefficient due to repeated scanning on datasets. We explored geometric properties of spatial data objects, and proposed techniques to prune the search space. This resulted in a branch and bound algorithm (BB). We provided a detailed comparative cost analysis for the BB method and performed extensive experiments to evaluate the empirical performance of the method. The experimental results show that the BB method constantly outperforms the SS method, and when the dataset cardinalities become large, the advantage of the BB method becomes more significant. As future work, we will compare the BB method with the methods proposed by Qi et al. (2012) both analytically and experimentally.

### Acknowledgment

This work is supported by the Australian Research Council’s Discovery funding scheme (project numbers DP0880250 and DP0880215).

### References


Mouratidis, K., Papadias, D. & Papadimitriou, S. (2005), Medoid queries in large spatial databases., in ‘SSTD’.


Energy Efficiency for MapReduce Workloads: An In-depth Study

Boliang Feng\textsuperscript{1}, Jiaheng Lu\textsuperscript{1}, Yongluan Zhou\textsuperscript{2}, Nan Yang\textsuperscript{1}

\textsuperscript{1} School of Information and DEKE, MOE Renmin University of China, 59 Zhong Guan Cun Avenue, Beijing, PR China 100872, Email: {blfeng, jiahenglu, yangnan}@ruc.edu.cn
\textsuperscript{2} Department of Mathematics and Computer Science University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Email: zhou@imada.sdu.dk

Abstract

Energy efficiency has emerged as a crucial optimization goal in data centers. MapReduce has become a popular and even fashionable distributed processing model for parallel computing in data centers. Hadoop is an open-source implementation of MapReduce, which is widely used for short jobs requiring low response time. In this paper, we conduct an in-depth study of the energy efficiency for MapReduce workloads. We identify four factors that affect the energy efficiency of MapReduce. In particular, we make experiments over four typical MapReduce workloads that represent different kinds of application scenarios and measure the energy consumption with varied cluster parameters. Our key finding is that with well-tuned system parameters and adaptive resource configurations, MapReduce cluster can achieve both performance improvement and good energy saving simultaneously in some instances, which is surprisingly contrast to previous works on cluster-level energy conservation.

Keywords: Energy Efficiency, Performance, MapReduce, Hadoop, Data Center

1 Introduction

Growing demands for large-scale data storage and processing have shifted the bulk of workloads to data centers that typically employ thousands of servers (Yahoo 2008). Energy efficiency is a crucial optimization goal in such large clusters. An early report (EPA 2007) estimated that in US the data centers accounted for roughly 61 billion kilowatt-hours (kWh) (1.5% of the total U.S. electricity consumption) in 2006 at a total cost of 450 million. The number is expected to be doubled by 2011 (Koomey 2008). Furthermore, inadequate approaches of energy management not only result in reduced business competitiveness but also decrease the system reliability. For example, without adequate resource configuration, most servers would be idle most of the time.

Most recent works on energy conservation of larger clusters (Chase et al. 2001)(Elnozahy et al. 2002)(Rajamani & Lefurgy 2003) focused on exploiting low utilization periods and attempted to temporarily switch off some of the servers to reduce the cluster’s energy consumption. This type of approaches improve the cluster’s energy efficiency by dynamically adjusting the active server set according to the current workload. However, their energy savings are always at the expense of performance degradation since the computational resource is limited by such cluster-level strategy.

On the other hand, MapReduce (Dean & Ghemawat 2004) has recently been emerged as a promising programming model for parallel computing in data centers. Unfortunately, energy efficiency has attracted little attention in the design of most existing MapReduce platforms, such as Hadoop (Hadoop 2007). We believe that there would be great opportunities to develop more energy-efficient MapReduce clusters since original MapReduce project did not give considerable attentions on energy efficiency.

Actually, the MapReduce programming model imposes new challenges on cluster-level energy conservations. First, existing MapReduce platforms, such as Hadoop, usually perform automatic parallelization and distribution of computations. The aforementioned energy conservation techniques based on dynamic resource reconfiguration are in conflict with the intention of this design. Obviously, turning off a part of servers will have a negative impact on the parallelization and load balance of the cluster.

Second, the cluster configuration should take both computing performance and data storage requirements into account in a distributed system, which complicates the trade-off as well. In addition, MapReduce also provides services for other workloads such as web service and data analysis. Furthermore, MapReduce incorporates mechanisms to be resilient to failures for instance, machine crashes and software failures. Such mechanisms may make negative affects on energy efficiency.

Inspired by the above observations, we conduct an in-depth energy efficiency study of MapReduce workloads on Hadoop, which is a popular open-source MapReduce platform. We aim to address the following two questions. (1) Which factors affect the cluster-wise energy efficiency of a MapReduce platform? (2) Is there any opportunity to perform trade-off between the energy saving and the performance in a MapReduce platform?

To answer the first question, we consider the impact of the architectural design of MapReduce and identify four factors that affect the energy efficiency of MapReduce: CPU intensiveness, I/O intensiveness, replica factors of the underlying distributed file system as well as the file block size. The last two factors are storage-independent parameters. With regard to
the second question, we elaborately design experiments for an in-depth study. We identify four typical workloads of MapReduce that present different kinds of application scenarios by measuring the energy consumption with varied disparate cluster scales and other related factors.

The main experimental findings can be summarized as follows:

1. For CPU intensive and some map-only workloads, such as WordCount and GrepSearch, the key points to minimize energy consumption are to guarantee load balance and the adequacy of CPU resource. Our experiments show that with good load-balance strategies and resource configurations, the energy saving for GrepSearch is 13.0% with response time reduced by 19.1%.

2. We observe that the default merge and grouping algorithm of MapReduce is not efficient in this two cases: (1) I/O intensive workloads, (2) workloads with low map/reduce ratio. Due to the lack of content-aware reducer assignment policy, the method to provide higher I/O throughput by adding more servers cannot always bring energy saving. The experimental results show that TeraSort with 10GB data set obtain less energy saving with the increase of the cluster size in certain cases.

3. While MapReduce is independent of the underlying distributed file system, the two factors: replica factor and block size are both important for energy efficient MapReduce. In general, relatively higher degree of replica factors would always save more energy in most instances. Furthermore, our results also show that GrepSearch with 20GB data set and 512MB block size can obtain 36.8% energy saving in comparison to default setting.

4. Due to the lack of content-aware reducer assignment policy, some methods for improving parallelism cannot obtain significant performance or energy saving improvement for low map/reduce ratio’s workload like TeraSort. However, for map-only jobs or high map/reduce ratio’s jobs (e.g., GrepSearch, WordCount), our results show that well-tuned MapReduce clusters can reduce energy consumption by 20.0%.

In summary, the MapReduce cluster can, in some cases, achieve performance improvement when the energy consumption is optimized. That is, the two optimization goals: performance and energy saving are coincident in some scenarios. And our results show that well-tuned MapReduce clusters can help saving energy significantly.

The remainder of this paper is organized as follows: Section 2 discusses the factors that affect the energy efficiency of MapReduce. In Section 3, we present the experiment design. An in-depth analysis of experimental results and recommendations are presented in Section 4. We briefly review related work in Section 5. Section 6 concludes the paper with future directions.

2 Factors Affecting Energy Efficiency of MapReduce

In this section, we start our analysis by briefly reviewing the MapReduce programming model and the execution flow of a MapReduce job and then conduct an energy efficiency profile for MapReduce.

2.1 MapReduce Preliminary

MapReduce is a programming framework and an associated implementation for processing and generating large data sets. It was originally developed by Google and now MapReduce programs are becoming popular applications in data center at companies like Yahoo!, IBM and Facebook. It has several prominent features in distributed data management. Its inherently parallel can simplify the complexity of running distributed data processing functions across multiple nodes in a cluster, since MapReduce allows a programmer with no specific knowledge of distributed programming to create his MapReduce functions running in parallel across multiple nodes in the cluster. Also, MapReduce platform can offer fault tolerance that is entirely transparent to programmers, which means it can run on clusters with cheap commodity machines. And in Yahoo!, the MapReduce program is running on cluster with up to 10000 cores (Yahoo! 2008).

The MapReduce programming model divides a program into tasks, of which there are two types: map() and reduce(). Both the two functions are defined with respect to data structured in <key, value> pairs. The Map function is applied in parallel to take one pair of data with a type in the input dataset and then produces a list of intermediate pairs for Reduce call. After that, the MapReduce framework collects all pairs with the same key from all lists and groups them together, thus creating one group for each one of the different generated keys. The Reduce function is then applied in parallel to each group, which in turn produces a collection of values. Each Reduce call typically produces either one value or an empty return. The returns of all calls are collected as the desired result list:

\[
\text{Map}(k_1, v_1) \rightarrow \text{list}(k_2, v_2); \quad \text{Reduce}(k_2, \text{list}(v_2)) \rightarrow \text{list}(v_3)
\]  

The node of jobtracker divides the input data of one job into appropriate size splits and the framework assigns one split to each Map function, which is known as job initialization. For a map task, jobtracker takes account of the load balance, parallelism and the tasktracker’s network location of the system, picking a task whose input split is as close as possible to the tasktracker. Once the map function is completed, jobtracker simply takes the next in list of yet-to-be-run reduce tasks as the reducer. And intermediate data will be shuffled to different reduce tasks according to their values. Overall, MapReduce are executed in four stages: job initialization, map process, shuffle and reduce process.

Hadoop is a successful implementation of the MapReduce framework. Optimizing replica placement distinguishes HDFS (HDFS 2007) from most other distributed file systems. Generally, each file will have 3 replicas in default to improve data reliability, availability, and network bandwidth utilization. For our work, we select Hadoop as the implementation of MapReduce.

2.2 Energy Efficiency Factors

In this section, we consider the impact of the architectural design of MapReduce and identify four factors that affect the energy efficiency of MapReduce: CPU

\footnote{Low map/reduce ratio means that the job’s most execution time is spend on map stage.}
intensiveness, I/O intensiveness, replica factors and block sizes.

2.2.1 Energy Consumption Model for MapReduce.

We present a mathematical model for energy analysis of MapReduce workloads in Formula (2). This model gives an overall view of the energy consumption during the process of a MapReduce job.

\[
\text{Energy} = PT = P_i T_i + P_{m} T_m + P_r T_r + P_s T_s
\]

where, energy is equal to power \( P \) multiplies time \( T \). \( P_i T_i \) is the energy consumption of job initialization. \( P_m T_m \) and \( P_r T_r \) represent the energy consumption during the map stage and reduce stage respectively. And we use \( P_s T_s \) to denote the energy consumption of intermediate data shuffling. Based on this model, there are four factors that affect the total energy consumption of a MapReduce job.

2.2.2 CPU Intensiveness Workloads

For the CPU intensive MapReduce workloads (e.g., WordCount), the power consumptions of map and reduce phases are high (Tsirogiannis et al. 2010). However, from the performance standpoint, the intuitive idea to add more CPU resources by increasing the number of servers can decrease the running time. On the other hand, this way would increase the power consumption of the cluster. Therefore, a good balance of this trade-off needs theoretical analysis and experimental testing, since energy consumption is equal to the product of power and time.

2.2.3 I/O Intensiveness Workloads

For I/O intensive MapReduce workloads, partial tasks should wait for the data to be ready for processing. Meanwhile, the idle CPUs may lead to the waste of energy. That is not energy efficient and implies longer response time especially for the map and reduce phases. The inherent parallel of MapReduce programming model can take good use of distributed resources in map stage, which can reduce the time of map by well-tuned parameters. On map side, HDFS’s rack-aware replica placement policy also helps improve the I/O efficiency and system throughput by distributing the stresses on I/O across all the nodes. In the current implementation of Hadoop, MapReduce can not produce a fine-grained reduce tasks assignment policy to avoid large size of intermediate data shuffle. According to Equation (2), there would be large energy consumption \( (P_s T_s) \) for such workloads. Therefore, the best way to provide higher I/O throughput is to add more servers.

2.2.4 Replica Factor.

Although the MapReduce programming model is designed to be independent of storage system, HDFS is closely related to MapReduce processing. HDFS’s rack-aware replica placement policy that improves data reliability, availability, network bandwidth utilization and system throughput, distributes 3 block replicas across the cluster.

If we do not take storage resources into account, then, intuitively, higher degree of replica factors will provide more choices for MapReduce tasks assignment, which improves load balance and parallelism of the system and decrease the response time \( T_m \) and \( T_r \). In Section 4, we will experimentally investigate if this intuition is true and how data replication affects the energy efficiency of MapReduce workloads and discuss the opportunity for replica placement optimization.

2.2.5 Block Size.

Block Size is another important factor of the implementation of Hadoop, which makes affect on energy efficiency for MapReduce as well. For MapReduce, Hadoop divides the input to a MapReduce job into fixed-size pieces (usually equals to block size) called input splits. Hadoop creates one map task for each split. Having many splits means the time \( (T_m) \) taken to process each split is small compared to the time processing the whole input. And, the process would be better load-balanced as the splits become more fine-grained. On the other hand, if splits are too small, then the overhead of managing the splits and of creating map tasks \( (P_i T_i) \) begins to dominate the total job energy consumption. This characteristics’ effect on energy efficiency for MapReduce workloads will also be discussed in Section 4.

3 Experimental Design

In this section, we introduce the performance metrics, appropriate workloads and the cluster setup in our energy efficiency studies of MapReduce, which is important for the analysis of software performance.

3.1 Metrics

Metrics are critical for our analysis on energy efficiency characteristics of MapReduce workloads. In this paper, we consider both energy consumption and performance. Generally, the energy cost in data centers is measured in kWh (kilo*Watt*hour) or Wh (Watt*hour). The performance, namely execution time, is a key issue for data center as users are “spoiled” and have not the patience for a long delay in the execution of their jobs in many scenarios. In addition, the degree of replica factor is also a key parameter in distributed file system. Therefore, a bundle of metrics are collected including response time, replica factor, number of machines, and energy consumption. These metrics can help us understand the relationship between the energy cost and the response time.

To measure the overall Energy Efficiency (EE) of a system, we use the following equation:

\[
\text{EE} = \frac{\text{Size of workload}}{\text{Energy} \times \text{response time}}
\]

This metric takes into accounts both the energy consumption and the execution time with the varying sizes of workloads.

3.2 Cluster Setup

We used a small cluster of six nodes on high-rate network traffic feed. Each datanode/tasktracker with a 2.4GHZ Intel Core Duo processor running Linux 2.6.31, 2GB RAM and 500GB SATA disk.

The master node that hosts the namenode and jobtracker is equipped with a Quad-core processor, 4GB RAM. All the nodes are connected to each other via 1000Mbps ethernet ports. In the average, the disk performance that we measured is approximately 100MB/s. This node consumes 41W at idle, 67W at
peak utilization, 1.6W at hibernate. We measured the energy consumption for the whole system. And, system energy is measured by using a plug-in power meter, whose minimal accuracy is 1Wh.

We ran Hadoop version 0.20.2. and java version 1.6.0. The property mapred.child.java.opts is set to "-Xmx1024m" and io.sort.mb is increased to 250. Other relevant parameters such as number of mappers and reducers per job are defined as follows. The number of mapper equals to the number of splits of the input files. In our experiments it is around 100 mappers/node. And the number of reducer is set around to 0.9 multiply the reduce task capacity, which will change according to the cluster scale.

3.3 Workloads
To comprehensively analyze the energy efficiency of MapReduce, we select the following four typical benchmarks.

1. TextWrite: This program writes a large unsorted random sequence of words from a word-list. This job is network-intensive.

2. GrepSearch: This program is a map-only CPU intensive job by matching regular expressions from input files. It is high CPU utilization in map stage.

3. TeraSort: It is essentially a well-known benchmark, sorting the official input datasets. It is CPU bound in the map stage and I/O bound in the reduce stage.

4. WordCount: This program is a balance between CPU-intensive and I/O intensive jobs. It reads the text input files, breaks each line into words and distributes them to multiple machines and finally counts them.

In order to measure the system performance under different load pressures, we run each job with three different data sizes from 1.5GB to 20GB. And the replica factor is varied from 1 to 5. The block size ranges from 16MB to 1GB. For each job, we run three trials to find the average energy consumption and response time.

4 Analysis of Results
Our results suggest that energy saving and performance are not two divided optimization goals in some cases. In this section we analyze the experimental results. In general, our results come in contrast to recent work (Leverich & Kozyrakis 2010), which suggested that energy efficiency would already be at the cost of performance degradation. Our results suggest that energy saving and performance are not two contradict optimization goals in some cases.

4.1 TextWrite
HDFS creates multiple replicas (the default is 3) of data blocks and spreads them throughout a cluster to enable reliable, extremely rapid computations. We use map-only workload TextWrite to write large data files with varied workload size, cluster scale, replica factor and block size. It is network intensive workload for data transfer.

Fig.1 shows the results of this experiment. The replica factor is set from 1 to 5. And, the workload size ranges from 1.5GB to 20GB. As depicted in Fig.1,a, it has almost a linear growth of both latency and energy consumption with the replica factor increasing. Analogously, a larger workload size would result in proportional increase of the response time as well. Obviously, the reason is that higher degree of replica factor means more data should be copied...
It is high CPU utilization in map stage. File with words list and the regular expression is set to use GrepSearch as a map-only and scan-like: job that can obtain both high performance and energy saving. The total energy consumption is the product of time consumption and the response time. We believe this reason is also can be used to explain why the larger the data set is, the smaller the energy E-E is, as shown in Fig.1.c.

We also show the results of TextWrite with varied cluster size and workload size in Fig.1.b (replica factor is the default 3). We see that with more nodes in the cluster, it takes great improvement on the performance, around 51.3%. Meanwhile, energy consumption is decreased from 71Wh to 49Wh. We believe this behavior is reasonable. This can be explained by the fact that increasing the number of nodes means the increase of power consumption of the whole cluster, the response time reduction can trade-off the energy consumption, as the total energy consumption is the product of time and power.

Fig.1.d shows the results with varied block sizes. When the block size is as small as 16MB, the system can obtain both high performance and energy saving. The reason is that small block size enables more tasks to be processed in parallel. And when the block size is larger than 64MB, the parallelism of the system is reduced, so that energy consumption increases significantly.

4.2 GrepSearch

GrepSearch is a map-only and scan-like job that searches files globally for lines matching a given regular expression. In our experiments, the file is a text file with words list and the regular expression is set to be "a" (match the words that begin with "a"). These experiments are also implemented with varied workload size, cluster scale, replica factor and block size. It is high CPU utilization in map stage.

Fig.2 shows the results of the GrepSearch workload. We have several interesting observations. No matter how large the workload is, the worst case always exists when the replica factor is 1. We think this reason is quite simple. Higher degree of replica factor means more choices for tasks assignment, improving the load balancing of the system. In other words, Jobtracker would have more choices to assign Map tasks, taking into account data location and request number of the targeted nodes. And, once the replica factor is larger than 2, the response time and energy consumption are approximately reduced by 19.1% and 13.0% respectively. This also prove that HDF5 replica placement policy not only improve data reliability, availability, but also improve the parallelism of the GrepSearch workload.

Fig.2.b and Fig.2.c present the results of GrepSearch with varied cluster size and workload size. With workload size increasing from 5GB to 20GB, the value of E-E is reduced. We believe that long latency makes irrelevant components run in low utilization and thus waste the energy. On the other hand, more nodes in the cluster means more resources and better performance that make the system more energy efficient. As depicted in Fig.2.b, when the workload size is large (20GB), the system can achieve more energy saving (20.0%) with the cluster size increasing. However, when the workload size is as small as 5GB, the initial cost for each tasks can not be amortized, and resources are sufficient. Thus, there is no obvious energy saving with the increase of the cluster size.

Fig.2.d shows the results with varied block size. When the block size is as small as 16MB, it means around 1280 maps should be initialized. The overhead for the creation of map tasks can not be offset by the superiority of parallel processing. We can see that, by well-tuned block size, the system can obtain energy saving by as much as 36.8% with the response time...
4.3 TeraSort

TeraSort is essentially a sequential I/O benchmark. It is CPU bound in map stage and I/O bound in reduce stage. We set the output file’s replica factor to be 1 to avoid additional network cost. During the reduce stage, CPUs are almost idle, waiting for I/O intensive jobs. Again, we present the results of TeraSort with varied replica factor, workload size, cluster scale and block size in Fig.3.

TeraSort is I/O bound program waiting for memory swapped in both map and reduce stages that making WAIT CPU occupies a large proportion of the total CPU utilization. There is a random fluctuation in the measurements when we run TeraSort with different replica factors. We run this experiment 5 times and calculate the average value, which is showed in Fig.3.a. We can not see significant changes of performance with varied replica factor. We believe this behavior is due to the process of MapReduce framework. The two key factors that affect the energy efficiency of I/O intensive workloads are load balance and reduce tasks assignment. More replicas would improve the load balance of map tasks, but there also would be large data transfer among nodes in the shuffle stage. Meanwhile, a task with high latency will affect the progress of the whole job. Unfortunately, MapReduce could not give a more fine-grained performance optimization, since the process of data transfer are related to the value of the intermediate key-value pairs.

As depicted in Fig.3.b, with varied cluster size and workload size, TeraSort shows a similar result of EE as GrepSearch. A larger workload would increase the job runtime, and more IT components will lead to more energy waste. Generally, the intuitive way to provide higher I/O throughput is to add more servers. So with the expansion of the cluster size, the system achieves energy consumption reduction by 20.2%. Let us take a closer look at the results at the point when the number of nodes is 3. At this moment, the job needs few data transfer after map tasks according to the data location and partition results. That also explains why there is obvious reduction of response time reduced from 1325s to 816s. Fig.3.c shows the value of E-E with varied cluster size and workload size.

Fig.3.d shows the results with varied block size, which are quite different from the results of GrepSearch, because most cost of such workloads comes from the shuffle stage and reduce stage. And, small block size means fine-grained input splits which will improve the performance of reduce sort. When the block size is 1GB, we can see the decreased response time and energy consumption. We believe that 1GB size of splits brings less data shuffle after map processing.

4.4 WordCount

WordCount achieves a balance between CPU- and I/O-intensive workload in both two stages: map and reduce. We set the output file’s replica factor to be 1 to avoid additional network cost. Each map takes a line as input and breaks it into words. It then emits a key/value pair of the word and 1. Each reducer sums the counts for each word and emits a single key/value with the word and sum. It is really energy intensive, employing almost 95% of the potential CPU. We implement this experiment with varied replica factor, workload size, cluster scale and block size.

It is an interesting observation that increasing the degree of replica factor can always improve the system performance. When the replica factor is larger than 2, the system achieves a significant performance improvement by 6.2% with energy consumption reduced by 7.2%. The reason is that larger degree of replica
factors implies more opportunities for load balance, which is good for CPU intensive workloads. Furthermore, WordCount is different from TeraSort. Its two stages map and reduce account for nearly half of the whole process but TeraSort’s map/reduce ratio is around 2/3.

With more nodes added in the cluster, both response time and energy consumption decrease by 56.8% and 18.2% with 20GB data set. We can see that increasing the number of servers can achieve certain energy saving. The increase of energy consumption due to the new nodes can be offset by the reduction of response time.

Fig.4.d shows the results with varied block size. As we excepted, WordCount shows similar behaviors to GrepSearch that small block size brings out high cost for tasks initialization that can not be amortized in the task execution time. On the other hand, a large block size such as 1GB will make negative effects on parallelism of the system, so that we can see the rising of both response time and energy consumption when the block size is 1GB.

4.5 Summary of Experiments

Based on the above experimental results, we give an overall observation combining these four factors in the following. And, we make several suggestions with regard to improving energy efficiency of MapReduce cluster.

4.5.1 CPU Intensive Workloads

For CPU intensive and high map/reduce stage time ratio workloads (e.g., WordCount, GrepSearch), the best way to minimize energy consumption is to guarantee load balance and adequacy of CPU resources, for example, increasing the cluster scale. These methods are also make positive affect on performance. The experimental results show that WordCount with 20GB data set can obtain 14 Wh energy saving when the number of nodes increases from 3 to 5. And, the response time decreases by 56.8%.

There is no surprise that high degree of replica factors will always give more opportunities for load balancing. Meanwhile, it is a great challenge for task scheduler of MapReduce on how to decide which node is the best one for a specific map or reduce task. And we need an appropriate cluster-level reconfiguration strategy for matching the number of active nodes to the current needs of the workload.

4.5.2 I/O Intensive Workloads

For I/O intensive workloads, fine-grained input splits are effective for shuffle and reduce stages, which is more energy efficient on condition that the energy cost for job initialization can be amortized. As MapReduce can not give a content-aware tasks assignment policy to avoid high cost of the shuffle process, larger scale of cluster can not give better parallelism and load balancing in some cases. It is worse for I/O intensive workloads, in which a large size of intermediate data should be shuffled, such as TeraSort. For example, we can see that TeraSort with 10GB data set running on 3 nodes consume less energy (64Wh) than running on 4 nodes (67Wh) in Figure 3. We believe that optimized replica placement policy of HDFS will help the map and reduce tasks assignment for energy efficiency and improved data partitioning algorithms and content-aware reduce tasks scheduling strategies are needed. And the ideal condition is that less data will be shuffled across the cluster before the reduce stage.
4.5.3 Replica Factor.
Most workloads running with higher degree of replica factors would obtain better performance and larger amount of energy saving especially for CPU intensive and high map/reduce ratio workloads. For these jobs, high degree of replica factor implies better load balance of the system, making a positive effect on both performance and energy efficiency. For I/O intensive workloads, this method can not provide significant energy saving.

4.5.4 Block Size.
Both the CPU intensive and high map/reduce ratio jobs show that a large block size will reduce the cost for tasks initialization, which will help improve the performance and energy efficiency of the system. But, for I/O intensive workloads such as TeraSort, a small block size means fine-grained splits, which will make positive effect on data shuffle and reduce process and bring the reduction on both response time and energy consumption.

4.5.5 Key Finding.
It is an interesting observation that the MapReduce cluster would obtain both energy saving and performance improvement in some scenarios. That is, energy saving and performance are not two opposite optimization goals in some cases for energy efficient MapReduce cluster. For example, by the well-tuned cluster setting, Terasort with 2.5GB data set can obtain energy saving by as much as 18.8% and response time decreases from 209s to 118s. And by well-tuned block size, GrepSearch with 10GB data set can obtain energy saving by as much as 36.8% with the response time reduced by 38.6%. And, with more nodes added in the cluster, energy consumption of TeraSort will decrease by 18.8% and response time decrease from 209s to 118s.

4.5.6 Recommendations.
Our findings are summarized into the following recommendations:
1. Workloads should be designed to be heterogeneous, taking advantage of as many as components of the system and allowing tasks on different components to overlap.
2. For write intensive MapReduce workloads, the system should power on nodes as fewer as possible without causing any loss of data dispersion and availability for predicted jobs.
3. For CPU intensive and high map/reduce ratio workloads, appropriate number of servers should be provided based on the workload size, ensuring load balance and adequate CPU resource.
4. For I/O intensive workloads, fine-grained input splits is effective for shuffle and reduce stages, which is more energy efficient on condition that the initialization cost can be amortized.
5. The default value of HDFS replica factor is appropriate for both performance and energy efficiency in most cases. In some cases, a lower HDFS replication level is better, but it is not possible when data durability is important.
6. The value of block size demonstrates a trade-off between the overhead for tasks initialization and parallelism of the system. Generally, the default value is too small for large files.
7. Improved data partitioning algorithms in map stage and content-aware reduce scheduling strategies are very important for energy efficiency, where more efforts on refinements and improvements are needed.

5 Related Work
Previous work on the energy efficiency of the large-scale data processing is committed to make a energy-proportional system (Barroso & Holzle 2007). On the cluster-level energy management, all the works focus on dynamic cluster reconfiguration, aiming to reassign resource of the data center with a principal focus on energy management, and design workload distributed polices to shift the workloads around for meeting power consolidation (Chase et al. 2001) (Elnozahy et al. 2002) (Rajamani & Lefurgy 2003). Researchers seek to make balance among load balance, time duration and energy consumption, or give a extreme method for energy saving, do not take performance into account. However, energy-efficient MapReduce cluster has not attracted enough attentions. Recent work (Leverich & Kozyrakis 2010) attempts to scale down the Hadoop cluster to obtain energy saving, but they did not give comprehensive evaluation in practice. Two broad strategies for MapReduce energy management are compared in (Lang & Patel 2010), and a new strategy called AIS is proposed in their paper. Their results show that the computational complexity of the workload will affect the effectiveness of the two cluster-level strategies, but they do not give further discussion about the MapReduce Workloads. Chen et.al. (Chen et al. 2009) add energy as another performance metric to consider in the design space for computer systems and data centers and study on MapReduce operation variables. They characterize the performance of the Hadoop implementation of MapReduce under different workloads and then give some conclusion. In this paper, we explicitly raise two questions about energy efficient MapReduce workloads, and comprehensively analyze the factors that affect energy efficiency based on the energy consumption model and the MapReduce Framework. Then experiments are elaborately designed for comprehensive analysis on energy-efficient of MapReduce workloads on a cluster view. On the application-level, efforts to develop energy-efficient data processing applications can also be found in (Graefe 2008) (Tsirigoriannis et al. 2010) (Xu et al. 2010). Lang et.al. (Lang & Patel 2009) present proposal for considering energy efficient query processing in a database management system. (Chen, Ganapathi & Katz 2010) analyze how compression can improve performance and energy efficiency for MapReduce workloads. It is a trade-off between IO consumption and CPU consumption. For several workloads, their compression decision algorithm can help provide significant energy saving. However, they do not give a view on cluster-level optimization. (Chen, Ganapathi, Fox, Katz & Patterson 2010) present a statistics-driven workload generation framework that distills summary statistics from production MapReduce traces and realistically reproduces representative workloads for better MapReduce energy efficiency evaluation mechanisms.

On the node architecture, several technologies such as PowerNap (Meisner et al. 2009), Intel’s Speed-
Step and AMD’s PowerNow! can give us fine grained control on energy management. Recent discussions on fundamentally new cluster server design are highly relevant to ideas of cluster-level energy management (J.Hamilton 2009)(Lang et al. 2010)(Vasudevan et al. 2009). Energy-efficiency benchmarks for MapReduce are discussed in (Poess & Namibiar 2008)(Rivoire et al. 2007). Meanwhile, related industrial standard about energy efficient data center has been proposed, such as PUE (PUE 2005) and DCiE (DCiE 2005). Finally, note that, although Hadoop Mapreduce framework has become a popular and even fashionable application in data center, energy efficient MapReduce cluster has not been well studied in the previous works.

6 Conclusion and Future Work

In this paper, we conducted an in-depth energy efficiency study for MapReduce workloads in an open source implementation Hadoop. In order to comprehensively analyze the opportunities for energy saving, we identified four factors that affect the energy efficiency of MapReduce. Then, we chose four typical workloads of MapReduce and measured the energy consumption with varied cluster scales and factors. The experimental results showed that with well-tuned system parameters and adaptive resource configurations, MapReduce cluster can achieve both significant energy saving and performance improvement simultaneously in some instances.

Based on the exciting finding in this paper, in the future, we plan to verify the results of this paper in a larger size of clusters. Furthermore, more benchmarks of MapReduce should be introduced in our experiments, such as web search and machine learning applications. In addition, we intend to investigate the effects of changing other parameters of MapReduce and Hadoop, such as IO parameters like the number of parallel reduce copies, memory limit, and file buffer size. We also intend to work on the content-aware assignment policy for reduce tasks to increase data locality, which is important to increase parallelism of MapReduce clusters.

Acknowledgement

This paper is partially supported by the Research Funds of Renmin University of China (No: 11XNJ003) , NSF China (Project No. 61170011) and Beijing Natural Science Foundation (Project No. 4112030).

References


Chen, Y., Ganapathi, A. S. & Katz, R. H. (2010), To compress or not to compress compute vs. io tradeoffs for mapreduce energy efficiency, in ‘ACM SIGMOD workshop on Green networking’.


J.Hamilton (2009), Cooperative expendable micro-slice servers (cems): Low cost, low power servers for internet-scale services, in ‘CIDR’.


Sentiment classification of blog posts using topical extracts

Zhixin Zhou  Xiuzhen Zhang  Phil Vines

School of Computer Science and Information Technology
RMIT University,
GPO Box 2476 Melbourne VIC 3001 Australia,
Email: {zhixin.zhou, xiuzhen.zhang, phil.vines}@rmit.edu.au

Abstract

Unlike news stories and product reviews which usually have a strong focus on a single topic, blog posts are often unstructured, and opinions expressed in blog posts do not necessarily correspond to a specific topic. This can lead to unsatisfactory performance of sentiment classification. In this paper we report our pilot study on addressing topic drift in blogs. We examine this phenomenon by manual inspection and establish a ground truth. Our annotations have shown that topic drift is indeed very common, with all documents sampled showing a considerable degree of drift, averaging over 80%. The topical sentences are extracted from each post to produce an extract data set. We propose to address the topical drift problem by classifying the blog posts using the sentence-level polarities of topical extracts. We propose and evaluate two models for aggregating the sentence polarities by comparing their performance to that of a popular word-based model. Our preliminary results suggest that topical extracts can provide a concise but more accurate representation of the sentiment polarity of the blog posts. More importantly, sentence-level polarities are potentially a more reliable evidence than word distributions with regard to document polarity prediction.

Keywords: sentence polarity, sentiment classification, topical extract

1 Introduction

The task of opinion mining, or sentiment classification, can be defined as “to find opinions targeting topic X” (Ounis et al. 2006). The blogosphere has attracted a lot of interest in the opinion mining community as blog posts provide a great source of web users’ opinions both in terms of the size of the collection and the coverage of topics. However, unlike product reviews and movie reviews, the blog posts are less structured; and unlike news stories, the blog posts are not subject to an editorial process. The author of a blog post can talk about anything in any style of writing.

Blog authors often start with commenting on another related entity before expressing his/her opinion on the topic of his/her true interest. It is particularly common in the political comments. The following example was extracted from a document on the topic of State of the Union Address by President George W. Bush.

“Though surprised today to find out that Hamas had an unexpectedly large VICTORY in the Palestinian elections, I am not shocked, and such a development should not have been unexpected if you follow politics, and the actions of George Walker Bush since he was selected to be our president back in the year 2000.”

Before attacking the actions of the Bush administration the author commented on another political event, the latter was used merely for contrasting. It is also worth noting that in this example, opposite opinions on two different entities are expressed in one single sentence.

It is also common for a post to have a few lines of description related to the author’s recent activity, but then followed by some reflections on his/her past experience. This is typical in movie and product reviews, where the author first briefly mentions the movie or product, and may then drift away to write about the emotions or memories aroused by the movie or product. The example below from a document on the topic of March of the Penguins typifies this approach.

“This is a must-see documentary about the mating ritual of Antarctic penguins... I sniffled and nodded - I know what that feels like... We received news yesterday that the donor’s ultrasound exam was normal, so we are good to go. Am feeling detached and ambivalent again...”

The author of this post watched the movie The March of the Penguins, and the birth-giving scenes in the movie caused her to reflect upon her own pregnancy in the past. In fact, the majority of this post in about her own experience, and not the movie.

Topic drift may lead to serious problems in sentiment classification. Previous studies on sentiment analysis of blogs have mostly focused on word-based models using the whole document (Ounis et al. 2006, Macdonald et al. 2007). However, the drifting portion of the document would change the word distribution and potentially compromise the accuracy of the classification models. To address this problem, an intuitive solution would be to extract the topical snippets first and perform classification on the topical extracts instead of the original documents. Both Pang (Pang & Lee 2004) and Lloret (Lloret et al. 2010) have done similar studies, but neither has observed consistently better results on the extracts (note that both works extracted the sentences that are topical and subjective). It has been shown that the length of the extract affects the classification accuracy (Pang & Lee 2004), which suggests that poor-quality extracts might be the reason of the counter-intuitive observation. More recent work by McDonald integrates...
Table 1: An outline of the two topics we used in the collection

<table>
<thead>
<tr>
<th>Topic #</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>869</td>
<td>Find opinions worldwide to the cartoons depicting the Muslim prophet Muhammad printed in a Danish newspaper.</td>
</tr>
<tr>
<td>903</td>
<td>Find documents stating opinions about Apple CEO Steve Jobs.</td>
</tr>
</tbody>
</table>

1 The number of posts we kept in the collection for the topic

sentence-level information into the learning process. While his main focus is on a joint-structured model that classifies the documents as well as the sentences at once, a cascaded model is also described which predicts the sentiment labels first and then pass the labels to the document classification model. In his work however, all the sentences in the document are used.

To our best knowledge, no existing work has applied sentiment classification on manual extracts. Because of this, it is not able to tell whether the problems result from quality of the extracts, or the method employed. In order to isolate these effects we have manually extracted relevant sentences. In our study we annotated randomly sampled documents from two different topics used in Blog06 and Blog07 tasks in TREC. The descriptions of these two topics are shown in Table 1. We adopted the document-level labels annotated by NIST judges, and adapted their annotation scheme to suit our case. We labeled each sentence in the documents both by topicality and sentiment orientation.

We propose two approaches to classify the document polarity with topical extracts. Both approaches are based on sentence-level polarities. We contrast their performance with that of a word-based classifier. Our preliminary experiments have shown that one of our approaches has better performance than that of the word-based classifier using the original posts on both topics. We also note that the word-based approach performs better on the extract than the original post on one topic, but has a lower average accuracy on the other. This suggests that sentiment classification on the topical extracts may indeed be more accurate than on the original posts, but models based on the word-distributions may not work well with the extracts. Pang extracted topical and subjective sentences with the minimum-cut algorithm and Lloret used the SUMMA toolkit (Saggion 2008).

We describe our manual annotation process in Section 3, and introduce the classification models in Section 4. The implementation of the classification experiments are detailed in Section 5. After that, we conclude our work and provide an outlook into the future work in Section 6.

2 Related Work

Sentiment classification, or opinion mining, can be done at different levels of natural language structure. Typically it is done at the document level (Turney 2002, Dave et al. 2003, Pang & Lee 2004, Das & Chen 2007). Turney (2002) extracted potentially opinion-bearing phrases from the documents with a POS tagger and calculated the sentiment orientation score for each phrase. The averaged score of the phrases in a document was used to classify the document as either positive or negative. The voting model proposed in this paper is similar to their work in that we used accumulated length of sentences of different polarities, instead of phrase scores, as the indicators of document polarities.

The most similar work to ours is McDonald et al.’s cascaded model briefly described in (McDonald et al. 2007), where sentence polarities are used to influence the document level predictions. However, we use only the topical extracts, while they use the whole document; the sentence polarities are used jointly with other features in their approach, while our approach are based solely on the sentence polarities. We also note that their main focus is a structured model that jointly predicts the document polarity and the sentence polarities at once.

Another piece of work by Pang et al (Pang & Lee 2004) proposed a more fine-grained model which first classifies the sentences by topicality and subjectivity so that only the topical and opinionated portion of the document is used for classification. Their approaches achieved a classification accuracy comparable to that on the original documents, and suggested that the extracts were not only more concise, but also probably “cleaner” representations of the intended polarity. Our study also compared the performance on extracts and original documents with a bag-of-words model, but we extended our scope to more topics. While Pang’s work was based on movie review data, we tested our approaches on a data set on political events and another data set relating to opinions towards a public figure.

Another piece of relevant work that contrasts classification performance on extracts and full documents is described in Lloret et al.’s (Lloret et al. 2010) paper on the problem of rating-inference, which aims to assign numeric rating numbers to the documents. This is different from our work as the documents in our study are associated with either positive or negative label.

The voting model and the logistic regression model proposed in this work employ sentence sentiment polarities to predict the document sentiment polarity. While manual annotations are used in this pilot study, automatic sentence-level classification techniques are to be integrated into the proposed models in our future work. Existing sentence-level studies include subjectivity studies (Hatzivassiloglou & Wiebe 2000), which only identifies whether a sentence is subjective; and polarity classification (Hu & Liu 2004, Kim & Hovy 2004, Nasukawa & Yi 2003, Popescu & Etzioni 2005, Khan et al. 2011), which predicts the sentiment orientation (positive, negative, etc).

Overall, our work focuses on predicting document polarities with regard to a user-given topic. And instead of making use of an opinionated lexicon (Hu & Liu 2004, Kim & Hovy 2004), our approach is based on a higher-level linguistic construct and employs sentence polarities.

There are several datasets (Wilson et al. 2005, Pang & Lee 2004, Ku et al. 2007) labelled by sentiment orientation. Wilson and Wiebe (Wilson et al. 2005) conducted an annotation experiment at the word- and phrase-level, and produced the MPQA dataset. The annotation scheme they adopted is rather fine-grained and captures the nested structures of the private states and speech events. The articles in this dataset are from the world press and are thus all news stories. Pang (Pang & Lee 2004) built a sentence-level dataset to test their classification model. All the sentences are extracted from movie reviews and are labelled as either subjective or objective. Ku (Ku et al. 2007) annotated news documents from the NTCIR collection over 32 topics, and
assigned positive, negative or neutral to each sentence. However, neither of the two data sets suffices for our study. First, they are restricted to certain domains, in which sentiment tends to be expressed in a more constrained way compared to a collection with unrestricted topic domains. Second, the sentences in these collections were not labelled by topicality. Therefore, we annotated our own data sets as part of this work.

3 Addressing Topic Drift in Blogs

To address the topic drift in blogs, we need first to examine how common this phenomenon is. This requires the ability to judge whether or not each sentence in the document is relevant to the given query. While there are existing technologies to accurately classify documents by topicality, it is much harder to classify by topicality at the sentence level. Relevant documents usually contain certain query terms about the topic, but a relevant sentence may have no topical term at all. For example,

- “Ok, here’s why this is stupid. 1) The menu ... you can also get the information in the store, by asking a McDonalds ... 2) ... “, extracted from BLOG06-20051206-033-0009894627
- “… I love french fries and when i eat them, especially Mickey Ds, I feel all warm and fuzzy ... Anyhow, french fries are my comfort food ... “

In the first example, the first sentence “Ok, here’s why this is stupid. “ is relevant, but looking at that sentence alone, it is impossible to tell what “this” really is. One would have to read the text after this sentence to link the comment to the brand McDonald. Similarly, in the second example, a reader couldn’t tell whether the “french fries” is from McDonald unless the context of the sentence wherein the phrase appeared has been examined.

We propose a simple annotation scheme to study the topic drift phenomenon. The scheme can be described as a set of rules, as explained below. We adopt the topic descriptions provided in the TREC Blog 06 data set to judge whether a sentence is topical. The descriptions for the two topics we used are shown in Table 1, and all the following examples are extracted from posts of topic 869, which is about “the cartoons depicting the Muslim prophet Muhammad printed in a Danish newspaper”.

3.1 Sentence Annotation Scheme

3.1.1 Topicality

A sentence is either topical or non-topical.

- A sentence that directly mentions the query topic is considered topical.
  
  A topical example:
  They claim he’s racist, largely because he reports stories like these: about appeasement in Europe: The Danish newspaper Jyllands-Posten recently published cartoons of Mohammed, and Danish Muslims went crazy, rioting and threatening the newspaper.

  A non-topical example:
  Little Green Footballs has interesting posts (but the comments sections isn’t usually very edifying, just so you know), but a lot of people don’t like Charles Johnson, the host of the site.

- If it can be implied from the context that the sentence is about the topic, the sentence should also be considered topical. Both sentences before and after the target sentence should be taken into consideration.
  
  “The UN is Appeasing Muslims - Again. “, followed by “UN Concerned over Prophet Cartoons by Ole Danmjaer... “

  This sentence itself does not directly mention the cartoons, but the word “appeasing” actually refers to the UN’s response to the cartoon incident, and is thus topical.

- A sentence that addresses more than one topic, including the target topic, is still labelled as topical.
  
  “Fjordman, the Norwegian blogger (how sorely his invaluable reports from Scandinavia will be missed when he quits blogging next week) that, the , has over the 12 cartoons [see them ] depicting the prophet Muhammad which were published in the Danish newspaper Jyllands-Posten last September.”

  This sentence is topical, although it also mentioned comments on Fjordman’s quit from blogging.

- Sentences from quoted content are processed in the same way as the sentences written by the author.
  
  “In a letter to the 56 member countries of the Organization of the Islamic Conference (OIC), she states: “I understand your concerns and would like to emphasize that I regret any statement or act that could express a lack of respect for the religion of others”... the 56 Islamic governments have asked Louise Arbour to raise the matter with the Danish government “to help contain this encroachment on Islam, so the situation won’t get out of control.”

  This sentence is topical because “a lack of respect for the religion of others” refers to the cartoon incident.

3.1.2 Sentiment Orientation

A topical sentence can have one of the four polarities in our system: neutral, negative, mixed, positive. The label unknown is automatically assigned to non-topical sentences, since such sentences are discarded in the classification models we proposed. The definition of the other four labels are the same as defined in the TREC Blog track.

- The polarity of the opinion is evaluated with regard to the query topic. If the sentence is about multiple topics, only the relevant opinion towards the target topic is considered when labeling. Opinions expressed in non-topical sentences are not labeled, and the sentiment polarity of all non-topical sentences is defaulted to unknown.

  This is because our task is to find and classify opinions targeted at a specific topic, therefore the irrelevant sentences are not useful.

  “Running a newspaper is a tough business these days. “ This sentence is opinionated, but not on the topic. We label it as unknown.

  “While one cartoon was particularly offensive because it showed the prophet as wearing a turban with a bomb attached to it, a great deal of the
anger had to do with the mere depiction of the prophet."

This sentence is right on topic, and explicitly shows a negative opinion towards the cartoons: “offensive,” thus labeled as negative.

- We do not identify the opinion holders. Whether or not the opinion was expressed by the author is not examined in the annotation process. In fact, many posts quote statements from newspapers and other media but not showing any opinion of the author himself/herself. Ideally, only the author’s opinions should be considered, as most users seem to be more interested in the author’s opinions. Nonetheless, our sentence-level annotation rules must be consistent with the document-level annotations, which we adopted from NIST annotators at the TREC conference. In their annotation scheme, the holder of the opinion was not taken into consideration. To be consistent, we followed their scheme.

“In a letter to the 56 member countries of the Organization of the Islamic Conference (OIC), she states: “I understand your concerns and would like to emphasize that I regret any statement or act that could express a lack of respect for the religion of others”. The 56 Islamic governments have asked Louise Arbour to raise the matter with the Danish government “to help contain this encroachment on Islam, so the situation won’t get out of control.”

The quoted content bears negative opinion toward the cartoons, though not expressed by the author. We label this sentence as negative.

- When a sentence bears opinions towards a statement which is related to the topic, we must take the opinion expressed in that statement into consideration.

“The Islamic governments have expressed satisfaction with the reply from Louise Arbour.”

Since Louise Arbour holds negative opinions towards the cartoons, the Islamic governments holds negative opinions as well. Thus the sentence is labeled as negative.

3.1.3 The Labeling Procedure

Two topics from the TREC Blog Track 2006 and 2007 tasks were used in the annotation. NIST annotators have already labelled the documents by topicality as well as sentiment polarity. We adopted the document-level annotations made by NIST assessors. For our study, we only used the posts labelled as 2 or 4, as people are generally more interested in these opinions rather than mixed and neutral. In order to reach more reliable conclusions in the classification experiments, we chose the topics which have a relatively large number of posts with a reasonable length. We used 300-words as the threshold for selecting posts, and after filtering out the short posts, we chose only the topics which have at least 50 posts in both the negative and the positive classes. When building the corpus for each topic, we sampled as many posts as possible, while keeping the number of posts the same in each class. The details of the topics are listed in Table 1, and the steps we followed to build the collection are explained below.

1. We extract the html section from the Blog06 collection, and run a preprocessing program to keep only the original post and the replies of web users. We save each post in a single document.

2. For each post, we split the text into sentences. We used a regex expression to perform this task. Note however, there is some noise in the collection due to sloppy punctuation and therefore sentences were not perfectly segmented. There are cases where multiple sentences are grouped as one, and cases where a single sentence is mistakenly split into two. Nonetheless, this does not significantly affect the accuracy of our classification models, as such cases are rare, and none of our models is based directly on the number of sentences. More explanation follows in Section 4, when the classification models are introduced.

3. By default, each sentence was labelled as irrelevant, and the sentiment polarity was labelled as Unknown.

4. The sentences and the posts were then uploaded to a database. The labels that an annotator applied to the sentence were also kept in a database.

Our annotators were able to view the full post when labelling each sentence, so that the context was taken into consideration. The sentences were shown according to the sequence they appeared in the blog post, but the annotator could always move backwards to change the labels made previously. As the task itself was rather subjective, we did not provide a detailed annotation guide. Instead, we showed the annotation rules on the side panel of the labeling interface, as indicated by Figure 1, and allowed the annotator to relate to their background knowledge when making the judgement.

3.2 Results

With our manually annotated corpus we were able to show some statistics on the phenomenon of topic drift. Table 2 shows some statistical details about the two topics. The drifting sentences constitutes as much as 86.36% and 83.40% on topic 869 and 903 accordingly. The amount of the four types of sentences has been normalized by the document length, and are shown in averaged percentages over all documents in each topic. It is also noteworthy that all documents in our collection have drifting portions with regard to the query.

The distribution of the sentences by sentiment polarity are shown in boxplots. Figure 2 shows the distribution of the sentences in the negative documents of topic 869, Figure 3 shows the distribution of the sentences in the positive documents of topic 869, and Figure 4 shows the distribution of the sentences in the negative documents of topic 903, and Figure 5 shows the distribution of the sentences in the positive documents of topic 903. It can be seen from the four graphs that sentences labeled as bearing mixed opinions are quite rare on both topics. Neutral sentences, on the other hand, have very different distributions on the two topics in the collection. This is mainly

<table>
<thead>
<tr>
<th>Topic</th>
<th>Doc</th>
<th>Neu</th>
<th>Neg</th>
<th>Mix</th>
<th>Pos</th>
<th>Drifted</th>
</tr>
</thead>
<tbody>
<tr>
<td>869</td>
<td>14323</td>
<td>0.23%</td>
<td>7.19%</td>
<td>0.06%</td>
<td>6.16%</td>
<td>86.36%</td>
</tr>
<tr>
<td>903</td>
<td>12429</td>
<td>6.56%</td>
<td>3.23%</td>
<td>0.14%</td>
<td>6.68%</td>
<td>83.40%</td>
</tr>
</tbody>
</table>

*Measured by # of characters and averaged over all documents in the topic.
due to a rather special topic description: “Find opinions worldwide to the cartoons depicting the Muslim prophet Muhammad printed in a Danish newspaper.” As the description specifically asks for opinionated content, the annotator ignored most of the neutral content. The amounts of positive and negative content have a rather strong correlation with the document polarities on both topics, and particularly so with topic 869.

In our data set every sentence has a sequence number that identifies its position in the document. With this information we are able to show the locations where subjective sentences appeared in the documents. We normalized the position by dividing the sequence number by the total number of sentences in each document, and plotted Figures 6 and 7 for topic 869 and 903 respectively. In the documents on both topics the first 20% of the text is a highly probable region where subjective sentences appear. The distribution of subjective sentences in other portions of the text varies significantly in the two topics.

4 Predicting Document Polarities

To tackle the problem of topic drift in the blogs we propose to do sentiment classification on the topical extracts. We validate the effectiveness of our approach by contrasting the performance on the full posts with that on the extracts. Our baseline approach uses the Naive Bayes classifier, which is a traditional classifier commonly used in text mining, and treats the blog post as a whole in the classification process. Reasonable performance of this model has been reported in existing literature (Pang et al. 2002). Also, when comparing our proposed approaches with this baseline system our main objective is not to improve the document-level classification models, but rather, to compare the performance of classification on full posts with that on topical snippets. For this purpose, this model is sufficient to validate our hypothesis.

Unlike probabilistic classifiers such as Naive Bayes, humans judge document polarities not by examining the word distributions, but by aggregating the polarities of sub-documental natural language structures. Motivated by this, we explore models that simulate this process. In this study we treat the sentence as the basic opinion-bearing structure, and aim to predict the document polarity by aggregating the sentence polarities. Existing work (Hu & Liu 2004, Kim & Hovy 2004, Nasukawa & Yi 2003, Popescu & Etzioni 2005, Khan et al. 2011) have shown that automatic sentence-level prediction can be done at a reasonable accuracy. Combined with our approach, it will then be possible to predict the document polarity with a reasonable performance. Such techniques are subject to further study and are not the focus of this paper.

In our work we directly use the ground truth – human-annotated sentence polarities, and propose two models: a voting model and a logistic regression model. Although our techniques could be applied to multi-class problems, in this paper we limit our scope to a binary-class problem, i.e. we are only interested in classifying documents bearing either positive or negative opinions. This decision is made primarily due to the lack of documents in other classes.

4.1 The Word-based Approach

We adopted a simple Naive Bayes model similar to the approach reported in Pang’s work (Pang et al. 2002). In this model, all of the words, aside from the stop words, from the post are included as features. Each document is represented as a vector consisting of the tf-idf values of the symbols

\[ D = \{ tf_1 \cdot idf_{w_1}, tf_2 \cdot idf_{w_2}, ..., tf_n \cdot idf_{w_n}, class \} \]  

where \( tf \cdot idf \) is defined as,

\[ tf \cdot idf = \frac{tf(t, d) \cdot \log \left| \frac{D}{|\{ d : t \in d \}|} \right|}{d} \]  

Here a symbol refers to any alphabetic token in the post, without any filtering process by means of a dictionary. As such, incorrectly spelled words, as well as commonly used abbreviations on the web (e.g. AFAIK, WTF, IMHO), are all included.
Here, \( tf(t,d) \) is the frequency of the term in the document, and \( \log \frac{|D_t|}{|D_t|} \) is the number of documents containing the term inverted by the total number of documents in the collection.

In the training process, the prior probabilities \( p(t_i|C) \) of each term appearing in the documents of each class are calculated. Then based on the Bayes Theorem, we have

\[
p(C|t_1, ..., t_i) = \frac{1}{Z} p(C) \prod_{i=1}^{n} p(t_i|C) \tag{3}
\]

where \( Z \) is a scaling factor based on the vector \( \{t_1, ..., t_i\} \), and \( p(C) \) is the prior probability of the classes. In our experiment we adopted the implementation in the Weka data mining toolkit\(^2\).

### 4.2 The Voting Model

Intuitively, a document with a larger portion of positive content is more likely to be positive than negative. Motivated by this, we propose a voting model that classifies the documents by comparing the amount of positive and negative contents in a document. The rule is simple, the majority class in the topical sentences is assigned to the document. The polarity \( P(D) \) of a document \( D \) is given by,

\[
P(D) = \begin{cases} 
\text{positive} & \text{if } |D_{\text{positive}}| \geq |D_{\text{negative}}| \\
\text{negative} & \text{if } |D_{\text{positive}}| < |D_{\text{negative}}|
\end{cases} \tag{4}
\]

In measuring the amount of the contents \( |D_{\text{positive}}| \) and \( |D_{\text{negative}}| \), either the number of tokens or the number of characters can be used. Experiment results are shown in Section 6.

Intuitively this model is not perfect, due to the subtlety of human language. It has been reported in previous work by Pang, et.al (Pang et al. 2002) that a specific problem by the name of thwarted exceptions exists, where the author sets up a deliberate contrast to his/her earlier discussion. In this case, our voting model would fail, as despite of the fact that the document may have more positive content, it should be classified as negative. Measures to tackle this problem is subject to further study. Surprisingly however, in our experiments we observed a significantly better performance than other approaches on one of the topics we used, and a slightly inferior performance on

\(^2\)http://www.cs.waikato.ac.nz/ml/weka/
the other topic.

4.3 The Logistic Regression Model

The logistic regression model describes the relationship between one or more independent variables and a binary response variable, expressed as a probability, that has only two values, which matches our problem. This model predicts the value of the response variable with the logistic function,

\[ f(z) = \frac{e^z}{1 + e^z} \]  

where \( f(z) \) represents the probability of an outcome, and \( z \) is the input, whose value is determined by the features. We use the accumulated lengths of sentences with each different sentiment polarity as features, and assume a linear relationship between the features,

\[ z(f_1, ..., f_n) = \sum_{i=0}^{n} x_i f_i \]

where \( x_0 \) is the intercept of the linear function, and \( f_0 \) arbitrarily set to 1. In our experiments, only the accumulated lengths of different types of sentences were used as features, and the predicted class label is given in the form of a probability value calculated with \( f(z) \).

When we use all four types of the sentences, the linear function is given in the following form,

\[ z(f_1) = x_0 + x_{ncu} f_{ncu} + x_{neg} f_{neg} + x_{mix} f_{mix} + x_{pos} f_{pos} \]

where \( x_0 \) is the intercept of the function, \( x_{ncu} \) is the accumulated count of the tokens in neutral sentences, \( x_{neg} \) is the accumulated count of the tokens in negative sentences, \( x_{mix} \) is the accumulated count of the tokens in mixed sentences, \( x_{pos} \) is the accumulated count of the tokens in positive sentences.

5 The Classification Experiment

We conducted our experiments on the annotated collection introduced in Section 3. All the documents we used have been annotated by NIST assessors as topical, and bearing either positive or negative opinion. We use their sentiment polarity labels as the golden standard for document polarities, and our sentiment-level annotations as the gold standard for sentence polarities. In this experiment, we only focus on a binary class problem, i.e. only documents labeled as either positive or negative were used, and those labeled as neutral or mixed were discarded. When constructing the corpus we have also intentionally kept an even class distribution.

The objective of this set of experiments is to validate our hypothesis that we can achieve better classification performance on the extracted topical extracts than on the original full posts. The steps below were followed to carry out the experiments,

1. We first generated the topical snippet collection by extracting the topical sentences, whose topicality labels are Relevant, from the full posts.

2. Stop words and non-alphabetic symbols were then removed from both collections. Words that appear less than 3 times in the documents of each class (either positive or negative) were also removed for robustness, to be consistent with usual data mining practice.

3. When classifying with the Naive Bayes classifier, each blog post document were transformed into a vector consisting of features which are computed from statistical information of each term. In our experiments we used the tf-idf values as features.

4. Five-fold cross validation was applied to evaluate the performance of the all the approaches. Note that the voting model we proposed is unsupervised. However, by applying the cross validation process we are thus able to compare the robustness of this approach. We did not use ten-fold cross validation mainly because we had only a limited number of documents in the collection.

The experiment results are shown in Figure 8 and Figure 9. With the word-based approaches full_NB and summ_NB, significant improvement in the classification accuracy has been observed on the extract data set of topic 869. On topic 903 however, the average accuracy dropped dramatically. This is probably caused by the much smaller feature set (254 words) for the extract data set compared to the full post data set, which has 4067 features. Note though, this
drop in accuracy is not statistically significant, as is shown in Figure 4. Considering the size of the topical snippets is only 13.64% of the full post on 869 and 14.37% on 903, this system may still be favored in a commercial setting. However, we cannot arrive at the conclusion that the system based on topical snippets has higher efficiency, as the computation cost of snippet generation is subject to further study, and a larger dataset with more topics is needed to evaluate how the system works in different domains.

For the voting model sum, VM, we used the accumulated count of the tokens to evaluate the amount of the positive and negative content. As mentioned in Section 3.1, some noise were introduced when splitting the document into sentences, where either a group of sentences were treated as one, or one sentence was split into several. With our evaluation measure, only the former case (a group of sentences were treated as one) will affect the classification process, in which case the amount of the content with the labeled polarity will be inflated. Such cases are quite rare in the collection. From Figure 8 and Figure 9 we can see that the voting model outperformed the word-based model on both topics. However, as is shown in Table 5, its improvement over the word-based approach is not statistically significant on topic 869.

For the logistic regression model introduced in section 4.3, we used the accumulated count of tokens in each type of the sentences as features. The results shown in Figure 8 and Figure 9 were based on only two types of sentences, namely, positive and negative. The performance of this approach is consistently better than the performance of the approach that uses the full post. Although preliminary, this result suggests that the topical extracts are not only highly abridged in size, but may also provide a more accurate presentation of the opinions shown in the posts towards the specific topic.

We have also experimented with all four types of the sentences (neutral, negative, mixed and positive) with the logistic regression model. Note that this experiment was only applicable on topic 903, as documents with sentences labeled as mixed are too rare on topic 869 to carry out five-fold cross-validation. Interestingly, we observed an accuracy of 75.72% with four features but 76.88% with two features (using only positive and negative sentences). This suggests that sentences labeled as mixed or neutral may not be useful in identifying the document polarity. This is also reflected in the coefficients $x_i$ learnt in the five fold cross validation process, as is shown in Table 3. Strong correlation between positive sentences and positive documents is observed, and negative sentences correlates to negative documents; whereas the correlation between the neutral sentences and the negative documents is very weak, and no reliable correlation between mixed sentences and the document polarity is observed.

### 6 Conclusions and Future Work

In this paper we have studied the problem of topic drift in blogs. By manual inspection of the blog posts we observed a high level of topic drift, and hypothe-
sized that classification on only the relevant portion would be more efficient both in terms of accuracy and efficiency. To validate this hypothesis, we have annotated a small collection of posts from two topics at the sentence level, thus able to produce manual topical extracts as ground truth. We should note that the snippets for both topics are much smaller in size compared to the full posts (13.64% for 869, and 14.37% for 903).

We have proposed two cascaded models that build upon the sentiment polarities of the topical sentences to predict the document polarity. On both topics, the logistic regression model has achieved higher accuracies than what the word-based approach achieved with the original blog posts. The voting model had a similar performance but its improvement over the word-based approach is not statistically significant. Our preliminary experiments have confirmed our hypothesis that classification on the topical extracts may result in better accuracy, but also suggests that approaches based on word-distribution may be less robust than those based on sentence-level polarities when using the extracts.

Our main contributions in this work are first the sentence-level annotations, which could be used for further analysis in the research community, and second the new approaches to classify the document polarity with sentiment polarities. Our future work will focus on automating the sentence classification process and expanding the data set, so that the effectiveness of our models can be tested in a more robust context.

References


Feature-based recommendation framework on OLAP

Yang Yang, Jinli Cao
Department of Computer Science and Computer Engineering
La Trobe University, VIC, 3086
Australia
y14yang@students.latrobe.edu.au
J.Cao@latrobe.edu.au

Abstract
The queries in Online Analytical Processing (OLAP) are user-guided. OLAP is based on a multidimensional data model for complex analytical and ad-hoc queries with a rapid execution time. Those queries are either routed or on-demand revolved around the OLAP task. Most such queries are reusable and optimized in the system. Therefore, the queries recorded in the query logs for completing various OLAP tasks may be reusable. The query logs usually contain a sequence of SQL queries that show the action flows of users for their preference, their interests, and their behaviours during the action.

This research investigates the feature extraction to identify query patterns and user behaviours from historical query logs. The expected results will be used to recommend forthcoming queries to help decision makers with data analysis. The purpose of this work is to improve the efficiency and effectiveness of OLAP in terms of computation cost and response time. Furthermore, the proposed OLAP system will be able to adjust some parameters for finding common behaviours from different users that make the recommendation system flexible and user-adaptive.

Keywords: OLAP, session feature, query recommendation.

1 Introduction
The OLAP analysis system, a major part of business intelligence, is an effective approach to processing complex queries in multi-dimensions so that the user acquires a multi-dimensional view from a data warehouse (Dinter, Sapia, Hoffling and Blaschka 1998). The typical applications of OLAP are in business reporting for sales, marketing, budgeting and forecasting, and similar areas. OLAP is interactive and is involved in a series of queries for either driven-down or roll-up queries to support specific tasks of decision making. During the analytical processing, a large amount of multidimensional data is required to be accessed and based on the online queries. Since OLAP is driven by the user query that the user needs a clear purpose and knowledge about areas of interest before operating OLAP processing (Sapia 1999). This approach is classified as query-driven centric as opposed to user centric techniques. However, when exploring a series of tasks, the user may not have clear ideas of what a next query should be made. This might cause analysis latency or lead users into an irrelative area, and thus reduce the benefits of using the OLAP system. To overcome this problem, we present a framework for query recommendation embedded in the OLAP system, which helps users predict forthcoming queries.

In business processing, users always utilize a sequence of queries to interact with the OLAP system to finish a task. The operation is often monotonous and interrupted since users could lose their orientation to the goal (Sarawagi 2000). Most frequently-used queries are recorded in the system as query logs. Those recorded queries may be reusable for completing similar OLAP tasks. The query logs usually contain a sequence of SQL queries that show the action flows of users for their preference, their interests, and their behaviours during the action. We will propose a new framework to provide navigation and forecast forthcoming queries which maximizes efficiency of using OLAP systems. A sequence of queries/requirements is called an analysis session (Giacometti Marcel and Negre 2008). The new framework is based on a collaborative filtering method (Herlocker, Konstan, Terveen and Riedl 2004). This method assumes if user A has the same interested area to user B, then they may be interested in the same data. The queries of user B may be reusable for user A. The proposed framework utilizes all users’ query logs and summarises these query records to capture different users’ query behaviour. The featured information is used to forecast what the forthcoming query could be.

It is important for query recommendations to identify the current user’s purpose in order to make an accurate recommendation. However, current users cannot supply much information at the beginning so that previous approaches could not match a session to current users. This problem can be considered as unbalanced data. Previous sessions may include too many features that could not provide the central themes of the sessions, while the current session has little information. To overcome this drawback, our framework introduces a new approach that balances the features between the previous session and the current session. Our approach can satisfy the user’s requirement and provides appropriate query recommendations. The proposed framework is flexible and user-adaptive for recommender systems.

This paper is organized as follows. Section 2 reviews some existing work about query recommendations. Section 3 introduces our framework and its instantiation. The experimental results are shown in section 4. Conclusions are presented in section 5.
2 Related work

Query recommendation is an important technique for business search engines. The majority of work on query recommendation focuses on measuring the similarity between the current query and the previous query (Fonseca, Golgher, Moura and Ziviani 2003) in order to expand the query or cluster of queries (Baeza-Yates, Hurtado and Mendoza 2004, Wen, Nie and Zhang 2001). The basic model of recommendation systems relies on two methods which are content-based methods and collaborative filtering methods (Adomavicius and Tuzhilin 2005).

2.1 Content-based methods

Content-based methods suggest items similar to the ones that users have previously shown interest in. They mainly extract information features, such as the different sales of cars each year, and decide which recommendations are appropriate. These methods compare various candidate items, and then the best matching items are presented to the user. Generally, they use a distance function to rate or order each candidate item. The cosine similarity measure (Adomavicius and Tuzhilin 2005) is one of the best vector measures that represents weight best-matching.

In the early data source, the significant information existed randomly. Content-based methods apply information retrieval on data sources (Baeza-Yates and Ribeiro-Neto 1999) and information filtering on recommendations (Belkin and Croft 1992). The methods extract useful profiles that contain information about users’ preferences and search behaviour. The profiles can be elicited from users’ queries.

The limitation of this method is obvious. The users need to explicitly describe their objects, so the system must first learn the user’s purposes. However, the user’s purpose is much harder to indicate when they get a new job task. Moreover, content-based methods cannot suggest forthcoming queries.

2.2 Collaborative filtering methods

Collaborative filtering methods recommend the items which have similar interest between the user and the other users. They can be categorized into memory-based and model-based methods by algorithm (Breese, Heckerman and Kadie 1998).

The memory-based algorithm exploits the weight of all previous items to compute the weight of the current session. The formula is below:

\[
 r_{c,s} = \frac{k}{\sum_{c' \in C} |\text{sim}(c, c')|} \times r_{c',s}
\]

\[
k = \frac{1}{\sum_{c' \in C} |\text{sim}(c, c')|}
\]

where \( r_{c,s} \) denotes a weight of the user c and item s. \( r_{c,s} \) multiplier k serves as a normalizing factor, and \( \text{sim}(c, c') \) is a distance measure between user c and user c’, which can be considered as a weight. \( \text{Sim}(c, c') \) is introduced in order to be able to differentiate between levels of user similarity. This formula means that the high weight depends on the comparability of both c and c’. Different recommendation applications can use their own user similarity measure, as long as the calculations are normalized using the normalizing factor k.

Model-based algorithms (Billsus and Pazzani 1998) use known weights to build a model, and then recommend a query by this model. Cluster models and Bayesian networks are techniques in this algorithm that rate probability expression on the purpose of interest. The model learns all features from the data. For example, in a car recommendation system, a user might prefer a kind of car with job purposes like ‘energy saving car’ and completely different type for sport like ‘four wheel drive’.

To improve the performance of the collaborative filtering method, several techniques that exclude noise, redundancy and exploit the sparsity of data should be integrated (Yu, Xu, Tao, Ester and Kriegel 2002). As in the case of the content-based method, the main improvement of the collaborative filtering method is that it uses other users’ information to make a recommendation even though the information never appeared before by current user.

Collaborative filtering methods also have their own limitations. They have the same problem as content-based methods. In order to make accurate recommendations, the user must offer abundant information to describe the job purposes. Another significant drawback is that other users’ information could include some irrelevant information that would affect the accuracy of recommendations.

2.3 Other methods

The graph model based similarity method describes the two consecutive queries (neighbouring queries) in the same query session. These have more similar weight than the queries which are not neighbours in the same session (Zhang and Nasraoui 2006). Instead, Fonseca et al. detects similar queries based on association rules. Each query in the query log is considered as a part of session where a single user inputs a sequence of related queries in a time interval (Fonseca, Golgher, Moura and Ziviani 2003). We will use a similar notion in this paper.

Many recommendation systems are applied in web searching, and some of those techniques can be adapted to the OLAP recommending system. Wen et al. (Wen, Nie and Zhang 2001) present four notions about clustering methods for query recommendation to measure query distance: the first notion is based on keywords or phrases; the second is on a string matching of keywords; the third is on common clicked URLs; and the fourth on the distance of the clicked documents in some pre-defined hierarchies. The first two notions will be adopted in our work.

The QueRIE (Chatzopoulou, Eirinaki and Polyzotis 2009) framework developed by Chatzopoulou et al. generates a predicted summary (\( S_0^{\text{pred}} \)) to capture the predicted degree between current user and the other users who have a similar query requirement. Where \( S_i \) represents the session summary of user i, when \( i = 0 \) denotes the current user. \( S_0^{\text{pred}} \) is described as a weighted vector, which represents the number of given tuples \( r \) appeared in database. This framework builds a predicted model which also returns recommendations to the current user. There are two different schemes for calculating \( S_i \) (Chatzopoulou, Eirinaki and Polyzotis 2009).
\[ S_0[\tau] = \begin{cases} 1 & \text{if tuple } \tau \text{ appears} \\ 0 & \text{if tuple } \tau \text{ does not appear} \end{cases} \]

This is a binary weighting scheme referring to matching tuple \( \tau \) or not.

\[ S_0[\tau] = \begin{cases} \frac{1}{\text{ans(Q)}} & \text{if tuple } \tau \text{ appears} \\ 0 & \text{if tuple } \tau \text{ does not appear} \end{cases} \]

This is the result of the weighting scheme, where \( \text{ans(Q)} \) is the result-set of query Q. It implies that \( S_0 \) could have a small outcome if many queries return results since this query is unfocused. In other words, if \( S_0 \) is large, then the query is very specific. The session summary is defined below:

\[ S_i = \sum_{Q \in \Omega} S_0 \]

This approach adopted the method of collaborative filtering. The predicted summary is defined as a function of the current user’s summary \( S_0 \) and the normalized weighted sum of the existing summaries (Chatzopoulou, Eirinaki and Polyzotis 2009):

\[ S_0^{\text{pred}} = \alpha \times S_0 + (1 - \alpha) \times \frac{\sum_{i} \text{sim}(S_0, S_i) \times S_i}{\sum_{i} \text{sim}(S_0, S_i)} \]

The value of the mixing factor \( \alpha \in [0, 1] \) specifies which users’ requirement will be considered. For example, if \( \alpha = 0 \), the result trends to past users’ requirements. \( S_0^{\text{pred}} \) also tends into content-base filtering when \( \alpha = 1 \). The value of \( \alpha \) can decide which approach is more favorable. This approach could exclude other users’ query requirement if the current session clearly declares the query’s orientation. Moreover, it is desirable that current user can get some useful knowledge from past users to guide or adjust successor queries. Our approach is able to predict queries that combine the results already observed by the past users.

3 Query recommendation based on user behavior

By reviewing the existing methods above, we observed that query recommendation needs further improvement that a more feasible method is necessary to apply for OLAP systems. Therefore, we propose a new recommendation system to better the performance on query recommendation.

3.1 Improve existing query recommendation system

The existing query recommendation systems extend from collaborative filtering and content-based methods. They depend on distance measure to generate related sessions or queries. However, each previous session may include many queries, and thus it may involve much independent information. This fact shows that some sessions may be unfocused if the session contains many queries. The current session only contains few queries in the beginning so that the current session cannot clearly describe the main purpose of the user’s requirements yet. Furthermore, if several previous sessions cover current session, the different focuses may offer different suggestions to current session.

This, therefore, motivates us to look for a novel method which can overcome the drawbacks of existing methods. Our proposed framework will satisfy the following requirements:

- It should filter unuseful information in the previous sessions to identify session patterns from historical query logs.
- It should summarize the main purpose of previous sessions. The expected results will be used to recommend forthcoming queries to help decision makers on data analysis.
- It should improve the efficiency and effectiveness of OLAP in terms of computation cost and response time.

3.2 Framework of feature-based recommendation

The feature-based recommendation framework can be formulated as follows: when interacting with a data warehouse, a user might have the same or similar requested task as a past user in the previous session. To extract the previous session, the system can use the log information to summarize the querying behavior from past users.

Our recommended framework (figure 1) works on the OLAP system using SQL queries. The user’s queries are sent to OLAP and the recommendation engine. The OLAP system processes every query with the data warehouse and returns the results to the user. Every user’s query is also recorded in the query log with the user ID to store its queries. The recommendation engine relies on current inputting queries and past user’s queries which are stored in query log to generate a set of query recommendations using our novel approach, and then the recommendation engine returns the recommendations to the user.

3.3 Definition of notions and symbols

We now introduce some notions and symbols which will be used in the remainder of this paper.

3.3.1 Models of sessions and queries

The basic operations of OLAP include three methods that referred as slice and dice, pivot and drill down, and roll up to analyze a series of users’ requirements by SQL queries.
These requirements (queries) of the same user can be considered as one task that may contain one purpose during the search job. Semantically, a series of requirements or queries will be reusable for other users who have similar tasks on their search job. These SQL queries are recorded into query logs in order to reuse them. The query logs are denoted as $L$ and one purpose of a task is called a session as denoted as $S_i$. Each session includes a series of queries that are entered by the same user, and each query denoted as $q^j_i$ for the $j^{th}$ query in the $i^{th}$ session. This relationship is represented as follows:

$$q^j_i \in S_j \text{, where } 1 \leq i, j \leq n \text{ and } S_j \in L.$$ 

### 3.3.2 Session Feature

In each search task, the user usually inputs a sequence of queries to accomplish one task. They may ask the sale results of various products in different locations or income of branch offices in different countries. It means that every session has some kinds of characteristics shown as the keywords in queries. Many keywords in one session may produce the characteristics of the session. Our work focuses on extracting the particular characters as the guide to make recommendation. Each particular character is called a session feature denoted as $F_i$. Session features can express the main purpose of the session. Different sessions could have similar search tasks (session feature) that means existing a probability of similarity between session $j$ and session $i$ which is denoted as $R^j_i$. The research of this paper is to mine the query logs and extract the session features from query logs that will be used to recommend for current users. The symbols will be used in this paper shown in table 1.

<table>
<thead>
<tr>
<th>$L$</th>
<th>Query logs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_i$</td>
<td>A sequence of queries entered by user $i$</td>
</tr>
<tr>
<td>$t$</td>
<td>A tuple of keywords</td>
</tr>
<tr>
<td>$q^j_i$</td>
<td>$j^{th}$ query of the $i^{th}$ session</td>
</tr>
<tr>
<td>$R^j_i$</td>
<td>similarity between session $j$ and session $i$</td>
</tr>
<tr>
<td>$F_i$</td>
<td>Session feature</td>
</tr>
</tbody>
</table>

**Table 1: Symbol summary**

### 3.4 Feature-based recommendation approach

Query logs are important data sources for query recommendations. There are two steps in our approach. First, summarize session features by utilizing past users’ recommendations. There are two steps in our approach. The Query log can be divided into sessions by many ways such as user id, timestamp and performance in a period. We combine user id and timeout threshold as a splitting sessions rule. The timeout threshold is set less than 12 hours. Thus, the query log is divided firstly into several parts by user ids, and then compares the timestamps of two consecutive queries. If timeout of these two queries exceeds 12 hours, the smaller timestamp of the query is a splitting point for a new session. Since each session could include many queries, the theme of a session is difficult to detect. Hence, we need extract keywords from each session to represent the features of this session.

Intuitively, for a session $S_i$ and current session $S_c$, the more common keywords exist between $S_i$ and $S_c$, the more $S_i$ is similar to $S_c$. Therefore, the problem of measuring similarity between $S_i$ and $S_c$ becomes the calculation of common features of $S_i$ and $S_c$.

Let us look at an example: suppose that $F_i$ and $F_j$ are two feature sets of previous sessions $S_i$ and $S_j$ respectively, and $S_i$ and $S_j$ include the same number of keywords with current session $S_c$.

Figure 2 shows the graphical example of feature sets $F_c, F_i$ and $F_j$. The shadow square box represents current session $F_c$. $F_j$ and $F_i$ contain $F_c$, that is, $F_c = |F_i| \cap |F_j|$. Since $|F_i| > |F_j|$, the proportion of similarity between $F_c$ and $F_j$ ($F_c / F_j$) is smaller than $F_j / F_i$. It is easily observed that current session $S_c$ is more similar to $S_j$ than $S_i$.

![Figure 2: example of session feature](image)

**Measure session feature.** Due to find suitable session to match current session, we need measure session features. Every session is composed of many SQL queries. Each query has a `select` statement as beginning, which includes several attributes from different tables in database. We consider these attributes as the keywords of the query. If a session $S$ consists of a sequence of queries, the features of session $S$ is the most frequently appeared keywords in those queries.

A tuple is one of the combinations from keywords in a query. We assume that the probability $P_i(t)$ represents the importance of a tuple with keywords, so we have:

$$P_i(t) = \alpha \times \frac{n_i}{N_i} + (1 - \alpha) \times \frac{n_s}{N_s}$$

Where $i$ denotes the number of keywords in tuple $t$, $n_i$ and $n_s$ stand for the number of tuple $t$ appearing in the query logs $L$ and session $S$ respectively; $N_i$ and $N_s$ denote all tuples in $L$ and $S$. The value of the moderator $\alpha \in [0,1]$ determines which approach is more favourable when computing the session features. If $\alpha = 0$, only user session information will be taken into account when measuring session features. That is, the content-based filtering method is used. When $\alpha = 1$, the collaborative approach is applied to measure probability of tuple $t$. Any value of $\alpha$ in an acceptable range allows us to adjust more importance to either side. This adjustment can be useful for two reasons. First, $n_i / N_i$ is able to find the important tuple that most users are interested in, but it cannot represent if a tuple has high frequency in the session instead of the query log. We do not want to exclude from unimportance tuples that seem to explain the purpose of session. Second, a particular search task in each session usually requests some typical tuples to expand on the following search task.
In other words, the tuple with high hits may have ability to predict what kinds of purposes with search task in this session. High hits have different means when they appeared in the query log and sessions, which represent a common view and an individual view respectively. The action of $\alpha$ is used to balance these two views.

The number of tuples generated from a query is the all combinations of keywords in the query. For example, a query $q$ contains three keywords $q = \{k_1, k_2, k_3\}$, all combinations of $k_1, k_2, k_3$ make the set of tuples of $q$ as follows:

$$tuples = \{(k_1, k_2), (k_1, k_3), (k_2, k_3), (k_1, k_2, k_3)\}$$

Where each element in the above combination set is called a tuple. $P(t)$ is a threshold used for determining the important tuple in a session. If $P(t)$ exceeds the threshold, the tuple is important in the session. It can be represented as one of session features, such as $F_t = ((k_2), (k_1, k_3))$. From the result of our experiments, our approach can extract session features effectively which match the purpose of each session.

**Weight of tuple.** As a set of session features, each feature can be modelled as a weight in the session.

Definition 1: Let the tuple frequency be the number of occurrences of tuple in the session with the notation $freq(t)$ . The weight of tuple frequency $TF(F, t)$ measures the relationship of a tuple $t$ with regard to the given session feature $F$.

There are many ways to measure the weight of each tuple. For example, we can simply define that as the tuple frequency against the total number of occurrences of all the tuples in the session. In this paper, we utilize a formula from the Cornell SMART system to normalize the tuple frequency. The formula is described as follows:

$$TF(F, t) = 1 + \log \left(1 + \log \left(freq(t)\right)\right)$$

**Computing $R^f_i$.** The next step in framework is computing the similarity and recommendation value. Similarity of two sessions is computed by session feature. In this work, we adopt the method of cosine similarity that measures the similarity between two sessions and takes values in $[0, 1]$.

Definition 2: Given two clustering session features $F_i = \{t_i^1, t_i^2, ..., t_i^n\}$ and $F_j = \{t_j^1, t_j^2, ..., t_j^n\}$, the overlapping tuples between $F_i$ and $F_j$ can be represented as $F_i \cap F_j = \{t_{e}^1, t_{e}^2, ..., t_{e}^n\}$, where $t_e^j$ denotes the common tuple in $F_i$ and $F_j$. If the session is the current session, the session feature $F_c$ includes all tuples by combinations from keywords. Here, we have:

$$R^f_i = \frac{\sum_k TF(F_i, t_k^e) \cdot TF(F_j, t_k^e)}{\sqrt{\sum_i TF(F_i, t_i^e)^2 \cdot \sum_m TF(F_j, t_m^e)^2}}$$

If there is a highest score of $R^f_i$ for current session $S_c$, it implies that $S_i$ has the similar search task with $S_c$. Therefore, $S_i$ can be recommended to $S_c$ for forthcoming queries.

The users usually input a series of queries when they finish one job task. The purpose of their task will be clearer when more queries are performed. During the processing current user’s queries, the system keeps the update of suitable recommendations. However, recalculating the recommendations takes much time because the query log may have large number of sessions. Therefore, our system generates top-$k$ recommendations to provide multiple choices for the current user. When the user inputs successional queries, the system generates new recommendation from multiple choices of top-$k$ recommendations rather than recalculating from query log. This top-$k$ recommendation set is computed by the $R^f_i$, and is defined as follows:

$$\text{rank}(R^f_i) \geq \text{threshold}$$

The $R^f_i$ has high rank if it is larger than the threshold. The top ranked sessions are returned as the recommendation set.

4 Experimental evaluation

In this section, we developed a virtual query interface system which realizes our framework and analyse actual data, as well as present experimental results of using our system.

4.1 Experimental data and methodology

The experiment was carried on a Core 2 Duo 2.4GHz computer with Window 7 Ultimate and 2 GB of main memory. All system was implemented in Java.

The system adopted the data of query log from sky server database. The query log recorded queries from year 2007 to 2010 and its size is 1.3 TB. We separated the query log into 4000 sessions by different IP addresses. We chose 55 sessions to construct current sessions and the number of queries in each session are no more than 6. For the rest of sessions, firstly, they were divided into 10 equally sized subsets as previous session logs. Secondly, we partitioned the rest of sessions into 10 different sized subsets, and the number of session in next subset is larger than the preceding subset. In order to analyse the performance of our framework, we use a smaller set of queries in each current session. For example, for each current session with $n$ queries, we extracted $n-1$ queries in each current session to build test set.

We evaluate the accuracy to analyse the effectiveness of each recommended sessions, using the following metrics:

$$\text{accuracy} = \frac{|k_e \cap k_i|}{|k_e|}$$

Where $k_e$ and $k_i$ represent all queries of keywords in a session. In our test, we have analysed all subsets and reported the best result in the experiments. Unless otherwise noted, we set $\alpha = 0.6$.

4.2 Result

Our first experiment evaluates the efficiency of the proposed approach to make the recommendations. Figure
Figure 3: efficiency analysis

Figure 3 shows the performance of test sessions according to the size of query log. The measure of this experiment includes computation of similarity with current session and ranking the candidates. The features of sessions are processed on off-duty time so that the extraction time does not count in efficiency analysis.

Figure 4: accuracy distribution

As can be seen from Figure 3, it is obvious that the trend of execution time is upwards with the log size. The execution time is acceptable with the size of 4000 sessions in query log.

In the next experiment, we evaluate the effectiveness of our approach. Figure 4 indicates the comparison of best performance between our approach and the memory-based approach (Breese, Heckerman and Kadie 1998). 55 current sessions have been tested among 10 equally sized subsets, which ranked by accuracy in decreasing order. It can be seen from Figure 4, 70% current sessions achieved high performance by our approach (around 40 current sessions have above 0.6 accuracy among total 55 current sessions tested). The system can provide valuable recommendations for most current sessions. There are two zero accuracy results among all tests, that is, the system may have 3.6% failure rate to predicate intention under some circumstances. We have also manually reviewed the generated data for the validation. No recommendation sessions have been obtained if low similarity $R_i^j$ has been calculated between the current session and past sessions. For memory-based approach, only 55% current sessions can get valuable recommendations, and most of accuracy is lower than our approach (refer to the dot line in Figure 4).

Figure 5 describes an average accuracy of our approach and the memory-based approach from total 55 current sessions tested. We can find that the accuracy of our approach is always higher than the memory-based approach. The figure shows that the accuracy slightly increases with the log size. The Figures 4 and 5 demonstrate that most current sessions can obtain a successful recommendation by our approach with accuracy above 0.6 from the query log.

Figure 5: average accuracy

5 Conclusion

To sum up, our feature-based recommendation framework highlights on personal intention. It can summarize the important information and filter out useless data to build session features from previous sessions. The experimental evaluation proves our framework is efficient and effective.
Our proposed approach is promising and has great potential to apply in real-life.

6 References


Evaluating Semantic Browsers for Consuming Linked Data

Fahad Alahmari\textsuperscript{1}  James A. Thom\textsuperscript{1}  Liam Magee\textsuperscript{2}  Wilson Wong\textsuperscript{1}

\textsuperscript{1} School of Computer Science and Information Technology
\textsuperscript{2} School of Global Studies, Social Science and Planning
RMIT University
Melbourne, Australia
Emails: firstname.lastname@rmit.edu.au

Abstract

The value of a single dataset is increased when it is linked to combinations of datasets to provide users with more information. Linked Data is a style of publishing data on the Web by using a structured machine-readable format, RDF, and semantically typed relations to connect related data. Its structured representation opens up new possibilities in the way these data can be accessed and queried, while posing new design challenges for human interactions such as overloading data, navigation style, or browsing mechanism. In this paper, we review 14 semantic browsers available for the consumption of structured Linked Data and evaluate them against our five criteria framework in order to establish how well these browsers bring the benefit of Linked Data to human users.

Keywords: Linked Data, Semantic Web, Semantic browsers, RDF browsers, Semantic interface, User interaction

1 Introduction

The Semantic Web (SW) has been famously defined as a “web of data that can be processed directly and indirectly by machines” \cite{13}. Just as the value of World Wide Web documents increases dramatically when they are linked to other documents, the SW enhances the value of data by providing standardised mechanisms for describing and linking them to other datasets. The emergence of Linked Data (LD) sets encoded in the Resource Description Framework (RDF) results in a powerful network of machine-processable information, making way for the next generation of the Web \cite{3}. Distributed on a global scale, LD in turn can be used by machines to generate new information and knowledge.

Resources on the Web of LD are identified using Uniform Resource Identifiers (URIs) and encoded in the Resource Description Framework (RDF) results in a powerful network of machine-processable information, making way for the next generation of the Web \cite{3}. Distributed on a global scale, LD in turn can be used by machines to generate new information and knowledge.

Resources on the Web of LD are identified using Uniform Resource Identifiers (URIs), which are used to specify a globally unique name for a resource. Resources can be further distinguished between resources that describe real-world objects such as people, places and cars, and resources that describe other resources, including documents, images and video. In the past decade, many large providers of data, including the US (data.gov\textsuperscript{1}) and UK (data.gov.uk\textsuperscript{2}) governments have adopted LD for publishing their data. As evidence of the growth of LD \cite{5}, the Linked Open Data (LOD) community project (Figure 1) has grown to 256 datasets, which include nearly 30 billion triples interlinked by 471 million links (as of August 2011\textsuperscript{3}). The LOD cloud contains data from a range of different domains, including media, government, the life sciences and geography. Significantly, these datasets are linked together by terms from the W3C base vocabularies — RDF, RDF Schema and OWL \cite{18} in order to make LD machine processable. As a further sign that LD is moving towards “mainstream” adoption, Google, Yahoo and Microsoft have recently agreed to share a collection of schemas http://schema.org/ that can be used to publish structured and LD.

While LD is designed for machine processing, human agents ultimately need to be able to navigate and query datasets. With the rapid growth of LD, users require powerful tools to browse and explore particular datasets of interest. However, there are considerable challenges for designing usable browsing and visualisation tools to explore the Web of LD as the requirements for browsing data are very different from the requirements for browsing documents. These challenges can be summarised as follows:

1. Exploratory challenges: the Web of LD connects a huge range of related data that includes both real-world and other web resources. How can a browser best present this wide range of data to the end users in a meaningful way?

2. Navigational challenges: the navigation techniques for LD differ from the navigation techniques in the Web of documents. How can a browser provide convenient mechanisms to allow for conventional forward and backward navigational techniques to users, while also making use of the linking context?

3. Interactive challenges: human users frequently need to interact and manipulate datasets in a number of ways. In the LD world, this is made more difficult due to the different languages used. While the formal languages of the SW — OWL, RDFS, RDF, RDFa, GRDDL and SPARQL —

\textsuperscript{1}http://data.gov
\textsuperscript{2}http://data.gov.uk
\textsuperscript{3}http://www4.wiwiss.fu-berlin.de/ldcloud/state/
are useful for defining resources for machine processing, they are often challenging to work with and understand, especially for human agents who are more familiar with document and scripting languages like HTML5, XHTML, CSS3 and JavaScript. How can a browser permit users to interact and manipulate LD?

The challenges above show that there are substantial gaps in bringing the benefit of LD to human users. Considerable research has been devoted to the design and development of “semantic browsers” that aim to overcome these gaps by providing mechanisms for navigating and visualising LD sets. While Dadzie and Rowe [8] provide a comprehensive review of current state-of-the-art semantic browsing tools, their emphasis is on the first two of the three challenges identified above. Here we present a further review, closely following their methodology, but with greater attention to the third of these challenges. Specifically, we examine the facilities and features that SW browsers provide for incorporating new datasets as well as interacting with existing LD sets.

The main contributions of this study are as follows:

1. We propose five criteria for evaluating semantic browsers.
2. We briefly review 14 semantic browsers that are currently available for the consumption and manipulation of LD.
3. We evaluate the semantic browsers according to our five criteria in order to determine the ones that are best suited for LD manipulation. Again following Dadzie and Rowe [8], we include in our evaluation a discussion of the pros and cons of the browsers for interacting with LD.

Section 2 of the paper gives overview on recent related work, including a brief review of semantic browsers. Section 3 proposes our five criteria and methodology for the browser evaluation study presented in Section 4. Section 5 provides a discussion and the conclusion.

2 Related Work

In recent years, there has been an increasing amount of literature on LD. While building upon the foundations of the SW architecture, LD focuses more specifically on the linking and usage of datasets. Recent studies point to considerable research attention on publishing LD [4], using LD [9], searching LD [19], sharing LD across a community [6] and browsing LD [8].

In 2006, Berners-Lee articulated several key rules for publishing data on the Web in order to expose it as LD [3]. These rules are:

1. Use of URIs as names for things. On the SW, URIs indicate entities such as people or places, in addition to content such as text or images.
2. Use of hypertext transfer protocol (HTTP) for URIs, as the key protocol for agents to resolve URIs.
3. Use of RDF and SPARQL 4, for representing and querying SW data.
4. Links to other URIs must be embedded within RDF datasets, in order to discover additional information.

In 2010, Berners-Lee has also developed a 5-star rate scheme, to encourage data owners in various domain areas, such as government, healthcare and multimedia, to expose their datasets as part of the Linked Open Data (LOD) cloud. According to [11], the 5-star system is as follows:

1 Star: “data is available on the Web (whatever format), but with an open license”

2 Stars: “data is available as machine-readable structured data (e.g., Microsoft Excel instead of a scanned image of a table)”

3 Stars: “data is available as (2) but in a non-proprietary format (e.g., CSV instead of Excel)”

http://www.w3.org/TR/rdf-sparql-query
4 Stars: “data is available according to all the above, plus the use of open standards from the W3C (RDF and SPARQL) to identify things, so that people can link to it”

5 Stars: “data is available according to all the above, plus outgoing links to other peoples data to provide context”

Since this scheme has important implications for how data is subsequently manipulated, we adopt it here as one of our evaluation criteria for semantic browsers, as discussed further below.

Dadzie and Rowe [8] recently carried out a survey on current approaches to visualising LD. They classified current semantic browsers into the following two overall types:

Text-based Browsers: these browsers use textual structures such as tables and lists to present LD entities, properties and relationships. Some also use advanced features such as faceted browsing to allow for more intuitive rendering and navigation of data. Examples of such browsers include: Dipper3, Disco4, Marbles5[7], Piggy Bank8[12], Sig.ma9[18], URIburner10 and Zitgist11.

Browsers with visualisation options: these browsers use primarily visual or graphic structures, such as images, maps, graphs and timelines (individually and in combinations) to represent LD. Examples of such browsers include: DBpedia Mobile12[2, 7], IsaViz13, OpenLink Data Explorer (ODE)14, RDF Gravity15, TheRelationshipFinder16[15, 14] and the Tabulator17[17].

In addition to using 14 of the 15 browsers they reviewed, we also adopt this distinction between textual and graphical browsers.

Dadzie and Rowe [8] also define the types of LD users and their different requirements for the consumption and production of LD. Their analysis follows Shneiderman [16], who classifies users into three types:

- Lay users with little or no understanding of the underlying semantic technologies. Such users might use semantic browsers for exploring large data sets or finding particular facts of general interest (on DBpedia for example),
- Technical users with expertise in the SW and LD. Such users might use semantic browsers for data retrieval, integration and analysis (so-called “mash-ups”), using advanced filtering and querying services, and
- Domain experts with expertise in a specific domain, but who may not be familiar with particular SW and LD technologies. Such users, such as medical researchers, might use semantic browsers for advanced domain-specific queries and ontology reasoning.

3 Criteria Framework and Methodology

We first propose five criteria to evaluate the facilities of the browsers surveyed by Dadzie and Rowe [8], with a specific emphasis on their data navigation, triage and manipulation capabilities. We then describe our methodology for evaluating these browsers using these criteria.

3.1 Evaluation criteria

Our proposed criteria for evaluating browsers from a LD perspective are as follows:

1. Data conversion: How easy is it to convert non-LD into LD using the browser?

2. Creating links to other URIs: How easy is it for a user to find new LD and add their own links to data? How easy is it for a user to navigate from a current dataset to a new dataset? How easy is it to navigate forward and backward using the browser? Are the navigation links generic or specific (“context-aware” in some sense)?

3. Data triage: What tools does the browser have to answer queries or questions? How effective is the tool (if any) to sort data based on the users’ needs? Does the query facility support text-search or SPARQL queries or both?

4. Browsing mechanism: What is the browsing mechanism? Does it support faceted, pivoting and “link-sliding” browsing mechanisms?

5. 5-star data: Berners-Lee suggested a 5-star deployment scheme for Linked Open Data18. We consider what level of support is offered for data consumption according to Berners-Lee’s scheme. If the browser also supports the production of data, where data is actively aggregated, augmented or reinterpreted by the browser, what is the level of this support?

Common formats for data include: Excel, CSV, XML, relational database files and RDF. A key aspect of working with structured data is understanding and manipulating the formats they may be represented in. Further, data schemas can vary from one dataset to another. Hence we introduce the Data Conversion criterion to evaluate how well semantic browsers handle conversions between different formats.

Once data is in an appropriate representation, it needs to be linked to other data. For example, when a user wants to look up a URI, this URI has to have some links to other URIs to provide more information about things. Users also often want to be able to create links of their own between heterogeneous datasets. We include the criterion of Creating links to other URIs to describe how well browsers facilitate both navigation of existing links, and the creation of new links between datasets.

We also consider facilities for retrieving, manipulating and reorienting LD sets under the criterion of Data triage and Browsing mechanism. Here we evaluate browsers support for “advanced” features such as SPARQL queries, sorting, faceted and pivoted views, and any other facilities for visualising datasets. Finally, we include Berners-Lee’s 5-star scheme, for considering how well browsers support the consumption and production of LD.

18http://linkeddata.org/journal/2010/06/04/ the-5-stars-of-open-linked-data/
While our evaluation follows that of Dadzie and Rowe [8], our criteria focus more directly on the levels of user interactivity with LD. Hence we evaluate features of greater relevance to technical users and domain experts.

3.2 Methodology
To enable the evaluation of these browsers, we installed all the browsers currently available for testing on our machine as a first step. In some cases, where the “browser” is a web service, this simply involved visiting the relevant URL. In other cases the process involved the installation of a browser extension or desktop application. We performed some initial tests to make sure all the browsers functioned correctly.

The tests also allowed us to familiarise ourselves with how these browsers work and how they can be used, as there are different ways to enable the use of these browsers to explore SW data. During this process, we noted three methods of operation: some worked as independent applications which can be downloaded (IsaViz and RDF Gravity), while some operate as browser extensions (ODE, Zitgist, Marbles, Disco and Tabulator). Another group run within the browser itself (Dipper, Piggy Bank, URIburner and Sig,ma). This in turn has implications for levels of interactivity, as we describe further below.

In this study, two types of evaluation were conducted to provide different methods for examining approaches of existing browsers. The first type of evaluation is a general review of the 14 browsers, to describe our experience and impression of the usability and capability of these browsers from a technical user’s perspective. The second evaluated the browsers according to our five criteria. Both types of evaluation assist in identifying the advantages and disadvantages of the existing browsers for interacting with LD by technical users.

To evaluate the first criterion, “Data conversion”, we used the following three steps to test the conversion process:

1. Find out whether the browser provides a service of converting any of these types: CSV, RDF, HTML or URL into LD style.
2. If so, identify the conversion procedures and whether the process happened on-the-fly or by the user.
3. Then, determine which kind of users can perform the conversion process (lay users or technical users).

The above steps allow us to undertake standard evaluation of the conversion process for non-LD into LD for all semantic browsers.

For evaluating the second criterion, “Creating links to other URIs”, we followed these steps:

1. Start from a URI which is maintained according to W3C standard for SW URIs ‘use so-called “Cool URIs”’ 19.
2. Once the results are retrieved, we look at the URIs to check whether they “Cool URIs” or not; if so, we use the “click” action to navigate to another URI that can fetch up a different datasets.

For example, we use the URI “http:// Dbpedia.org/resource/Berlin” to explore data about Berlin starting from the DBpedia dataset (a LD version of Wikipedia) and then navigate to the other datasets using the URIs that appear in the results.

The third and fourth criteria are evaluated by observing the browsers features that are related to both. In criterion 3, if a browser provides support for a SPARQL endpoint, we run an SPO (“Subject Predicate Object”) SPARQL query and test for the endpoints. The text search provided by some browsers are also evaluated by typing a simple query. In criterion 4, the evaluation was conducted by observing the mechanism view of browsers.

The fifth criterion evaluates the level of data support within the browsers. Since all the browsers are semantic-based, they all support data in RDF format. However, for those that did not directly support LD facilities, they were scored 4 rather than 5 stars. A 5 star rating would be applied only when data is linked to another URIs. In addition, some browsers actively convert RDF data, producing integrated views of more than one data set. We distinguish this capability in terms of “producing” rather than simply “consuming” LD, and also rated this feature accordingly. Our method to evaluate links for this criterion extended the approach in criterion 2 by testing links between multiple datasets to identify whether the data are discoverable or not. The test included both inward and outward bound links.

4 Evaluation Study
Our evaluation is in two parts. First we review the 14 semantic browsers to highlight their general usability, and specific functionality that supports user interaction. We then conduct the evaluation against our five criteria.

4.1 Review of SW browsers from a technical perspective
We follow the classification structure as well as the actual browsers analysed by Dadzie and Rowe [8], as described in Section 2. We have, however, excluded one browser, namely, Zitgist that seem to be a defunct project at present. Here we focus on the level of user interaction supported, in particular on the functions provided for technical users to explore heterogeneous data.

4.1.1 Text-based browsers
Text-based browsers rely predominantly upon textual representation to present LD resources.

Dipper is a text browser that allows for the exploration of RDF data stores. In our initial review, the aims of Dipper appears ambiguous; supporting documentation is sparse, and does not describe the way the software works, or even what it was designed for. Dipper’s user interface is not particularly intuitive, as it is hard for even advanced users to determine where to source and consume data they need. Dipper does however provide links to some of the public RDF stores, such as OpenLibrary, NASA,
BBC-Backstage and data.gov.uk. Most datasets are stored on repositories provided by the developer of the browser, Talis.

We observed that there did not seem to be an organising principle on how datasets were collected together. For instance, the only airport we found in the airport dataset is Birmingham International Airport. The data provider has not given any explanation as to why these datasets were chosen. This means queries provide ad hoc and partial results, without obvious means for loading further datasets and extending these results. In addition, during testing we faced error messages such as “Could not retrieve data: error”, without providing any explanation on why that error has occurred.

In terms of LD, the browser does not provide a mechanism to go beyond the current data store to find other data. Navigational control is limited, without backward and forward options, and data retrieval also seems slow.

**Disco** is described as a simple browser for navigating the SW as an unbound set of data sources. It renders all information it finds on a particular resource as HTML. It allows navigation between SW resources by dereferencing HTTP URIs and by following rdfs:seeAlso links.

**Marbles** is an application that resides on the server formatting SW resources for HTML clients, such as HTML browsers, through using Fresnel lenses and formats. Its user interface is sparse, which in certain contexts may be a strength as this would not overwhelm an average user exploring the SW. When provided with a URI to display, it tries to dereference it, as well as querying Sindice and Falcons for a data source that may contain information about that resource. It also uses owl:sameAs and rdfs:seeAlso to retrieve more data about the resource in question. For example, providing a URI such as [http://www.w3.org/2005/04/fresnel-info/](http://www.w3.org/2005/04/fresnel-info/) to display, it retrieves and formats the resource for suitable HTML viewing.

**Sig.ma** is an application and a browser that integrates LD from multiple sources allowing data navigation (see Figure 2). The initial interaction is driven by the user with a free text search, which is a useful way to begin data exploration especially when compared with the other browsers which prompt for a URI to begin data exploration. The ability to use a URI as an entry point is also still available for users. Sig.ma is built on top of Sindice, a semantic search engine, that provides a search service allowing technical users to find resource descriptions.[20]

**URLburner** is a software service that delivers structured data about Web resources, generating an RDF graph of the resource’s metadata using existing well-known ontologies as well as site-specific knowledge. It takes a Web URI, and can represent the dereferenced resource as HTML or RDF. It also provides a Firefox extension which can be used to bookmark the URI of interest.

**4.1.2 Browsers with visualisation options**

In this section, we discuss browsers and applications that use graphic visualisations to represent SW resources.

**DBpedia Mobile** is a client application, designed for mobile phone use, that allows users to access information about objects stored in DBpedia. For example, real-world entities such as cities, streets and landmarks that have been described and asserted in the DBpedia knowledge base can be queried automatically based on a user’s current geolocation. DBpedia augments these resource descriptions so that users can explore other resources on the SW. DBpedia Mobile uses Marbles to render Fresnel-based views of these Web resources that have information in DBpedia. Users can moreover use these data as “jumping-off” points to other SW datastores such as GeoNames, Eurostat and Revyu.

**Fenfire** is a service that enables users to explore LD by dereferencing URIs, and following available rdfs:seeAlso properties to retrieve related
datasets [10]. The user can enter the starting URI to begin navigating LD.

**IsaViz** is a visual application for browsing and editing RDFs as graphical objects. The software helps to see the graph structure of the RDF. Clicking on a vertex node of the RDF graph brings up the textual descriptions of the vertex. Users can further edit names of RDF nodes, as well as add and delete both nodes and links between nodes.

**LESS** enables users to create their own templates to view LD [1]. It uses the LESS Template Language (LeTL) which define a text-representation to view the output from URI dereferencing or SPARQL queries.

**OpenLink Data Explorer** (ODE) is a browser extension that allows Web users to explore raw data and relationships of a Web page. Once installed, when a user is browsing a Web page which they would like to obtain LD for, they simply click on the ODE extension, and that Web page is turned into a suitable LD format viewable in a browser.

**RDF Gravity** is another application for visualising RDFs as graphs and the ontologies. It allows a user to customise the view of results by using local filters.

**TheRelationshipFinder** (RelFinder) is a Web application for exploring connections between objects in a SW knowledge base. It offers a new way to get insights on how instance data in a SW knowledge base relate together. The knowledge base used in RelFinder is the DBpedia dataset.

**Tabulator** is an RDF browser, designed both for users and developers of RDF content. A key feature is it allows RDF data providers see how their data interacts with the rest of the SW. The browser is designed to be as easy as possible for a new user to pick up, and easy for developers to extend with their own ideas. Our experience of trying Tabulator came up with a lot of networking errors due to some technical limitations in the tool. None of the available versions worked particularly well, posing difficulties for more detailed evaluation. This evaluation is assisted therefore from supplementary resources: a movie demonstrating the browser (http://dig.csail.mit.edu/2007/tab/tutorial/editing.mov) and an associated paper [17].

### 4.2 Evaluating browsers using our criteria

In this section, 14 semantic browsers are evaluated according to our five criteria described in Section 3.

#### 4.2.1 Data conversion

Some browsers, such as Dipper and Sig.ma, do not support any data conversion capabilities. Dipper assumes that the resource in question is already in LD format. Sig.ma handles a mixture of LD and non-LD but does not facilitate the conversion of non-LD to LD. The Marbles browser supports data conversion of HTML; users can view any Web page as LD through the click of a button. Piggy Bank has similarly impressive function that uses a screenscraping approach to turn the HTML content into RDF content. However, the available screenscraping is limited to a number of websites such as the ACM Portal Scraper and the Flickr Photo Scraper. Additional screenscrapers need to be developed for other websites in Javascript or through use of an associated tool, Solvent, based on an awareness of the underlying document structure of the targeted website.

URIburner has extended facilities that make it fairly trivial to convert non-LD into LD, so long as the original sources are interpreted via an associated Virtuoso middleware platform. Together with Virtuoso, URIburner provides the following URL patterns for browsing the description of URIs:

**HTML:** http://linkeddata.uriburner.com/about/html/[URLscheme]/[hostname]/[localpart]

**RDF:** http://linkeddata.uriburner.com/about/rdf/[URLscheme]/[hostname]/[localpart]

where rdf can be replaced with any other format such as: xml, n3, nt or ttl.


Although DBpedia Mobile uses some of the capabilities of Marbles to explore data, it does not provide a direct facility for data conversion. The client browser starts the exploration from DBpedia. IsaViz, however, does not provide this feature for users; it is only useful for browsing LD that are already in RDF.

In ODE, a browser extension\(^\text{22}\), it is easy to convert any HTML page into LD. Figure 4 shows an

---

\(^\text{22}\)http://ode.openlinksw.com/
Table 1: Evaluation of textual LD browsers from an interactivity perspective

<table>
<thead>
<tr>
<th></th>
<th>Dipper</th>
<th>Disco</th>
<th>Marbles</th>
<th>Piggy Bank</th>
<th>Sig.ma</th>
<th>URIburner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data conversion</td>
<td>✓</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Unstructured data to RDF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structured data to LD</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Export RDF/XML, JSON format</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Creating links to other URIs</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Forward navigation</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Backward navigation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDF model navigation</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Links filtering</td>
<td>✓</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Navigating Local LD</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Navigating Global LD</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Data triage</td>
<td>✓</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Simple (keyword search or filter)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Complex (SPARQL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use of search engine</td>
<td>✓</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Caching results</td>
<td>✓</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Aggregated results</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Editable results</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Tagging results</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Browsing mechanism</td>
<td>✓</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Entity-Attributes-value view</td>
<td>✔</td>
<td>✓</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>HTML view</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Facet view</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Map view</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Visual view</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grid view</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Layout control</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>&quot;Order by&quot; option</td>
<td>¥</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>5-star data schema</td>
<td>Producer 5 stars</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>Consumer 5 stars</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

Table 2: Evaluation of visual LD browsers from an interactivity perspective

<table>
<thead>
<tr>
<th></th>
<th>DBpedia Mobile</th>
<th>FeriVe</th>
<th>LESS</th>
<th>ODE</th>
<th>RDF Gravity</th>
<th>ReFinder</th>
<th>Tabulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Conversion</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Unstructured data to RDF</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Structured data to LD</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Export RDF/XML, JSON format</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Creating links to other URIs</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Forward navigation</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Backward navigation</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>RDF model navigation</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Links filtering</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Navigating Local LD</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Navigating Global LD</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Data triage</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Simple (keyword search or filter)</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Complex (SPARQL)</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Use of search engine</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Caching results</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Aggregated search results</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Editable results</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Tagging results</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Browsing mechanism</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Entity-Attributes-value view</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>HTML view</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Facet views</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Map view</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Visual view</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Grid view</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Layout control</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>&quot;Order by&quot; option</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>5-star data schema</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td></td>
<td>Producer 5 stars</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Consumer 5 stars</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>
example of converting a Wikipedia page to LD. As shown on screen A, the RDF detective service (A-2) allows for the extraction of RDF triples embedded in HTML, which in this case yields 212 triples (A-1). To convert the contents of the Wikipedia page from hypertext to LD, ODE (A-3) extracts the raw data and displays it as LD. The output from the data conversion process also shows related data resources, here resulting in the 2470 triples shown in (B-1). However, we noted that commonly used structured data formats, such as CSV, cannot be converted by this browser.

RDF Gravity, RelationshipFinder and Tabulator do not facilitate converting data. The RelationshipFinder does however work on RDF datastores that contain LD in RDF form.

Generally, browsers can be discriminated in terms of facilities for accessing and combining resources that have different formats, schemas and ontologies. More capable browsers in this regard will allow users to consume, explore and interact with LD more efficiently.

4.2.2 Creating links to other URIs

Dipper is a closed system that does not permit exploration of arbitrary URIs. RDF data stores or other sources are only given the opportunity to explore LD that is stored inside Dipper datastores. Although there are labels for Lookup, Store and Starting Points in Dipper, however, it is hard for the user to realise where to start from to consume the data they need. The user has to explore and click the links to get started but in terms of what should the user click, the browser does not facilitate this for users. Also, the predicates given to explore datasets are not very visible and obvious, e.g., http://ckan.net/ontology/downloadURL is a predicate given for freebase.

The Marbles browser allows the user to navigate to new links. It also gives the user details of the provenance of the data, i.e., where the data was retrieved from. Piggy Bank has a notion of Semantic Bank where multiple users can store and share the Piggy Bank–converted SW contents. This allows new LD to be found and linked to other datasets. However, it does not allow global discovery of new LD resources. Marbles and Piggy Bank are available as Firefox extensions, therefore users can conveniently navigate backward and forward using the browser’s in-built navigational features.

Sig.ma facilitates the user to find new LD from the results of queries, through navigation of the links provided in the search results. URIburner allows users to navigate from one domain of LD to a different one linked from the current dataset. It supports both forward and backward navigation.

DBPedia Mobile has good highlighting of links between datasets. However users can only navigate to selected datasets on the SW, such as Geonames, Revyu, Eurostat and Flickr. IsaViz service does not support navigating links into another datasets. It only supports browsing a single RDF graph.

ODE provides the user with the facility of navigation between datasets. So the user can click on other LD from the given results to navigate to another source. In ODE, users also can view data based on previously selected predicates: links will then be traversed based on their custom properties as shown in Figure 4 (B-2).

RDF Gravity does not provide a facility for highlighting links between different data resources.

The RelationshipFinder gives users the relationships, if any, between different RDF nodes. However, users do not have the opportunity to navigate to new LD using this application. They can find the relationships between existing LD in an RDF database using this application. One can save the results of the search using a URI and this URI can be dereferenced.

LESS and Tabulator provide support for exploring new LD by following links. However, the user can not custom the browsing data based on the data type.

4.2.3 Data triage

In Dipper, there is no specific support for sorting data or constructing queries on new data. Some datasets with public SPARQL endpoints can be queried and explored, but users need to know about SPARQL query language to write a query. This feature can be leveraged by developing a client user interface for these SPARQL endpoints. However, not all datasets provide a SPARQL endpoint service for accessing their data using the SPARQL query language. Also, some of the default datasets, such as http://lists.broadminsteruniversity.org/lists/demo, need official authorisation of a user’s credentials to explore the data, which imposes challenges during initial evaluation of the tool.

The Marbles browser offers a SPARQL endpoint, that is a useful facility for technical users, though less so for lay users and domain experts. Marbles uses the search engines Sindice and Falcons for querying a user-entered URI once it has been dereferenced. It can then combine data from different resources into a single view. Marbles does however have limitations; it does not support advanced interaction features such as exporting data or tagging result sets.

Piggy Bank also provides users some facilities for data triage, such as text search and tagging. However, there are no facilities for SPARQL queries or for result aggregation, and domain specific support is also limited.

In Sig.ma, the search terms are text-based, and users can search based on RDF properties. As shown in Figure 2, Sig.ma uses a mixture of query planning, word disambiguation, distributed data source selection, and parallel data gathering to return a list of results for the search query. The confidence or quality of the search results is also provided, a strength of Sig.ma. This allows the user to validate and verify the result even after search results have been shown. For example, if the user is not satisfied with any of the results, they can ask for those results to be changed, resulting in improved results in future searches.

The URIburner service makes use of a public SPARQL endpoint. There is also a Search and Find browser endpoint at http://linkeddata.uriburner.com/fct/. Both free text search as well as URI-based searches are supported.

DBpedia Mobile provides for both simple filters that allow simple text and list based querying facilities, and complex SPARQL queries for complex manipulation of linked datasets. DBpedia Mobile also generates a summary view of selected resources by building on the Marbles browser, using semantic search and integrating the results.

IsaViz does not provide any support for data triage, while ODE supports basic sorting of data based on user preferences. As shown in Figure 4 (B-3), ODE users can retrieve and view the requirements based on either an automatic or editable data retrieval schema, and the depth of the link traversal.
can be adjusted by the user. It provides support for caching results as well.

RDF Gravity has three simple query facilities: support for RDF query language (RDDL — a precursor to SPARQL) so a technical user can write queries based on their needs; support for text search over concepts, properties and instances specified in a RDF data set; and enabling global and local filters to specify the view of results.

RelationshipFinder provides free text search of objects or elements in a plain text view. The user enters two objects of interest to them, which are preferably described by articles in the English Wikipedia. These selected elements are first semi-automatically mapped to unique objects of SW datasets in the underlying RDF database. These datasets are then crawled for relationships to present to the user.

Finally, in Tabulator, the user can select fields to use as simple query parameters. Due to current technical limitations, query and manipulation options could not be further explored.

To conclude this part, some of the more functional browsers demonstrate impressive capabilities for data triage, retrieval and extraction. In particular, they show strong support for SPARQL, providing a flexible range of options for sorting, retrieving and finding new data. In cases where datasets can not be queried via SPARQL, such as when an endpoint is not exposed, some of the browsers also utilise text-based search for data retrieval. In the case of Sig.ma, result sets can also be reviewed, approved and rejected, providing useful feedback to the search heuristics and algorithms used.

### 4.2.4 Browsing mechanism

Browsing mechanisms for visual browsers differs considerably. At one end of the spectrum, some, such as Dipper and Marbles, use basic HTML text representation, making them closer in approach to pure text-based browsers. Marbles also includes a text summary feature that describes resources.

Other browsers use more advanced visual approaches. Sig.ma, for example, enables users to browse search results by dividing the screen into two panels: the left panel is used to view the aggregated data sources, and the right panel is used to present the facts between the resources. Although it has a search engine, Sindice, whose features and APIs can be leveraged to perform faceted browsing, Sig.ma only supports "flat" one dimensional browsing.

In contrast, other browsers offer different data views. Piggy Bank has a faceted view that provides a summary about an item with different aspects. URIburner presents data in the form of "Entity-attributes-value", which uses textual representation for viewing information. It also provides an extra feature that enables users to pivot collections around the current data view.

The browsing mechanism in DBpedia Mobile is a map view complementing HTML information, as well as, where available, some visualisation of specific objects themselves. It also gives the user the opportunity to have view control over the results by using zoom-in and zoom-out options.

In IsaViz, RDF data is presented as graphs, which can be navigated along graph vertices and edges. ODE supports multiple ways of viewing data, including faceted view, grid view, and visual view, and also includes data ordering options as shown in Figure 4(B-5)-(B-6). RDF Gravity also supports graph-based views, while RelationshipFinder supports both graph- and tree-based views. Finally, in Tabulator, there are different views, such as map view, timeline view and graph view. Due to difficulties mentioned earlier in Section 4.1.2, we were unable to effectively evaluate Tabulator.

#### 4.2.5 5-star data

Dipper scored a five star rating since it supports LD objects — using URIs to identify entities and RDF to represent data. As discussed earlier, there were however some problems with actually following LD in this tool. Marbles also scored five stars, although it only consumes and cannot produce LD output.

Much of the data in Piggy Bank is a mixture of 2 star, 3 star and 4 star — that is, it is represented in more or less structured formats. Increased usage of the SW could raise some of this data to 5 stars, as more of it becomes linked — however, in some cases the browser does not provide a full context to users, as there are no outgoing links to follow in the datasets currently used.

The data support in Sig.ma and URIburner are both 5-star, but in the case of URIburner some links needs to be further curated, in order to support further resource discovery. DBpedia Mobile is a 5 star system as it produces RDF data, and also provides links to other external datasets such as GeoNames, Revyu, EuroStat and Flickr.

ODE is also rated at 5 stars as it produces LD in RDF, and also allows navigation to other external data linked to and from the produced LD. Tabulator similarly produces new data that is structured, and provides links to other datasets, so this is also rated 5 stars.

Finally, LESS, IsaViz, RDFGravity and ReFind offer only semi-structured data, and therefore do not produce LD output.

### 5 Conclusion and Future Work

This study set out to determine the criteria to evaluate the interactive capabilities of semantic browsers. The main focus here is on user interaction features, which are particularly useful to more technical users, for exploring and consuming LD. An extension of this study would investigate more deeply organisational factors, HCI relevant issues and query implementation issues.

For the LD world to be useful to the end users, powerful semantic browsers need to be developed that can perform well against the kinds of criteria we presented here. First, browsers should provide support for data conversion, for users to convert their data from non-LD into LD. Second, browsers should allow for the creation of links to other URIs to connect more information. Third, browsers should provide query capabilities for data triage along with result sets that can be viewed in different ways by users. More importantly, SPARQL support is crucial as it allows results to be extracted readily from very large datasets. Fourth, users will interact with LD if the browsing mechanism is easy and flexible to use, and presents data in an intuitive way. Fifth, browsers should support highly structured and LD, in accordance to Berners-Lee’s 5 star criteria.

A further finding of our study suggests that the Dadzie and Rowe [8] classification between text and visual semantic browsers can be further distinguished into 4 kinds: RDF model browsers, domain-specific browsers, generic-domain browsers and links-relationship browsers. Selection and ranking of these browsers depends upon end user requirements.
those requirements are limited to the use of RDF format only, then IsaViz or RDF Gravity rate highest in terms of capabilities. For domain-specific purposes, where the data set is predefined, then the Dipper, Piggy Bank or RDF Gravity browser would be the most appropriate according to our evaluation. For exploration of the global SW, browsers such as Marbles, Sig.ma and URIburner provide text-representation of data, while browsers such as ODE and Tabulator provide more options for data visualisation. For the understanding of the relationships between linked datasets, RelFinder presents the relationships in an intuitive manner.

The outcome of this study indicates that the Sig.ma, URIburner and ODE are clearly the most powerful browsers currently available for exploring the Web of LD. They provide sophisticated functions that manipulate, integrate and explore different data resources and, this is of particular benefit to more technical users.

We also observe that the Marbles, Sig.ma, URIburner and ODE browsers have shown more capabilities for exploring and manipulating data, and therefore were evaluated more highly in this study than their corresponding evaluation according to the criteria presented by Dadzie and Rowe [8]. These browsers substantially reduce the challenge of user interaction with LD. Our evaluations of Dipper, Disco, the Tabulator, DBpedia Mobile, and RelFinder arrive at the same outcome as Dadzie and Rowe [8], which suggests that these browsers are also useful when applied to interacting with LD. However, in our study, Piggy Bank, Fenfire, IsaViz, LESS and RDFGravity were evaluated more poorly, which suggests that their strengths lie more on the visualisation of RDF data than in its exploration and manipulation.

The future development of interactive browser capabilities based on the suggestions put forward in this study will further bridge the gap between the world of LD and human users. Improved LD visualisation and interaction tools will enable users to more readily comprehend LD.

Acknowledgment: This research was supported under Australian Research Council’s Linkage Projects funding scheme (project number LP0990509).

References


Discovering Social Media Experts by Integrating Social Networks and Contents

Zhao Zhang  Bin Zhao  Weining Qian  Aoying Zhou

Abstract

Social media are media contributed by common users and distributed in social networks. There may exist thousands of answers to a single question provided by different users. However, it is difficult to evaluate the authority of a user to a specific question. We introduce a new method for identifying experts in social media. Both the structure of the social network and content of the media are used in a unified graph model for evaluation of users. Extensive experiments show that our approach can determine authority experts on specific domains.

1 Introduction

Social media contains huge volume of information. However, it is difficult to filter noise and low quality data out from social media. To find experts to a specific topic and then collect content contributed by experts to the topic is a natural way to acquisition of high-quality knowledge. It has proved to be an effective, and attracts much attention in social media mining research [10, 6]. However, existing methods are usually designed for a specific type of social media, such as blogs [8], online focusms [19], and microblogs [9]. The structure of social networks and contents of the media are considered separately. We take another approach, which aims at the general problem of expertise search on social media. It relies on a unified graph model. The expert finding problem is then solved via a mutual reinforcement process in the network.

In this paper, our work provides comprehensive expertise analysis in general social media. The following two questions should be answered to solve the problem.

Who are experts to a specific topic? We call this task as expert finding. That is to say, given a topic query (describing the area in which expertise is being sought), a ranked list of user names is returned.

Which topics is she an expert in? We call this task as expert profile finding. In other words, given a user query (describing the user in which expertise is being sought), a ranked list of topics is returned.

There are several challenges to expertise analysis. First, social media is the mixture of social network and content. A typical social media site is shown in Figure 1. It contains several posts and part of their comments. A set of users build relationships by commenting the post. In other words, social networks and contents are mutually reinforcing. That is to say, if a user is an expert of a specific area, the users who have strong social relationship have high probability to be experts on the same area. So we need to find a model to represent this indirect relationship. In this paper, we propose a novel graph model to represent contents and social networks simultaneously.

Second, the lists of friendship and followship embed much noise. Since there are a large number of inactive users in social media sites, we cannot rely on friendship list to build social networks for social media sites. We have to extract active social relationships among users. In this paper, we only consider active social networks built by posting and commenting among users.

1.1 Our contributions

We made the following contributions to attack the problem of expertise search on social media in this paper.

• A tripartite graph model is introduced, which simultaneously represents features of social networks and contents in social media. This graph model makes our analysis simple and convenient.

• Active social relationships are used for expertise analysis. Since lists of friendship and followship embed much noise, only active social networks built by posting and commenting among users are used.

• Expert and expert profile finding are formally defined. Social media has been saturated with a large number of human generated contents. There exist many folk experts in social media. In this paper, we present formal definitions about expert and expert profile in social media sites.

• A random walk with restart (RWR) algorithm for tripartite graph is presented. Many researches showed RWR is a good correlation measurement method between nodes in graph. However, the main challenge of RWR algorithm is its efficiency. In this paper, we present an improved RWR algorithm for large tripartite graph based on star schema.

• Extensive experiments over real life data sets are conducted. We compare our algorithm with the
initial RWR on two real data sets. Our experimental results (see section 6) show significant benefits in time consumption.

1.2 Paper organization

The rest of this paper is organized as follows. The problem of expertise analysis is formally defined in Section 2. Section 3 introduces the random-walk-with-restart (RWR) algorithm. In Section 4, a star-schema-based optimization technique for RWR in tripartite graph is presented. The procedures for expert and expert profile finding are introduced in Section 5. Experimental results are shown and analyzed in Section 6. The related work are introduced in Section 7, followed which Section 8 is for concluding remarks.

2 Problem statement

A unified tripartite graph model that represents both content and structure of social networks is introduced in this section. It is the basis of expert and expert profile finding, which is introduced in detail in Section 5. The symbols and notations that used are listed in Table 1.

2.1 Social media preliminaries

There are two types of entities in social media, i.e. users and pieces of information. Pieces of information may contain multimedia content. In this paper, only content of text is considered. Both text content and other types of multimedia content can be handled via semantic annotation.

![Figure 1: An example of social media](image_url)
Usually, users are connected via social networks. Relationships between users include, for example, following in Twitter, or friendship in Facebook. However, these types of relationships are relatively static. We argue that active social networks are more important than static relationships. Here, active social networks are social networks in which relationships capture the interactions between users. Such kind of dynamic relationships include retweeting in Twitter, like in Facebook, and commenting in online forums.

Thus, there are four types of information that should be included in the unified model:

**Users** A user is essentially an identifier identified as the author of any pieces of information or entities involved in a social network.

**Texts** Text is a piece of information in text form. It is used to represent the original form of content contributed by users.

**Active social networks** An active social network is the social network that captures dynamic relationships implying interactions between users.

**Terms** A term is a semantically meaningful word or phrase that represent the semantics of texts. Note that a text may be annotated by several terms, while a term may be used to annotate multiple texts.

An expert query is a set of terms, while the result should be a ranked list of experts who are good at topics defined by those terms. The list is ranked in descendant order based on the goodness of experts. An expert is also a user. It is formally defined in Definition 1.

Definition 1 An expert query \( Q_e \) is a set of terms: \( \{ t_1, t_2, \ldots, t_k \} \), in which each \( t_i \) is a term. The result of \( Q_e \), denoted as \( R_{Q_e} \), is \( < u_1, u_2, \ldots, u_k > \) satisfying that \( r^e(u_i, Q_e) \geq r^e(u_{i+1}, Q_e) \). Here, \( r^e(u_i, Q_e) \) is a score function that denotes the possibility of user \( u_i \) being experts on the domain defined by \( Q_e \).

Slightly different, an expert profile query is a set of users (experts). The result should be a ranked list of terms which denotes the domain(s) those experts are good at. It is formally defined in Definition 2.

Definition 2 An expert profile query \( Q_p \) is a set of users \( \{ u_1, u_2, \ldots, u_n \} \), in which each \( u_i \) is a user. The result of \( Q_p \), denoted as \( R_{Q_p} \), is \( < t_1, t_2, \ldots, t_l > \) satisfying that \( r^p(t_i, Q_p) \geq r^p(t_{i+1}, Q_p) \), in which \( r^p(u_i, Q_p) \) is a score function that denotes the authority degree of user group \( Q_p \) on domain denoted by term \( t_i \).

In real-life applications, the expert and expert profile queries are top-k queries. Thus, only top-k users and terms with highest score function values are to be returned.

Thus, the essence of the problem is a reasonable definition of score functions \( r^e() \) and \( r^p() \), and efficient search of \( u_i \) and \( t_j \) with top-k values given queries and score functions.

2.2 Conventional model for modeling contents

A simple yet natural way for expert and expert profile finding is to directly analyze social media content, e.g. texts and terms. Bipartite graphs, as it is shown in Figure 2, are often used to model relationships between users and terms. The vertices on the left are users, while those on the right are terms. The edges are weighted, in which weights are term frequency of a user mentions a specific term. Bipartite graphs are often used in text mining. Since the structure of social networks, whether static ones or dynamic ones, are not used, we argue that this model may not capture the important features of users’ expertise.

2.3 Tripartite graph model

Intuitively, users and terms are not directly connected. Terms are actually associated with users’ actions, such as posting, commenting, or retweeting. Thus, we extend the conventional bipartite graph model to a tripartite graph model. A tripartite graph have three types of vertices and two types of edges. The first type are used to represent users, while the second type is for their actions, and the third one is for terms. It is formally defined in Definition 3.

Definition 3 A tripartite graph \( G \) is defined as a quintuple \( (V_1, V_2, V_3, E_1, E_2) \), in which \( V_1 \) is the set of users \( \{ u_1 \} \), \( V_2 \) is the set of texts \( \{ p_1 \} \), and \( V_3 \) is the set of terms \( \{ t_j \} \). \( E_1 \) is the set of edges \( \{(u_i, t_j)|u_i \in V_1, t_j \in V_2\} \subseteq V_1 \times V_2 \), while \( E_2 \) is the set of edges \( \{(p_i, t_j)|p_i \in V_2, t_j \in V_3\} \subseteq V_2 \times V_3 \).

There is an edge \( (u_i, p_j) \) in \( E_1 \) if that user \( u_i \) contributes the piece of information \( p_j \), while the semantics of \( p_j \) is represented by terms that are connected to \( p_j \) by edges in \( E_2 \). Thus, actions of users can be represented by this tripartite graph. Furthermore, for dynamic relationships between users, such as commenting, retweeting, and etc., users interact with each other are both connected to the same \( p_j \), and thus establish an indirect relationship in the tripartite graph. Thus, the active social network is successfully embedded into our tripartite graph.

A tripartite graph corresponding to the posts in Figure 1 is shown in Figure 3.

3 RWR in tripartite graphs

Though the tripartite graph model elegantly captures the (active) structure and content of social media, the definition of score functions used in expert...
and expert profile queries are not implied intuitively. The problem of expert and expert profile finding are essentially ranking correlation score between users and terms.

Several link-based relevance functions have been proposed in graph, including simrank [5] and random walk with restart (RWR) [4]. SimRank can compute relevance of a node-pair \((a, b)\) based on similarity of multi-step neighborhoods. RWR can simultaneously obtain relevance scores between given node \(a\) and other nodes except for node \(a\) in a graph. Considering efficiency and effectiveness of the algorithm in large graphs, we adopt RWR approach in this paper.

We define \(r^e(u_i, Q_v)\) as the sum of \(r^e(u_i, t_j)\) where \(t_j \in Q_v\) are terms in the query, i.e.

\[
r^e(u_i, Q_v) = \sum_{t_j \in Q_v} r^e(u_i, t_j).
\]

Similarly, \(r^p(t_i, Q_p)\) is the sum of \(r^p(t_i, u_j)\) where \(u_j \in Q_p\) are users in the query, i.e.

\[
r^p(t_i, Q_p) = \sum_{u_j \in Q_p} r^p(t_i, u_j).
\]

Given the tripartite graph \(G\), RWR is a natural way for definition of \(r^e(u_i, t_j)\) and \(r^p(t_i, u_j)\) [15], which can be defined by Equation 1, in which \((1 - c)\) is a random particle that starts from vertex \(i\). Matrix \(w\) is a transition matrix for graph \(G(V_1 \cup V_2 \cup V_3, E_1 \cup E_2)\) transformed from tripartite graph \(G(V_1, V_2, V_3, E_1, E_2)\), with column normalized. Elements in each column sum up to 1. \(\overrightarrow{r}_i\) is a vector that the \((i\text{-th})\) element is 1 and other elements all are 0. Equation (1) is convergence which has been proved in reference [13].

\[
\overrightarrow{r}_{i+1} = (1 - c)w \overrightarrow{r}_i + (c) \overrightarrow{e}_i
\]

\[
\begin{align*}
a_1 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad p_1 & \quad t_1 & \quad t_2 & \quad t_3 & \quad t_4 \\
a_2 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 1/4 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \\
a_3 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \\
a_4 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \\
p_1 & \quad 1 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 2 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \\
p_2 & \quad 0 & \quad 1 & \quad 2 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 2 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \\
p_3 & \quad 0 & \quad 0 & \quad 1 & \quad 2 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 2 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \\
t_1 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 3 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \\
t_2 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 1 & \quad 3 & \quad 1/4 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 \\
t_3 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 1/4 & \quad 1 & \quad 0 & \quad 0 \\
t_4 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 0 & \quad 1/4 & \quad 0 & \quad 0 \\
\end{align*}
\]

Figure 4: Transition Matrix of Figure 3

Actually, as it is stated in reference [11], gave a node \(u_i\), to compute the relevance score of \(t_j\), it can be obtained via several random walks starting from \(u_i\), and count the number of times that we visit \(t_j\). This count reflects the relevance of between \(u_i\) and \(t_j\). The probability of visiting \(t_j\) from \(u_i\) is the relevance score we need.

We can use a \((|V_1| + |V_2| + |V_3|) \times (|V_1| + |V_2| + |V_3|)\) matrix \(w\) to represent a tripartite graph. If there is an edge from node \(i\) to node \(j\) then \(w_{i,j} = 1\), otherwise \(w_{i,j} = 0\). Figure 4 shows an example \(11 \times 11\) transition matrix \(w\) of above tripartite graph (Figure 3).

\[
r(v_i, v_j) = \sum_{v_i \rightarrow v_j} p(\pi)c(1 - c)^{\text{length}(\pi)}
\]
To evaluate the score function, it can be observed that:

\[ r(u_i, v_j) = r(u_i, p_k) + r(S_k, S_l) + r(p_l, t_j), \]

in which \( r(u_i, p_k) \) and \( r(p_l, t_j) \) can be directly obtained given the stars, and \( r(S_k, S_l) \) denotes the relevance score in the new transformed graph where \( S_k \) and \( S_l \) are vertices in the new graph that correspond to stars centered at \( p_k \) and \( p_l \). Thus, the problem of evaluation of \( r(u_i, v_j) \) is transformed to the problem of evaluating \( r(S_k, S_l) \) over the new graph. Thus, the size of transition matrix is reduced from \( (|V_1| + |V_2| + |V_3|) \times (|V_1| + |V_2| + |V_3|) \) to \( |V_2| \times |V_2| \). Thus, the process of RWR can be much more efficient in the decomposed graph compared with that in the tripartite graph. Experimental results reported in Section 6 verify our approach’s efficiency and effective.

5 Expert and expert profile query processing

Given the relevance score evaluation method introduced in Section 4, in this section, we introduce the whole process for expert and expert profile query processing, which is made up of four steps, as it is illustrated in Figure 7. The details on those four steps are introduced as follows.

5.1 Step 1: Construction of the tripartite graph \( G \)

The social media content are parsed. The bipartite graph of users and texts are constructed. Then, after the semantic annotation of the texts, the whole tripartite graph is constructed.

5.2 Step 2: Construction of the transformed graph \( G_s \) given the tripartite graph \( G \)

Intuitively, a star is a summary of a portion of the original tripartite graph. In this step, firstly, we find all stars \( S_l \) in the tripartite graph. Then, the star graph \( G_s \) is constructed, where stars \( S_l \)'s are treated as vertices, while the relationships between vertices, i.e., edges, are established and weighted.

Algorithm 1 shows more details about the procedures of constructing star graph \( G_s \) given the tripartite graph. This is a costly procedure. However, the computation can be offline and incremental. Thus, it will not affect the query processing performance.

5.3 Step 3: Constructing query graph \( G_q \) on the basis of star graph \( G_s \)

When a query \( Q_e \) (or \( Q_p \)) is submitted, we only need to add a query node and corresponding edges to star graph \( G_s \). If \( Q_e \) (or \( Q_p \)) and a star \( S_l \) have common vertices, an edge that connecting query node \( Q_e \) (or \( Q_p \)) and star component \( S_l \) is added. The weight of this edge is the number of common vertices of query \( Q_e \) (or \( Q_p \)) and star \( S_l \).

Algorithm 2 shows more details about constructing query graph \( G_q \) based on star graph \( G_s \).

5.4 Step 4: Finding experts \( E_t \) or expert profile \( E_p \) in graph \( G_q \)

5.4.1 Finding expert \( E_t \)

After the star graph is constructed, an inverted list for search of stars \( S_l \) given a vertex \( u \in V_1 \) is constructed. When a query is posed, after conducting RWR on query graph \( G_q \), we can get relevance scores of query node and each star \( S_l \). Then, for all \( u \in V_1 \), we only need to accumulate the relevance score of query node to each star \( S_l \in S_u \) contained vertex \( v \). The accumulation value is the relevance score \( r(u, Q_e) \). Afterwards, we rank all \( u \in V_1 \) based on the relevance scores, and get top \( k \) u’s. They are experts \( E_t \) to query \( Q_e \).

Algorithm 3 and 4 show more details about finding experts \( E_t \).

5.4.2 Finding expert profile \( E_p \)

The problem of expert profile query processing is symmetric to the expert query processing. The process of finding experts can be easily adapted for finding expert profiles. Therefore, we omit the details here.

6 Empirical study

In this section, we perform extensive experiments to evaluate the performance of our algorithm on two real-life datasets.

6.1 Datasets

Two real-life datasets are used. They are introduced as follows:

- Chinese online forum dataset The first dataset contains all posts (and replies and comments) from a Chinese online forum, namely the Liba BBS \(^1\) from June 25 to July 25 2011. There are 1025 topics, each of which have a post and a series of comments and replies. 3528 users are involved. Conventional natural language processing methods are used for Chinese word segmentation. 6897 terms are extracted. Thus, in the tripartite graph, \( V_1 \) contains authors of posts or comments, vertices in \( V_2 \) are topics, while \( V_3 \) is for terms extracted.

- DBLP dataset The DBLP Bibliography dataset \(^2\) is used in experiments. 5534 papers with 8136 authors from three research areas, including database, data mining, and information retrieval, are used. 2018 terms from paper titles are used as terms. Similarly, in the tripartite graph, \( V_1 \) contains authors of papers, \( V_2 \) is the set of papers, while vertices in \( V_3 \) are terms extracted.

\(^1\)http://bbs.liba.com/.
\(^2\)http://www.informatik.uni-trier.de/~ley/db/.
Figure 7: Four steps for expert and expert profile query processing.

---

**Algorithm 1: Star graph construction**

**Input:** Tripartite graph \( G = (V_1, V_2, V_3, E_1, E_2) \)

**Output:** Star Graph \( G_s(V, E, W) \)

1. Initialize star graph \( G_s \) as empty;
2. for \( v_i \) in \( V_2 \) do
   3. \( \text{star}[i] = \) all paths of length 1 starting from \( v_i \) to vertices in \( V_1 \) and \( V_3 \);
4. end
5. Each star component \( S_i \) in \( \text{star}[i] \) is used as a vertex in \( G_s \);
6. for \( S_i \) in \( \text{star}[i] \) do
   7. for \( S_j \) in \( \text{star}[i] \) do
      8. if \( V(S_i) \cap V(S_j) \neq \emptyset \) then
         9. Add an edge to star graph \( G_s \) from \( S_i \) to \( S_j \);
      10. \( w(S_i \rightarrow S_j) = |V(S_i) \cap V(S_j)| \);
   11. end
6. end
7. end
8. Return star graph \( G_s \);

---

**Algorithm 2: Constructing query graph given a star graph**

**Input:** Star graph \( G_s(V, E, W) \), the query \( Q \)

**Output:** Query graph \( G_q(V, E, W) \)

1. Initialize star graph \( G_q \) as empty;
2. \( G_q = G_s \);
3. Add a new node \( Q \) to \( G_q \);
4. for \( S_i \) in \( \text{star}[i] \) do
   5. if \( V(S_i) \cap V(Q) \neq \emptyset \) then
      6. Add an edge to \( G_q \) from \( S_i \) to \( Q \);
      7. \( w(S_i \rightarrow Q) = |V(S_i) \cap V(Q)| \);
   8. end
5. end
6. Return query graph \( G_q \);

---

**Algorithm 3: Capturing experts \( E_t \) by RWR in graph \( G_q(V, E) \)**

**Input:** matrix \( W \) of \( G_q \), all star components \( \text{star}[i] \), query \( Q_e \), the number of experts \( k \), restarting probability \( c \)

**Output:** the experts \( E_t \)

1. Initialize \( \vec{r}_i = 0 \) except that the \( i \)-th element is 1;
2. Initialize \( \vec{p}_i = 0 \) except that the \( i \)-th element is 1;
3. Construct adjacent and transition matrix \( w = \text{colnorm}(M) \) of graph \( G_q \);
4. repeat
   5. \( \vec{r}_{i+1} = c \vec{r}_i + (1 - c) \vec{p}_i \);
6. passing 4 parameters \( r_i, Q_e, \text{star}[i] \) to Algorithm 4;
7. \( E_i = \) the output of Algorithm 4;
8. until not changes to \( E_i \);
9. Return \( E_i \);
6.2 Methods to be compared

We carry out 50 expert queries and 50 expert profile queries randomly. They are conducted over three approaches in two real datasets. We compare three methods based on different graph models.

- Bipartite graph model (BG) We construct a bipartite graph based on authors and terms, and implement the RWR algorithm on this bipartite graph to find experts and expert profiles. We call this approach as bipartite graph model, denoted as BG.

- Tripartite graph model (TG) We construct a tripartite graph based on users, topics (papers), and terms, and implement the RWR algorithm on this tripartite graph to find experts and expert profiles. We call this approach as tripartite graph model, denoted as TG.

- Star graph model (SG) We construct a star graph based on the tripartite graph, and implement the RWR algorithm on this star graph to find experts and expert profiles. We call this approach as star graph model, denoted as SG.

6.3 Measurements

We evaluate above three approaches on two datasets. Several measurements are used to evaluate those three methods.

- Efficiency Both time consumption and iteration times are used to measure the efficiency of three approaches.

- Effectiveness The effectiveness is only evaluated over the DBLP dataset. We did not evaluate it over the online forum dataset since there is no ground truth, and the evaluation is subjective. For the DBLP dataset, we use search results of ArnetMiner\(^3\), a service for academic data search, mining and visualization, as ground truth to evaluate the effectiveness of three approaches.

- Expert finding We use expert query results of BG, TG and SG, and respectively compute edit distance and overlap rate between the query results of ArnetMiner and query results of three approaches.

6.4 Experimental results

6.4.1 Efficiency

We respectively use three graph models to find expert and expert profile. Figure 8 and Figure 9 respectively show time consumption and iteration times for three graph models over the DBLP dataset, where time consumption and iteration times are averages over 50 randomly selected persons and 50 randomly selected terms. Figure 10 and Figure 11 respectively show time consumption and iteration times for three graph models over the online forum dataset.

From the four figures, we know our star graph model is more efficient than other two graph models. It is because our graph model reduces the size of the transition matrix. As well, the bipartite graph model is more efficient than tripartite graph model due to the size of the matrix. It shows that the matrix size dominates the efficiency of RWR.

6.4.2 Effectiveness on expert finding

Using ArnetMiner as ground truth, we evaluate the effectiveness based on edit distance and overlap rate between the query result of three approaches and the query result of ArnetMiner. It is noted that we

---

\(^3\)http://www.arnetminer.org.
Figure 9: Iteration times over the DBLP dataset.

Figure 10: Time consumption over the online forum dataset.

Figure 11: Iteration times over the online forum dataset.

Figure 12: Edit distance to top-\(k\) results returned by ArnetMiner.

Figure 13: Overlap rate to top-\(k\) results returned by ArnetMiner.

6.4.3 Effectiveness on expert profile finding

For the DBLP dataset, expert profiles of a person are his research areas. However, the expert profile finding result of ArnetMiner are not ranked, and only contains 3-4 research area. We list the results of three researchers returned by three graph models, as they are shown in Table 2. From this table, we know the result is similar of three graph models, and the result of ArnetMiner cover the results of three graph models.

7 Related work

Our work in this paper is broadly related to several areas. We review some in this section.

7.1 Expertise mining in social media.

In the past few years, experts and expert profile finding is a hot topic. Krisztian Balog [2, 1] discusses people search in the enterprise by a generative probabilistic modeling framework for capturing the expert finding and profiling tasks in a uniform way. Small-Blue [7, 3] mainly depends on social network among
company. It focuses on “who knows what?”, “who knows whom?” and “who knows what about whom?”

Recently, along with the growth of web 2.0 applications, more and more researchers are devoted to expertise finding problem in social media. Jun Zhang et al. [18] and Zhao Zhang et al. [19] studies the problem on online forums. The former work only considers reply networks in online forums, while the latter one only considers contents. Junjie Yao et al. [16] model users’ expertise in folksonomies of tagging systems. Xiaoling Liu et al. [8] studied the problem of identifying topic experts in the Blogspace. In this paper, our approach can handle all kinds of social media, and perfectly combine social networks with contents.

7.2 Random walk with restart and its improvement.

Faloutsos et al. treats RWR as a good means to score relevance between nodes in a graph [4]. Hang-hang Tong and others present several good applications using RWR [11, 12]. The issue of efficiency is great challenge of RWR [13]. Reference [14] proposed fast solutions to this problem. It uses low-rank matrix approximation and the community structure in graph to increase the query response of RWR.

8 Conclusions and future work

In this paper, we have addressed the problem of finding expert and expert profile in social media. Our work distinguishes with others in three aspects. First, a unified tripartite graph model is used to capture both content and structure information in social media. We show that a single random walk with restart procedure can be used to evaluate the relevance of a user and a term based on this graph model.

Second, a star-based optimization method is proposed to accelerate the RWR computation over tripartite graphs. Analysis show that this method can greatly reduce the online computation cost since it reduces the size of transition matrix.

Last but not the least, extensive experimental results over two real-life datasets show that our method outperforms previous bipartite graph model based method and the native tripartite graph model approach in terms of both effectiveness and efficiency.

Our future work include the exploration of data management techniques for star-based tripartite graph indexing that support RWR computation, and applications of expert and expert profile query in recommendation systems and online advertisement.

---

**References**


8. X. Liu, Y. Wang, Y. Li, and B. Shi. Identifying topic experts and topic communities in the blogspace. In Yu et al. [17], pages 68–77.


---

Table 2: Expert profiles of three researchers returned by different methods.

<table>
<thead>
<tr>
<th>Author</th>
<th>Approaches</th>
<th>First term</th>
<th>Second term</th>
<th>Third term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jiawei Han</td>
<td>SG</td>
<td>mining</td>
<td>data</td>
<td>pattern</td>
</tr>
<tr>
<td></td>
<td>TG</td>
<td>mining</td>
<td>data</td>
<td>using</td>
</tr>
<tr>
<td></td>
<td>BG</td>
<td>mining</td>
<td>data</td>
<td>using</td>
</tr>
<tr>
<td></td>
<td>ArnetMiner</td>
<td>data mining</td>
<td>efficient mining</td>
<td>spatial data mining</td>
</tr>
<tr>
<td>Philip S. Yu</td>
<td>SG</td>
<td>data</td>
<td>mining</td>
<td>clustering</td>
</tr>
<tr>
<td></td>
<td>TG</td>
<td>mining</td>
<td>classification</td>
<td>model</td>
</tr>
<tr>
<td></td>
<td>BG</td>
<td>mining</td>
<td>clustering</td>
<td>discovery</td>
</tr>
<tr>
<td></td>
<td>ArnetMiner</td>
<td>data mining</td>
<td>data streams</td>
<td>data mining techniques</td>
</tr>
<tr>
<td>Eamonn J. Keogh</td>
<td>SG</td>
<td>time series</td>
<td>finding</td>
<td>data mining</td>
</tr>
<tr>
<td></td>
<td>TG</td>
<td>time series</td>
<td>time</td>
<td>data</td>
</tr>
<tr>
<td></td>
<td>BG</td>
<td>time series</td>
<td>time</td>
<td>mining</td>
</tr>
<tr>
<td></td>
<td>ArnetMiner</td>
<td>time series</td>
<td>time series data</td>
<td>dynamic time warping</td>
</tr>
</tbody>
</table>


Combining Content and Quality Indicators in Ranking Ambiguous Query Results On Flickr

Hao Zhong          Ying Zhou

School of Information Technologies,  The University of Sydney
NSW 2006
Email: {hzho2468, ying.zhou}@sydney.edu.au

Abstract

When a user submits a text based query to content sharing sites like Flickr, a list of ranked results with limited refinement options are normally provided. Typical options would allow a user to rank the results in different ways such as relevancy, time, or quality respectively. The downside of such approach is that relevant results might not be of high quality while high quality results are often irrelevant. Possible ambiguity of query terms makes it even more difficult to get high quality and relevant results. In this paper we apply link based analysis to combine content and quality indicators for ranking query results in Flickr. Experiment show that our approach are able to identify high quality photos that match a query user’s intention and put them at the top of the list. The precision is better than original quality based ranking and possible query expansion results. Our approach relies on a set of seed users representing content and quality preference. We prove experimentally that the ranking is not sensitive to seed user selection, which makes it very practical.

Keywords: Social Networks, Link Analysis, Ambiguous Query, Flickr

1 Introduction

With the advance of various new web technologies, sharing content with friends and other people becomes a major part of the online social activity for many Internet users. The users are willing to and are able to publish all sorts of resources including many non-textual resources on social network based content sharing sites. Most non-textual resources such as images or videos are described by textual information such as titles or tags. Such textual information are often short and noisy by nature. Yet, they are the basis for text based queries for such resources, which poses lots of new challenges for the design of the query engine and the ranking algorithms. One of the most explored area is the disambiguation of query terms. Ambiguity of query terms is a very common problem in non-textual resources retrieval because the content range is huge while the textual information involved is limited and the vocabulary is uncontrolled. Terms that are not considered ambiguous otherwise may refers to several underlying concepts in those systems. Figure 1.(a) shows an example of query result for term “bear”. The result is sorted by “relevance”. In addition to animal bear we can see certain “surprising” bear related content such as bear claws, bear karte, bear-theme meal and a mountain named “bear peak”. It also includes several unrelated photos.
user generated judgment, together with other factors can be used to provide quality based ranking. For instance, Flickr provides rank by \textit{interestedness} option. Yet such quality based rank largely sacrifices content relevancy. Figure 1(b) shows the same \textit{bear} query ranked by \textit{interestedness}. Only one out of the top ten photos is related with the query term \textit{bear}.

Quality is a relatively subjective measure. One user’s favorite might not be another user’s favorite if they have very different tastes. In recommendation systems, a user’s taste is represented as a profile established by analyzing this user’s previous judgments. A group of users with similar profiles can be used to recommend items to each other. Apart from the implicit profile, explicit social network also serves as a good indicator of a user’s special tastes.

In this paper we apply link analysis on social network formed by users to re-rank text based photo query result in Flickr. The link analysis combines the content and quality indicators to provide relevant and high quality results matching a query user’s preference. Our work has the following contributions:

1. We conduct a study on social network formed by contact relations in Flickr and its impact on the quality indicators of photos. We observe that contact relation can reflect users’ shared preference in terms of content and quality.

2. We apply link analysis combining Personalized PageRank (PPR) and Weighted HITS to re-rank query result. Experiment shows that our approach achieves higher precision compared with original results and possible query expansion results. In particular, we show that using only a few seed users for a general category is sufficient enough to generate accurate results for all topics in that category and the result is not sensitive to seed user selection.

In section 2, we briefly describe flickr social network and quality factors involved. Section 3 describe the link analysis approach and its application in re-ranking flickr result. In section 4, we describe the experiments and analyze the results. We describe some related work in section 5. Section 6 concludes the paper.

2 Social network and a user’s preference on content and quality in flickr

Flickr is an online photo sharing website that allows users to upload, share, organize and view photos. Thousands of photos are uploaded every minute. Flickr users can form various relationships. A flickr user can assign other users as friends and give them permission to view photos that are not public. A flickr user can also assign other users as contacts. The contact relation creates a shortcut for one user to easily view all contacts’ public photos. Moreover, a user will be notified about any new photos published by her contacts. The friends network is relatively small and private, it is more of an access level control mechanism. The contact network is a public and forms a large social network where social ties off line. A flickr user can assign a user as friends. The contact relation is a way of showing interest on another user’s photo. Contact is a way relation. User A assigns user B as contact does not mean user A is also a contact of B(Lerman & Jones 2006). Lerman & Jones (2006) uses several data sets to study the social browsing behavior in flickr. It discovers strong correlation between a photos view number and it owner’s reverse contact number (the number of users assigned the owner as contact). It also discloses that a large proportion of a photo’s comments are from its owner’s reverse contact. This indicates that the social network formed by contact relation helps to reveal a user’s preference and viewing habit.

Flickr also provides mechanisms for users to explicitly specify their opinion or preferences on a particular photos. They can write comments on photos, display photos in their gallery and vote as favorite in photo. Favorite voting is the most commonly used rating feature in Flickr, the voting summary of a photo is public to the viewer and shown on the photo’s main page. Voting as favorite can be seen as a direct and positive feedback to the photo.

\textit{Gallery} is a relatively new feature, it can be seen as an advanced version of \textit{favorite}. A user can create a few galleries to feature her favorite photos. Galleries can reflect one’s preference more properly since they are more organizable. A photo can be included in multiple galleries of a user. In particular, favorite voting and galleries are independent. A photo can appear both in a user’s favorite list and galleries; or it can appear just in one place.

Commenting feature is one of the oldest interactive mechanisms used in many websites. It is also used very extensively by Flickr users. However, a user may leave positive, negative or off-topic comments on photos. Language processing is required to understand the nature of comments. Therefore, we will particularly look into the favorite voting and galleries in our research and use them as photo’s quality indicators.

Lerman & Jones (2006) observes that users do view and leave comments on their contacts’ photo. We argue that a user may share preference in terms of content and quality with her contacts. To assess the content similarity, we examine the meta data of photos uploaded by a user and by her contacts. To assess the quality preference, we examine if a user has a high proportion of photos as favorite or in gallery from her contacts’ uploads.

The content similarity is computed using the \textit{vector space model} (Salton et al. 1975) by mapping a user as a document consisting of tags from all her uploaded photos. We select 4 sets of target users differ in the number of contacts (around 50, 100, 200, 300 respectively). A target user and her contacts form the contact collection \textit{C}. We also select a set of random users (more likely to be “strangers”) for each target user to form a random collection \textit{R}. The size of \textit{R} is determined by the size of \textit{C}. We compare the average similarity between a user and her contacts with the average similarity between a user and random users. The result shows that a user is more “similar” with her contacts than with strangers (37.73% greater as shown in table 1), which means a user is more likely having shared preference with her contacts in terms of their uploaded photos. We do not see very high similarity values between user and her contacts because most users’ content preference are diverse and it is not likely for two users to match 100% in terms of content.

We take a sample of around 2000 users from flickr and collect their contacts, favorite photos and favorite photo owners. We compare a user’s contacts with the favorite photos of the user. Assigning a user as contact is a way of showing interest on another user’s photo. Contact is a way relation. User A assigns user B as contact does not mean user A is also a contact of B(Lerman & Jones 2006). Lerman & Jones (2006) uses several data sets to study the social browsing behavior in flickr. It discovers strong correlation between a photos view number and it owner’s reverse contact number (the number of users assigned the owner as contact). It also discloses that a large proportion of a photo’s comments are from its owner’s reverse contact. This indicates that the social network formed by contact relation helps to reveal a user’s preference and viewing habit.

Table 2 shows the
result. The first row shows the overall averages of the important measures. Depending on the total number of contacts a user has and the total number of favorite photos she has voted, the overlapping percentages are slightly different. We show the stratified averages in this table.

Table 2 clearly shows that a large proportion of a user’s favorite photos comes from her contacts. When a user has 100+ contacts, more than half of the favorite photos are from the contacts. There is also a large proportion of a user’s contacts has at least one photo being voted as favorite. When a user has 1000+ favorite photos, more than half of the contacts has a photo voted as favorite.

Gallery serves a similar purpose with favorite voting from the perspective of photo quality, we are interested to see if they overlap a lot. We take a sample of photos that have non-zero favorite counts and have at least been featured in one gallery. The photos are classified into 4 groups depending on their favorite counts. We obtain each photo’s gallery counts, gallery owners and favorite count and compute average values for each group. Table 3 shows the result. Although the gallery count increases with the favorite count, the large gap between the two indicates gallery featuring is not as popular as favorite voting. Discrepancy between gallery counts and gallery owners shows that a user may include a photo more than once in her galleries. We compute the number of overlapping users who both votes a photo as favorite and includes it in galleries. The result shows only 30.94% gallery users vote their photo as favorite, which means gallery and favorite features are relatively independent. Therefore, it is worthwhile to use gallery featuring as one of the quality indicators of photos.

These observations show that the contact network in flickr is a good indicator of shared taste in content and photo quality. In other words, it is safe to say that we can infer a user’s content and quality preference from its contacts. It can be used as a trust network in terms of content and quality.

<table>
<thead>
<tr>
<th>size</th>
<th>avg. sim in C</th>
<th>avg. sim in R</th>
<th>+%</th>
</tr>
</thead>
<tbody>
<tr>
<td>±50</td>
<td>0.341668</td>
<td>0.249671</td>
<td>36.84%</td>
</tr>
<tr>
<td>±100</td>
<td>0.2697</td>
<td>0.202808</td>
<td>33.17%</td>
</tr>
<tr>
<td>±200</td>
<td>0.379732</td>
<td>0.288658</td>
<td>31.55%</td>
</tr>
<tr>
<td>±300</td>
<td>0.319567</td>
<td>0.246634</td>
<td>29.57%</td>
</tr>
<tr>
<td>total avg.</td>
<td>0.32749</td>
<td>0.24674075</td>
<td>32.73%</td>
</tr>
</tbody>
</table>

3 Link Analysis and Query Result Ranking

Link based object ranking has long been used in the context of web information retrieval with two notable algorithms PageRank and HITS(Getoor & Diehl 2005). The query independent algorithm PageRank (Page et al. 1999) computes importance scores for all web pages. The scores can be biased towards a topic computed using a variation of the algorithm called Personalized Page Rank. In contrast, HITS(Hyperlink-Induced Topic Search)(Kleinberg 1999) computes a hub and authority score for each page in a broad topic query results. HITS often operates on a sub-graph constructed from the query result and is considered as query dependent algorithm. There are quite a few efforts trying to apply, to extend or to combine this two approaches for better and more resilient ranking(Zheng et al. 2009, Ding et al. 2002, Lempel & Moran 2000). In particular, Dell’Amico et al. Dell’Amico & Cioni (2008) propose SOFIA(Social Filtering Algorithm) to make social recommendation in a robust way. SOFIA operates on two networks: a social network among users and a judgment network between users and objects. It applies Personalized PageRank to quantify trust intention and use the trust intention to compute subjective HITS scores for recommending objects to a target user.

We adopt the idea of trust network and judgment network from SOFIA. Since the query user can be an anonymous Internet surfer or any registered user, it is not practical to provide fine tuned results for each individual user. We use pre-defined categories (such as animal, architecture) as preference indicators. For each category, a few seed users who have uploaded photos related with that category are selected as filters for it. We have proven experimentally in section 4 that general category together with query term can effectively pick high quality photos in many noises and the results are not sensitive to seed user selection.

3.1 Trust Network and PPR computation

Personalized PageRank (Page et al. 1999) expresses interest or topic preference as a set of pre-defined web pages call seed. In addition to following hyper links in a page, a random walker has a predefined probability (computed based on a given damping factor) to jump to any of the seed pages. The PPR can be applied in computing a particular user’s trust vector by setting this user as the seed. Mathematically, a vector $E$ is used to represent the seed. $E$ has a value 1 at the position corresponding to the seed user and 0 in all other positions.

For a trust network $G \in (V,E)$, let $O$ be the number of out-links (one’s contacts): $I$ be the number of in-links; for a user $u \in G$, $t_u$ represents the trust vector. The trust value a user $u$ imposes on a user $v \in G$ can be defined as below:

$$t_u(v) = (1 - d) \sum_{qeI_u} \frac{t_q}{O_q} + dE_u(v)$$

$d$ is a damping factor defined in (Page et al. 1999), setting to a value representing the probability of deviating from random walk along hyper links. The recommended value for $d$ is 0.15.

Not all users have contacts. A user with no contact becomes dangling nodes in the social network. Page et al. (1999) suggests to remove dangling pages before the computation. However, this approach is impractical to here because users without contacts are still deserved to possess a trust value as long as they are reachable from the seed user. Ng et al. (2001) and Lempel & Moran (2000) make another assumption that each dangling page should be a special page that points to all the pages; In the random surf model, this can be explained as the surfer has a equal chance to jump to any other page when he/she is about to stop at a dangling page. This assumption is convincing when dealing with the dangling pages in PageRank computation. Yet in terms of the trust propagation, we should not allow trust distributes to a random node.

Another approach to address the dangling problem is to create a “dummy” node(Bianchini et al. 2005). The dummy node is pointed by all the dangling node from the graph and has a self link.
This approach eliminates dangling nodes yet produces a heavy weighted dummy node. In fact, such a “dummy” node occupies a large proportion of ranking score in our PPR approach, because values from all dangling nodes aggregates at “dummy” node and the self link prevents them from being distributed again.

To overcome the dangling node issue properly, we simply combine the ideas from above: for each dangling node in our trust network, we create a back link to the seed node. This ensures the sum of trust scores on all nodes equals to 1.

### 3.2 Ranking based on Judgment Network

With any set of photos $P$, we can construct a bipartite graph with links from a user set $U$ to the photo set $P$. We add a link from user $u$ to photo $p$ if $u$ votes $p$ as favorite; adds $p$ in her galleries; or if $p$ is simply uploaded by $u$. It is easy to see that such link represents certain positive judgments from a user to a photo. We call the bipartite graph a judgment network. The weight of the link $w_{up}$ is computed by counting the number of judgments. For instance, if $u$ votes $p$ as favorite and add $p$ in two of her galleries, then $w_{up} = 3$. We aim to compute a rank score for $p$ using HITS algorithm. The computation is based on collective judgments from users and the trusts a query user placed on those users.

In our judgment network, users are pure authorities and photos are pure hubs. Therefore the authority values can be used as the ranking scores. (Lempel & Moran 2000) propose the idea of subjective HITS. (Dell’Amico & Capra 2008) further develop an algorithm to include trust value in the forward stage of computing hub value based on authority values. We adopt the similar idea with some modifications. For each user $u \in U$ and photo $p \in P$, the subjective HITS algorithm will generate the hub $H$ and authority $A$ iteratively through following 2 operations:

$$ H(u) = \sum_{p : w_{up} > 0} \sum_{v : w_{vp} > 0} \frac{T(v) \cdot w_{vp}}{T(u) \cdot w_{up}} \cdot A(p) \quad (2) $$

We start the computation by initializing the hub value to $|P|$ for each user. This ensures a total score of 1 to be divided among authorities and hubs in each forward and backward steps respectively. Equation 2 represents the backward step. For each photo $p$, we distribute the hub value to all pointing users based on their trust value. As this algorithm runs iteratively, the authority value of a photo will also be affected by the trust of all users who place positive judgments on it. Algorithm 1 gives the steps of computing hub and authority scores iteratively.

**Algorithm 1: trust weighted HITS**

**Input:** $G = (U, P, E)$: a judgment network, $T$: a trust vector

**Output:** $A$: Authority values for all $p \in P$; $H$: Hub values for all $u \in U$

Let $z$ denote the vector $(\frac{1}{|P|}, \frac{1}{|P|}, \cdots, \frac{1}{|P|}) \in R^{|P|}$

Initialize $A$ to $z$

while not converged do

  foreach $u$ in $U$ do

    apply Equation 2 to compute $H(u)$

  foreach $p$ in $P$ do

    apply Equation 1 to compute $A(p)$

return $H, A$

### 4 Experiment and Results

#### 4.1 Constructing social network

A list of typical ambiguous query terms have been used in many experiments in IR field(Cai et al. 2004, Goldberger et al. 2006, Liu et al. 2009). We are interested in not only individual terms, but also the top category those terms come from. One of the top categories where many ambiguous query terms...
come from is animal since animal names are used in various places. For instance, widely recognized ambiguous animal terms include tiger, which also refers to a baseball team (Detroit tiger), a golf player (tiger wood), a type of flower (tiger lily), apple operation system and so on; jaguar which also refers to a car model; Pluto which can refer to a planet; raptor which also refers to a type or aircraft. We choose a few top categories containing many ambiguous terms to start our experiment data collection. These include animal, flower, car and a few others. We then locate a few large groups in that category using flickr’s group search function. We download all photos from each group and use those as starting point to build a contact network by progressively crawl each level of contacts. We have obtained 72.9M contact links among 3.45M users, in which 300K users have at least 1 contact and over 120K of them have less than 50 contacts. These form the underlying social network.

4.2 Experiment design and metrics

We use raw Flickr query results as test collection. Each query may represent one or many underlying topics belonging to various top categories. We also select seed users for top or category by matching their preferences with the target category. For simplicity, each user’s preference is represented by top tags in her photo collection. We run experiments to test the personalization power of our approach and compare it with simple query expansion approach.

Any web search returns a large number of results. Most people care only the top few results. Precision at (top)k is a typical measure used in ranked query to evaluate the quality of results (Manning et al. 2008). We adopt this measure in our experiments. We set k=10 and 20 respectively. The relevancy is judged by human evaluators based on user preference and the photo content.

4.3 Results and Discussion

First, we run a set of experiments to see if users representing a general top category can effectively pick out photos in that category from an ambiguous query result. We choose two ambiguous terms jaguar and raptor to construct the test collections by querying flickr to obtain the top200 results ranked by relevancy and by interestedness. Both terms return results sets containing two major topics: jaguar animal vs. jaguar car and raptor animal vs. raptor aircraft. We select users representing animal, car and aircraft as seed to re-rank the results.

The ranking result for each query is shown in figure 2. Results from our approach are denoted as “personalized” while original flickr ranking are denoted as “original”. The personalized results show improvement in jaguar animal on precision@10 and raptor animal/aircraft on both precision@10 and @20. We can that in the original results, photos of animal category from both queries have high precisions(at least 0.7), which means they are the dominant topic for the respective ambiguous terms. Our approach performs well for less dominant topics(e.g. for jaguar car, the precision is increased from 0.2 to 0.6 at top10 and from 0.1 to 0.5 at top20). The performance of dominant topics is also promising (e.g. for jaguar animal, the precision is increased to 1 at top10 and for raptor animal the precisions are increased to 1 at top10 and top20). In certain cases, the popular topic’s personalized result might not be as good as the original one because our test collection has a fixed number of photos and there is little room for improvement.

When query results are not desirable, it is not uncommon for a user to expand the original query with additional keywords. Such simple expansion may help to filter out unrelated results. For example, the original bear query contains lots of noise including photos about flowers and landscape (as shown in figure1. A user interested only in bear animal may have a second attempt by expanding the query to specify its top category such as bear animal. However, simple query expansion does not always improve the quality of results. Flickr interestingness result for an expended query bear animal contains only 3 out of top10 relevant results.

The next set of experiments compare our approach with simple query expansion approach. It also examines the sensitivity of seed users selection. We first select two top categories animal and flower; for each category, we choose several topics such as tiger, lion, wolf, bear in animal and rose, lotus, pansy, peony in flower. For each topic, we run a single term query and an expanded query to obtain two sets of results from Flickr. Expanded queries are formed by adding the category name to the original term. Next we find...
six seed users for each general category. We ensure that seed users have common interest in the top category, while each may have preference in certain topics within that category.

![Image](animal.png)  
**Figure 3:** Precision comparison for category animal

We apply trust weighted HITS algorithms on single term query results for each seed user and compare the precision with that of original single-term and expanded query result. Figure 3 and 4 give the detailed precision comparison over the two general categories. For the personalized result, the figure shows the average precision of all 6 users with the error bar showing the maximum and minimum individual precision.

Single-term query result in general has the worst precision. Expanded query performs better than single term query in all cases. The average precision of all 6 seed users are higher than that of expanded query result, but use some seed users may not get better results than simple query expansion. For instance, in figure 3 the lowest precision@10 from personalized result for query lion is lower than the precision achieved in expanded query. This is more obvious in precision@20 chart.

An interesting observation is that the precision values of personalized results are not sensitive to seed users selection. The error bar shows the range of precision among all users. There are some big ranges in certain sub-topics. However, all seed users achieve much better precision than single query results; majority of the users achieve better precision than expanded query result. Such insensitivity is a good indicator for the practical value of our approach. It implies that we only need to precompile a few seed users for some top categories.

The variation of precision are caused by preference differences among the users representing the same top categories. If a seed user has preference on one or two particular sub-topic. The personalized rank based on that user would achieve the highest precision for queries of that particular sub-topic. From the animal category, we extract user 78****@N07 who prefers lion and user 80****@N03 who prefers tiger and plot the precision against those from original and expanded queries in Figure 5.(a). We can see the relatively high quality personalized results of user 78****@N07 and 80****@N03 gained at their special interests(0.8 and 0.93 respectively). For the flower category, we extract user 25******@N04 who prefers general flowers and user 40******@N08 who has a special interest in rose and any pink flowers and
plot the precision comparison in Figure 5.(a).

However, such biased preference towards a specialized topic does not prevent a seed user from being a good filter for other sub-topics. Sometimes the broad interests of a user’s contacts compensates the effect. In Figure 5 (a), it is clear that both users perform better than original and expanded queries across all four sub-topics. After evaluating the preferences of their contacts, we found that most contacts of user 80****@N03 are interested in animal, but not being restricted to tiger. Similarly, in category flower, none of the two users are particularly interested in topic lotus, but our approach can generate quality results for lotus query with precision above 0.7.

5 Related Work

We adopt the idea of trust and judgment network as presented in SOFIA(Dell’Amico & Capra 2008). However, SOFIA is focused on recommendation and the presence of seed user and its judgment in the judgment network is important for predicting judgment on other objects in the collection. We focus more on re-ranking query results based on content preference and quality. Our algorithm applies different initialization and weight distribution operations to achieve subjective HITS scores. It does not rely on a seed user’s judgment links. More importantly, the ranking is not sensitive to seed user selection hence it is very practical.

It is impossible to compute a PPR vector from scratch at query time. Jeh & Widom (2003) gives an efficient way of computing PPR vectors on the fly using precomputed partial vectors and hub skeleton. We adopt the idea in our implementation.

6 Conclusion

In this paper we apply link analysis methods to re-rank ambiguous query results based on seed users’ social contacts and quality indicators of non-textual resources. The heuristic behind our approach is that contact relationship between users can reflect their shared preference, which is confirmed by observations on sample data from Flickr. We further define the behavior of adding contacts in the social network as a way of expressing trust and quantify such trust via Personalized PageRank(PPR). The ranking is performed on a judgment network based on original query results and associated users. Particularly to Flickr data, we use three types of quality indicators to quantify the judgment between users and photos: ownership, favorite voting and gallery featuring. We apply a trust weighted HITS on such judgment net-
work to generate re-rank based on the preference of the seed user.

Experiments are carried out on Flickr datasets. We compare the precision against original and simple query expansion results. Our approach can generate better ranking based on a seed user and its social contacts. In particular, our approach only require a small set of seed users representing top categories, the experiment shows that such seed users are able to re-rank topics within the top category. The results are not sensitive to the selection of seed users in terms of the size of their social network and the judgment links they may have. Results may vary for different seed users but in general, the precision is higher than that of original and expanded query results. Although the algorithm and experiment are mainly based on Flickr network. The general idea presented can be used in other content sharing sites with an underlying social network.

References


Indexing RFID data using the VG-curve

Justin Terry  Bela Stantic  Abdul Sattar
Institute for Integrated and Intelligent Systems (IIIS)  Griffith University, Gold Coast, Queensland, Australia,  Email: J.Terry, B.Stantic, A.Sattar@griffith.edu.au

Abstract

Existing methods for the management of multidimensional data typically do not scale well with an increased number of dimensions or require the unsupported augmentation of the kernel. However, the use of multidimensional data continues to grow in modern database applications, specifically in spatio-temporal databases. These systems produce vast volumes of multidimensional data, and as such, data is stored in commercial RDBMS. Therefore, the efficient management of such multidimensional data is crucial. Despite it being applicable to any multidimensional vector data, we consider Radio Frequency Identifications (RFID) systems in this work. Due to RFID’s acceptance and rapid growth into new and complex applications, together with the fact that, as with commercial applications, its data is stored within commercial RDBMS, we have chosen RFID as a pertinent testbed. We show that its data can be represented as vectors in multidimensional space and that the VG-curve combined with Multidimensional Dynamic Clustering Primary Index, which can be integrated into commercial RDBMS, can be used to efficiently access such data. In an empirical study conducted on three, five and nine dimensional RFID data we show that the presented concept outperforms available off-the-shelf options with a fraction of the required space.

Keywords: Access Method, Multidimensional Data, Radio Frequency Identification - RFID

1 Introduction

There are many multidimensional indexes proposed in the literature, but very few have been adopted by the major database vendors due to their complexity and costs of integration. Many require unsupported access to the block manager so they cannot be readily constructed. Others are only possible as external indexes that do not inherit the industrial strength concurrency and recovery of the database system.

The vast majority of proposed multidimensional indexes require the unsupported augmentation of the kernel and are thus not readily available to support current RFID applications in commercial database systems. Currently to support and access multidimensional vector data in a commercial RDBMS it is possible to use bitmaps indexes, inbuilt R-Tree methods, approximation methods or one dimensional transformation methods. Bitmap indexes have been proposed for handling RFID data (Hu et al. 2005). Space requirements are reduced by using a bitmap data type that compactly represents a collection of identifiers. However, they have significant update costs and it may not work well when the data in the same cluster are not continuous or in applications that do not lend themselves well to grouping based on a common property (Lin et al. 2007).

Some commercial RDBMS has inbuilt R-Tree indexes, e.g., Oracle spatial, which has only recently supported intersections on three dimensional data. It is well known that the R-Tree’s performance deteriorates significantly above 4 dimensions. They suit a variety of objects and thus have overheads in complexity, as well as not supporting intersection queries beyond three dimensions (Murray 2005).

Approximation methods like the VA-file (Weber et al. 1998) and the IQ-tree (Berchtold et al. 2000) are based on the belief that above a certain dimensionality a full scan is more efficient than an index, so it is best to improve the scan. The IQ-tree integrates compression into its index based query processing, using a three-level index structure to combine a tree with a scan using quantization. The VA-file is a simple vector approximation method that uses an array of compact geometric approximations. Queries are answered by excluding most vectors through an approximate filtering step on the entire VA-file itself. The VA-file reduces the number of disk accesses, however it incurs higher computational cost in decoding the bit-string and computing bounds. Another problem with the VA-file is that it works well for uniform data, but not for skewed data (Jagadish et al. 2005), due to the pruning effect of the approximation vectors deteriorating.

One dimensional transformation methods utilize the available single dimensional index structures, e.g., B-Trees, to index the data based on a scalar key. They employ a two stage query filter, similar to spatial querying, to extract approximate results that then have any false hits removed to produce a final result set. Examples include The Pyramid-Tree technique (Berchtold et al. 1998), iMinMax (Ooi et al. 2000), The P+-tree (Zhang et al. 2004) and i-distance method (Jagadish et al. 2005).

Single dimension transformation methods are typically not bijective functions and are lossy in nature, thus they rely more heavily on the exact filter than bounded regions in the feature space. Their performance is affected by false hits occurring due to objects being far apart in the original space but close in the transformed space. For distance transformations such as i-Distance there may be many objects similarly close to a reference point, but it is unlikely they are all close together and many will need to be put through the exact filter.
Space Filling Curves (SFC) are bijective transformation functions that can produce a scalar key. The most well known SFC method is the UB-Tree (Berchtold et al. 1999), which integrates a SFC and a B-Tree creating a primary index for multidimensional data. It is an efficient paginated index where each leaf node represents a page of data on a segment of the curve. However, like other SFCs, the segments are typically not hyper-cubic and may even represent disjoint space, this typically increases the number of pages read that do not contribute to the answer and more importantly the UB-Tree requires changes to the kernel for integration so cannot be readily used as an internal index (only integrated in Transbase database).

One of the most prominent d dimensional point data structures is the K-D-Tree (Bentley 1975) and its variants: the hB-Tree (Lomet & Salzberg 1989), the BD-Tree (Ohsawa & Sakauchi 1983), the hybrid tree (Chakrabarti & Mehrotra 1999) and the quad-Tree. The K-D-Tree is a binary search tree that uses a recursive subdivision of the data space into partitions by means of d-dimensional hyperplanes. A disadvantage common to all K-D-Tree methods is that for certain distributions, no hyperplane can be found that divides the data objects evenly. Like the K-D-Tree, the quad-tree (Samet 1984) decomposes the universe by means of iso-oriented hyperplanes. An important difference however, is the fact that quad-trees are not binary trees anymore. The subspaces are decomposed until the number of objects in each partition is below a given threshold. Quad-trees are therefore not balanced and the subtrees of densely populated regions need to be deeper than sparsely populated regions, giving a bad worst case behavior. Disadvantages of space partitioning methods in general are that they can suffer from poor minimum node utilization, or have a high space complexity. Tree based space partitioning methods are typically unbalanced, increasing the worst case performance.

Some methods for efficient management of temporal data which can be incorporated within commercial database management systems have been presented in literature (Stantic, Topor, Terry & Sattar 2010), (Stantic, Terry, Topor & Sattar 2010). However, these methods cannot efficiently support high dimensional queries.

RFID data is naturally spatial and temporal in nature having time and location as two of its basic elements and additional attributes can be seen as additional dimensions. Therefore, it is well suited to being represented as vectors in multidimensional space where multidimensional queries as well as spatial and temporal predicates can be applied in a straight forward manner. Data warehouse applications will also need an efficient multidimensional access structure to support the growing demand for ad-hoc querying (M.Stonebraker & U.Cetintemel 2005, Stockinger et al. 2002). Typical multidimensional sample ad-hoc query could be: “Find containers that were picked up by fleet T for district D in the last 7 days and delivered by today”. In order to efficiently answer these kinds of queries an efficient access methods that scales well in volume and dimensions and is available within commercial RDBMS is crucial where these data records will be stored.

To be able to suit a variety of applications and data models we must consider medium to high dimensional data. The most commonly used access methods for medium to high dimensional vector data are access methods available off-the-shelf in commercial RDBMS (Rudolf Bayer and Volker Markl 1998).

RFID deployments are most commonly within a commercial relational database where the best database services such as industrial strength concurrency and recovery are available. Using off-the-shelf indexes to manage RFID data guarantees these services to the application. Thus we do not consider index methods that require kernel modification such as the UB-Tree method or access method that are known not to scale well with increasing dimensions like the R-Tree in Oracle’s Spatial Index.

In this work, we have built on top of the concept proposed in VG-Curve (Terry et al. 2011) and have shown that the VG-Curve combined with Multidimensional Dynamic Clustering Primary Index can efficiently manage and access RFID data. The proposed concept is not sensitive to the number or order of dimensions restricted in the query and is easily constructed within existing commercial database management systems and can efficiently manage the large volume of multidimensional RFID data. Storing the data in its multidimensional feature space allows processing of a wide variety of queries. The most important of these for spatial and temporal predicates is the multidimensional interval query that allows many spatial and temporal queries to be applied in a straight forward manner. In empirical evaluation, we demonstrate the performance of presented concept and show its superiority to the currently available off-the-shelf index methods for multidimensional RFID data.

The remainder of the paper is organized as follows; Firstly we review some key background information on RFID technology focusing on the tags and requirements of managing the data. In section 4 we describe our experimental study and in section 5 we analyze the results, finishing with a conclusion and further work in section 6.

2 Multidimensional Nature of RFID

Tagged objects moving through a RFID-based pervasive environment are automatically sensed and observed with their identifications, locations and movement paths. These observations are filtered and recorded in a database producing spatial, temporal and many additional dimensions (attributes) of data.

RFID, having time and location as two of its most basic elements, is naturally spatial and temporal in nature. Therefore, it is well suited to being represented in multidimensional space where spatial and temporal predicates can naturally be applied. Using a multidimensional access method that preserves the original feature distances allows many spatial and temporal predicates to be applied in a straight forward manner to the data without further data transformation.

Interval queries are an integral requirement for both spatial and temporal predicates in multidimensional data space. The efficiency of spatial and temporal queries relies on an efficient interval access method for multidimensional space. Thus any efficient multidimensional access method must be efficient at answering interval queries.

Fixed scanning devices have a known location and mobile scanners can be combined with global positioning to give accurate physical positioning of the location an object scan occurs. The spatial co-ordinates can then be stored and spatial predicates such as inside can then be run against the RFID data. This can be combined with the scanning time to answer spatio-temporal predicates such as inside x during T where T is the time interval.
Besides spatial and temporal attributes, RFID data can have many other attributes of interest. Current RFID systems with three to seven dimensions have been identified in the literature (Lin et al. 2007, Wang & Liu 2005) and the use of more complex application-specific tags requires even higher dimensionality. RFID applications will need an access method that can efficiently cope with data from a variety of dimensions to handle both simple and complex tag data as well as a higher level of data modeling such as those produced from data compression (Darcy et al. 2007).

The difficulties associated with multidimensional data grow with the number of dimensions. Once data has more than three or four dimensions, additional problems begin to arise which is loosely termed the ‘curse of dimensionality’ that can severely deteriorate an access method’s performance. Higher dimensionality characteristics include: exponential volume growth with additional dimensions, high probability of objects being near an edge, data sparsity, pages remaining unsplit in some dimensions and not cubic in shape and queries having very large extensions in each dimension (Berchtold et al. 1998, Bohn 2000, Weber et al. 1998). Due to these characteristics, the performance of traditional multidimensional access methods deteriorates rapidly as the dimensions increase (Orlandic & Yu 2002). Thus the majority of proposed access methods do not scale well to higher dimensions.

Same as for any spatio-temporal data the efficient management of RFID data is challenged by its large volume and multidimensional nature. The current and future needs of RFID applications will require an access method that is not only efficient, both in space complexity and in query performance, but is able to scale well with an increasing amount of volume and dimensionality. A reluctance to go above low dimensionality due to efficiency concerns may limit the design of RFID applications to low dimensional models, even when a higher dimensional data set has proven to be more effective and efficient (Lin et al. 2007).

3 Efficient Management of RFID Data

RFID data is generated quickly and automatically, and can be used for real-time monitoring or accumulated for object tracking. To filter and clean the high volume of real-time RFID data efficient methods are essential, especially for real-time applications.

RFID applications require a dynamic scalable access method that enables the spatial, temporal and additional dimensions of data to be queried in any combination and independently of each other. The use of multiple single dimensional or compound indexes to achieve this is not an efficient solution in terms of space complexity and update costs, though it is the most common solution to the problem.

3.1 The Primary Multidimensional Index

We build on top of VG-curve concept where variable regions that are formed are influenced by the distribution of data and blocking factor. It is important to mention that the concept does not enforce one-to-one relationship between regions and database blocks. However, it relies on the Primary index data being clustered on the cluster key, which in our case is unique region identification number - RegionID. To achieve the benefits that storing data in order gives to query efficiency, we have combined one dimensional transformations into clusters based on a SFC, with multi-stage query processing. Producing two structures, a primary index managed by the DBMS, that stores the data in order and a summary structure that manages the keys given to the data in form of directory.

The use of a linear ordering of dynamic (locally or defined) data clusters that are stored contiguously so that nearby clusters have a high probability of being in nearby blocks, i.e., a Multidimensional Dynamic Clustering Primary Index (MDCPI).

A Cluster is the group of similar (nearby) objects sharing the same cluster key (in our case RegionID) that will be stored contiguously in the base relation. We denote population pop as the number of objects located in the cluster. The cluster factor is the maximum number of objects allowed in a cluster before it is typically split into two clusters.

A Base Relation is an indexed relation altered and ordered by a cluster key (RegionID) plus a unique object key. It also contains one column for each dimension of the object and may contain other non-indexed attributes. This relation, typically a B+-Tree, is maintained and kept in order by the DBMS thus is sometimes known as an Index Organized Table. The base relation can be thought of as the access structure and leaf pages of the access method.

A Directory is a compressed representation of the base relation containing its RegionIDs and their population.

Control Processes are coded algorithms that manage and query the clusters including defining the order of clusters and hence data.

A MDCPI can retain the advantages of a primary access method while being able to exploit the particular characteristics of a variety of clustering schemes compared to a traditional clustering index.

The concept of MDCPI differs from existing transformation methods as it determines partitions at the local level which are managed by the directory using flexible user defined control processes as well as multistage query processing.

Three major benefits of using a MDCPI are:

- Efficient Interval queries: Interval queries benefit from the primary index organization as it is likely that several groups of sequentially stored clusters will be accessed to answer any interval query. These sequential clusters will be stored contiguously and can thus be efficiently retrieved by performing a range scan on the base relation for each sequence of clusters.

- Low Space complexity: the size of directory records is small, two fields, and the directory only stores the populated clusters. If the average population is 500, the directory will be less than 0.2% of the base relations size. We address the space complexity problems associated with block oriented (paginated) space partitioning methods as a cluster consumes only the physical space needed to store its objects, plus free space reserved for updating of the ordered base relation. This contrasts with block oriented methods that store each region’s data in a block regardless of block utilization.

- Simple Integration: The MDCPI is suitable for any relational or object-relational database system that offers an ordered relation and procedures. The access method is constructed with standard objects from off-the-shelf DBMS without the need for kernel modification, and as such, inherits database services such as industrial strength concurrency and recovery.
3.2 The Concept of Proposed Method

We present a space partitioning method that has common characteristics with Space Filling Curves (SFC). It suits a MDCPI as it forms a linearly ordered set of multidimensional clusters (regions). Previously proposed SFC methods employ a curve that passes through all points in multidimensional space, however, a Variable Granularity Curve (VG-curve) (Terry et al. 2011) connects regions of various granularity.

An additional benefit is that the space filling curve only connects regions populated with objects.

Though the partitioning is similar to quad-tree partitioning the VG-curve is not a paginated index and only one dimension at the time it is split, which is a considerable advantage as the number of dimensions grow. Splitting on additional dimensions to enhance the discriminatory power of the index is a characteristic we share with the TV-tree (Lin et al. 1994). Our method has three main differences to the TV-tree; we use a space partitioning strategy that creates an unbalanced virtual tree (the directory) representing the occupied data space, our method uses a multi stage filter process on separate data and directory structures and as it is not paginated it can be constructed without kernel modification.

Though the base relation stores the data in a balanced structure, the VG-Curve regions (in the directory) can be considered as an unbalanced binary tree where empty leaf and internal regions are not required to be stored. Not requiring internal nodes to be stored is a property that we share with the linear quad-tree (Gargantini 1982) but presented concept with VG-curve is not paginated but built using the MDCPI components.

Three major benefits of using variable size regions are:

- **Regions can efficiently partition the data:** The volume $r_v$ of a region decreases exponentially with its address length, $L$:

  $$r_v = v \times (2^{-L})$$

  allowing a fine partition with a relatively short address length.

- **Intersection calculations are simpler:** Intersecting regions will be fewer than intersecting points, particularly as it is only populated regions, and thus it is faster to calculate the intersecting regions compared to calculating the intersecting points of other SFC based methods.

- **The regions are hypercube like in shape:** Unlike all other SFC’s segments the VG-curve regions will have side lengths that differ at most by a factor of two making them hypercube or hyper-rectangular in shape. Having hypercube like shaped regions is widely recognized to produce the number of regions that overlap with a query interval improving efficiency for query processing.

Regions can hold up to the blocking factor (BF) number of objects. If the number of objects exceeds the BF the region is split along the next dimension as splitting is done in circular order of dimensions. The Algorithm assumes a fixed ordering for dimensions, but any ordering can be used, since the approach is almost non-sensitive to it (only the number of empty regions can slightly vary depending on such an ordering). An overfull region may require a child region to be split if the first split does not produce two under full regions (Terry et al. 2011). This occurs when the data objects are contained wholly within one child of the region. In this case, splitting of the overfull child will continue until two under full child regions are produced or until the pixel size region is reached. When a pixel becomes overfull it is not split and it’s population is allowed to grow beyond the clustering factor, similar to the concept of super-nodes for X-tree high-dimensional indexing (Berchtold et al. 1996). This is possible as the physical storage of a region is not limited to a page.

Multiple splits to partition a cluster do not increase the number of directory regions, as only populated regions are stored in the directory. When a split creates an empty child region, the density of the index increases as the sibling child has the same population as its parent in half the volume, thus the empty regions make a denser index.

The different SFC partitions derived from UB-Tree and VG-Curve methods are shown in Figures 1. They show the top half of a data space containing identical sets of data where each block contains 8 objects. The three UB-Tree areas in Figure 1 are (0,2,1,0),(1,1,2,1),(2,0,0,0). The VG-Curve regions in Figure 1 are derived using the above mentioned partitioning scheme which is in detail explained in (Terry et al. 2011).

Unlike the UB-Tree partitioning, our SFC partitioning scheme allows the identification and exploitation of empty areas to improve query processing. This can be seen in Figures 1 where the query interval (shown as a dotted rectangle) intersects three UB-Tree pages but only one VG-Curve page.

![Figure 1: Three (shades) pages using (a) UB-Tree partitioning and (b) VG-Curve partitioning, two of the nine VG-Curve regions contain two shades representing regions stored in two pages. The query interval shown as a dashed line.](image)

The percentage of data space that a region represents is not dependent on the dimensionality, but on the length of its address. Thus the space can be efficiently divided regardless of the dimensionality.

3.3 Query Processing Method

There are several query types of interest for point objects stored in multidimensional space. Relevant examples are Interval Queries (IQ) and Exact Match Queries (EMQ). If $p$ is a point in a $d$ dimensional space and $i_q$ is a $d$-dimensional query interval then the above queries can be represented as:

- **Interval Query (IQ)**

  - $p \in i_q$, find all objects that are contained within the query interval

- **Exact Match Query (EMQ)**

  - $p = i_q$, find all objects that have the same value as the target point for each $d$ dimension.

In this paper, we focus on the efficient processing of interval queries on medium dimensional point data ($d = 2-9$) as well as the exact match query, as it is a specific type of interval query. Multidimensional
range searching, such as interval queries, plays an important role in the way modern applications query their data. It covers many different query predicates in different data models (e.g., temporal, spatial etc).

Queries are performed in three stages, the directory is preprocessed to remove some regions that cannot contain answers then a primary (spatial) filter is used to select intersecting regions from the remaining regions and a secondary filter to remove false hits in regions overlapping the query interval (see Figure 2). Intersecting regions can be either Contained regions C or Overlapping regions O. Contained regions only have answer objects whereas overlapping regions will need to have their objects checked for false hits.

In the preprocessing of the directory, we identify the first and last directory region that can contribute to the result set. We then consider only these regions and the regions between them.

Similarly to spatial filtering, the intersecting query evaluation algorithm will return all intersecting data objects. As can be seen in Algorithm 1. In the primary filter all regions intersecting the query interval are returned. At first, all preprocessed regions from the directory are loaded into a cursor in a depth first order. Each region is then tested for intersection with the query interval without any I/O's on the directory. We have investigated tree is highly unbalanced or a query interval is large it may cause the CPU time to blowout, in a similar but less dramatic fashion then standard SFC methods. Therefore, in cases with a highly unbalanced tree or a large query interval, consulting the directory is more efficient.

4 Experiment

In order to empirically prove the efficiency of the method presented in this paper on medium dimensional data (3-9 dimensions), we have conducted an extensive empirical evaluation against the most common indexes used to support medium to high dimensional vector data ( Rudolf Bayer and Volker Markl 1998). Our aim is to follow commercial RFID applications and efficiently manage RFID data within a commercial relational database where the best database services are available.

In line with the study performed to evaluate the efficiency of the UB-Tree (Rudolf Bayer and Volker Markl 1998) and VG-Curve (Terry et al. 2011), we compare with the best available methods in off-the-shelf commercial RDBMS for medium to high dimensional data, i.e., multiple secondary indexes, compound indexes and table scans. As it has been shown that even for six dimensions a compound index outperforms secondary indexes when the result set is greater than 0.000015 % of the relations population (Rudolf Bayer and Volker Markl 1998).

Unfortunately, we could not compare with the UB-Tree method, as it requires kernel modification. Also, we did not compare with access methods that are known not to scale well with increasing number of dimensions like the R-Tree in Oracle’s Spatial Index. The performance of other SFC methods (e.g., Z-curve) deteriorate rapidly above a few dimensions and as the query interval grows due to a blow out in CPU operations, which we confirmed in initial testing, thus they were found to be unsuitable for this experiment.

Currently, the most widely used technique to handle multidimensional interval queries is the use of a
secondary index for each dimension (Rudolf Bayer and Volker Markl 1998). The performance of multiple secondary indexes however deteriorates rapidly as the dimensions grow and is only useful for very small result sets, e.g., in (Rudolf Bayer and Volker Markl 1998) for six dimensions a compound index outperforms secondary indexes when the result set is greater than 0.000015 % of the relation’s population. Multiple secondary indexes have an additive behavior whereas the VG-curve has a multiplicative behavior. Assume $N$ objects in the universe and that $p_i$ % of objects lay in the query intervals restriction of the $i^{th}$ dimension. Then additive behavior means we need to fetch $N \times \prod_{i=1}^{d} p_i$ % objects or object identifiers for the $i^{th}$ dimension. Requiring in total $\sum_{i=1}^{d} N \times p_i$ % objects to be fetched.

For the VG-curve the amount of data to be fetched is approximately proportional to the interval query result set i.e., $N \times \prod_{i=1}^{d} p_i$ %. As we are focussing on interval queries at medium to high dimensionality and as we do not wish to restrict the result sets to very small we have focussed on comparisons with the compound index and table scan.

To efficiently index multidimensional RFID data with compound indexes requires the full set of combinations of compound indexes covering from 2 dimensions up to the dimensionality of the data. This set of compound indexes can then efficiently answer a query restricting any combination of dimensions. The number of compound indexes required for a data set is given by:

$$\sum_{r=2}^{d} \left(\frac{d!}{(d-r)!r!}\right)$$

where $d$ = the number of dimensions of data. For three dimensions, this means four indexes i.e., ((D1,D2), (D2,D3), (D1,D3) and (D1,D2,D3)). For five dimensions twenty-six compound indexes are needed and for nine dimensions five hundred and two compound indexes are required. This explosion of combinations of required indexes is in line with the curse of dimensionality hypothesis that renders most indexes inefficient due to the exponential space and complexity that each additional dimension contributes (Bohm 2000).

4.1 Data Set

We have tested the performance of VG-Curve on RFID data with three different dimensions:

For three dimensional Raw RFID data we have considered the following schema: (EPC, ScanTime, LocationID). Data represents a chain of 50 stores where every store has 20 scanners monitoring 20 product lines with 500 items in each line.

For 5 dimensional data, we used the container-locations data from Lin et al. (2007), with data on five fleets of 20 trucks servicing 10 districts containing 100 stores over a 3 month period. Five dimensional data has the following schema: (EPC, LocationID, TimeStart, TimeEnd, TransportMeansID)

The application specific 9 dimensional scenario proposed is for a Shipping Consortium running an international container shipping application monitoring the movements of containers in the past 100 days across 25 nations with 40 ports and each serviced by 4 fleets of 25 ships using 1000 container terminals with 20 readers in each. Containers are ordered by 20,000 importers from 10,000 exporters. The following schema has been considered: (EPC, ScanTime, LocationID, ShipID, DepartPortID, ArrivePortID, ExporterID, ImporterID, ShipDate)

All VG-curve relations used a blocking factor of 1,000 and a max depth of 85.

4.2 Query Set

One dimensional queries like the tracking query: “Given the EPC find the history of object” can be efficiently answered with a standard one dimensional index so are not considered. However, two dimensional queries such as: “Find all objects at a certain location at a certain time” can not be efficiently answered with a one dimensional index.

We have tested queries where between two and all dimensions are restricted to demonstrate the performance of our multidimensional access method as a replacement for all combinations of compound indexes for the given relation.

For ease of comparison, all queries aim to return approximately 5000 rows. We compare against the ideal compound indexes for each query i.e., compound indexes who’s indexed dimensions match the query restrictions. All query subgroup ranges start from a randomly selected identity number. To be able to achieve the a consistent result set size from queries above six dimensions we simply used the a query of the same percentage restriction in each dimension. Due to space constraints we can describe only a few example queries from each of the five and nine dimensional sets.

For the 3 Dimensional relation, we used the following queries:

Q1. Find all items in Product Line P scanned at Store S.
Q2. Find all items in Product Line P scanned at Store S for the month M.

Example queries for 5 dimensional relation:

Q1. Find containers that were picked up by a contaminated truck T in the last 9 days.
Q2. Find containers that were picked up by fleet F for district D in the last 7 days and delivered by today.

Example queries for 9 dimensional relation:

Q1. Find containers ordered by an exporter E shipped during the past 50 days.
Q2. Find containers moved from nations N by fleet F shipped more than a month ago arriving at a container terminal before the past fortnight.

Each query has its restrictions randomly instantiated 20 times and average performance figures are reported.

4.3 Environment

All experimental results presented in this Section are computed on a Sun Fire V880 server with 8 x UltraSPARC-III 900MHZ CPU using 8GB RAM, running Oracle 10g RDBMS. Database block size was 8K and SGA size was 1GB. At the time of testing database server had no other significant load. We used built-in methods for statistics collection, analytic SQL functions, and the PL/SQL procedural runtime environment. All queries had the buffers flushed before running.

5 Result and Analysis

As anticipated, our results show that the VG-curve combined with MDCPI concept is an effective replacement for multidimensional combinations of compound indexes for a given data set at a fraction of the space. Compared to the best compound index for each query our method is clearly superior for 3 and 5 dimensions with on average reductions in CPU and I/O, as can
Table 1: Average percentage decrease (negative) or increases on 3, 5 and 9 dimensional data sets of the VG-curve compared with the ideal compound index for each query.

<table>
<thead>
<tr>
<th>VG C vs Ideal Comp</th>
<th>3D</th>
<th>5D</th>
<th>9D</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O (%)</td>
<td>-86.7</td>
<td>-86.0</td>
<td>-39.9</td>
</tr>
<tr>
<td>CPU (%)</td>
<td>-11.9</td>
<td>-9.8</td>
<td>298.4</td>
</tr>
</tbody>
</table>

be seen in Table 1. In this table best compound index value is considered 100% and therefore -86.7% for I/O means our concept required 86.7% less I/O’s that the best compound index. According to the theory of indexability a physical disk accesses are considered to be the most important aspect to be taken into account (Hellerstein et al. 1997). For nine dimensions our method requires less disk I/O’s however due to the query algorithm requires more CPU, which is due to the increased overlap at the high dimensions. However, it is important to mention that our method can basically efficiently answer any combination of restricted dimensions. In contrast using compound indexes to be able to answer any combination of restricted dimensions according to the equation 1 would require 502 different compound indexes for nine dimensional data. It is also important to highlight that the compound indexes in our experiment are constructed to have the same dimensions as the query of the same dimensionality, i.e., they are the best compound index for that query. To support this claim, it is important to highlight that in experiments for different queries we used best possible compound index to efficiently answer specific query and in all cases compound index was different, however, for all queries same VG-curve structure was used built on top of MDCPI concept.

Even for just 3 dimensions adding the compound indexes needed increased the space used by 370%, by 9 dimensions they increase the space needed by 315 times. As can be seen in Figure 3 our method’s base relation achieved the storage utilization guarantee of the underlying B*-Tree.

Figure 4: CPU usage for three dimensions

We show the performance comparison in both CPU and I/O’s for three dimensions in Figures 4 and 5, in five dimensions in Figures 6 and 7 and in nine dimensions in Figures 8 and 9. The number of dimensions restricted in each query is shown on the legend in brackets next to the query number. Queries are displayed in the charts in the same order as they appear in the legend.

Figure 5: Physical Disk I/O’s for three dimensional data

Figures 5 and 7 show our method to be superior for all multidimensional queries on three and five dimensions compared to the best of the compound indexes for each query.

Figure 6: CPU usage for five dimensional data

Figure 7: Physical Disk I/O’s for five dimensional data

Despite the fact that we build the best possible compound indexes for each query, out of 502 possible for 9 dimensional relations, the physical disk I/O of our method (Figure 9) is superior for 5 of the 6 multidimensional queries. The exception is the 2 dimensional index on the 2 dimensional query on the 9 dimensional data set. The I/O performance of our method demonstrates its suitability for medium to high dimensional data by showing its scalability over several dimensions without the usual severe performance deterioration.

Limiting the combinations of compound indexes to a subset due to space considerations runs the risk
of poor performance as can be seen in the results for all relations and especially Figure 8 for the 2 and 4 dimensional compound index, where queries restricting dimensions not matching the indexes can perform very poorly.

In summary, the presented concept differs from the K-D-Tree since in the VG-Curve approach partitions are at predetermined positions and it is a secondary (disk - oriented) not primary memory storage method. It differs from Quad trees since the VG-Curve dimensions are split one at a time. It differs from the grid file since the partitions are applied locally to the node not across the whole dataspace. It differs from the UB tree since regions are hyper cubic, or hyper rectangular with two side lengths of x and 2x. It differs from other SFCs since it uses a directory and a two stage query processing. We differ from the curve dimensions are split one at a time. It differs from Quad trees since the VG-Curve method combined the Multidimensional Dynamic Clustering Primary Index is superior to the off-the-shelf compound index, which is tailored for the specific query, for processing range queries on RFID data of medium dimensionality.

We have shown that multidimensional data can be organized in a way suitable for employing a primary index structure which guarantees better performance.

Through an empirical study, we have demonstrated that the VG-curve method combined with the Multidimensional Dynamic Clustering Primary Index is superior to the off-the-shelf compound index, which is tailored for the specific query, for processing range queries on RFID data of medium dimensionality.

In future work we intend to investigate the applicability and performance of the VG-curve method on a wider variety of query types and data sets while identifying how to best exploit the parameter blocking factor to suit different data characteristics and local conditions.

References


Zhang, R., Ooi, B. C. & Tan, K.-L. (2004), Making the Pyramid Technique Robust to Query Types and Workloads, in ‘ICDE ’04: Proceedings of the 20th International Conference on Data Engineering’.
TIP Spatial Index: Efficient Access to Digital Libraries in a Context-Aware Mobile System

Wendy Osborn\textsuperscript{1}  Annika Hinze\textsuperscript{2}

\textsuperscript{1}Department of Mathematics and Computer Science, University of Lethbridge
4401 University Drive West, Lethbridge, Alberta, T1K 3M4, Canada
Email: wendy.osborn@uleth.ca

\textsuperscript{2}Department of Computer Science, University of Waikato, Private Bag 3105, Hamilton 3240, New Zealand
Email: hinze@cs.waikato.ac.nz

Abstract

We present a framework for efficient, uniform, location-based access to digital library collections that are external to a context-aware mobile information system. Using a tourist Information system, we utilize a spatial index to manage the context of location. We show how access to resources from within and outside of the tourist information system can be carried out in a seamless manner. We show how the spatial index can be navigated to continually provide information to the user. An empirical evaluation of the navigation strategy versus traditional spatial searching shows that navigation is efficient and outperforms traditional spatial search. In conclusion, our work provides a strategy for context-aware mobile systems to co-operate with digital libraries in a seamless and efficient manner.

Keywords: context-aware mobile systems, location-based, spatial indexing, digital libraries

1 Introduction

A mobile tourist information system provides information to a traveller based on several factors, including location. In addition to providing the user with information managed by the system itself, information that is available from local, external repositories should be made available to the user. In particular, digital libraries contain a wealth of information for the user to access. The overall goal of our information system is as follows: While travelling with a mobile device such as an iPod, the user is continually presented with information from not only the mobile information system, but also from digital libraries, based on their location and their user profile.

To accomplish this requires efficient location-based organization and retrieval of information that is both internal and external to a mobile tourist information system. Two issues arise around the location-based indexing of information: 1) There exists many municipalities in the world, each of which has its own co-ordinates. (For example, World Gazetteer (http://www.world-gazetteer.com/ contains over 300,000 co-ordinates.) 2) in addition to existing municipalities, a use may want information from a location that is not specific to a municipality (e.g. an iceberg that has migrated to a location off the coast of New Zealand). Given both sources of co-ordinates, there is the potential of having to index close to a million co-ordinates. Therefore, information must be organized by its location in such a way that efficient information retrieval based on the user’s current location is possible.

In this paper, we present an efficient location-based index structure. We show the use of of index when accessing digital libraries that are external to a context-based mobile system. Our primary goals are twofold. The first is to provide a framework for efficient and uniform access from a context-aware mobile information system to resources contained externally in digital libraries, as well as resources within the mobile information system itself. This is accomplished by utilizing a spatial index. The second is to propose an algorithm for efficiently navigating the spatial index to continually provide the user with information whenever it exists. An empirical evaluation of the navigation algorithm versus traditional spatial searching shows that the navigation algorithm is efficient and outperforms repeated spatial searching.

Initially, we focus on the location context in this work. We also focus on external resource access from a mobile tourist information system. However, it is expected that a self-contained module will evolve from this work that could be utilized by any mobile information system to access external resources. We also note that, although we present some visualization in order to provide context for the paper, the presentation/visualization of any resources that are fetched by TIP is outside the scope of the work described in this paper.

The remainder of this paper is structured as follows. Section 2 provides a brief summary on TIP (Tourist Information Provider), digital libraries and the mqr-tree spatial index, all of which are required
for our work. Section 3 summarizes related work and its limitations. In addition, this section will present our contributions. Section 4 presents our first user scenario that demonstrates uniform access to both internal and external resources. Section 5 presents the required modifications to the TIP architecture, while Section 6 presents the TIP tree, its organization of internal and external resources, and the algorithm for continuous navigation of the index. Section 7 presents our second scenario that demonstrates how the TIP tree is continuously navigated in order to provide updated information to the user. Section 8 presents our empirical evaluation of the TIP tree. Finally, Section 9 concludes the paper and discusses outstanding issues.

2 Background

This section presents a brief introduction on TIP, digital libraries, spatial indexing and the mqr-tree, all of which are required for this work. TIP (Hinze et al. 2009) provides the user with information based on their contexts such as current location, interests, and travel history. It is organized by separating the core system logic and TIP data from the various services that interact with it. Therefore, this architecture would allow us to: 1) modify how the location context is handled, and 2) extend it to provide access to external resources. These features are the reasons why we chose TIP for our work.

A digital library (Witten et al. 2009) is a collection of documents that are digitized, individually catalogued, and organized for online dissemination and access. Digital library software systems provide the means for searching and browsing. Some digital libraries can be searched and browsed using different metadata. Many digital library software systems exist, including Greenstone (New Zealand Digital Library Project 2008, Witten et al. 2009), DSpace (Tansley et al. 2006) and Fedora (Lagoze et al. 2006).

Spatial indexing (Shekhar & Chawla 2003) provides an efficient mechanism for accessing data using location, such as longitude and latitude co-ordinates. Many spatial indices have been proposed in the literature (see (Gaede & Günther 1998, Shekhar & Chawla 2003) for surveys). A spatial index that has been proposed recently, the mqr-tree (Moreau et al. 2009), which organizes point and object data using the inherent spatial relationships that exist between them (e.g. one object is northeast of another). This provides support for efficient navigation within the spatial index itself. In addition, performance comparisons versus other spatial indices (Moreau et al. 2009) shows that the mqr-tree supports a one-path search when indexing point data such as co-ordinates – a feature that most other spatial indices cannot achieve. For these reasons we chose to modify the mqr-tree for our work.

3 Related Work

Many mobile tourist information systems have been proposed, including AccesSights (Klante et al. 2004), CATIS (Pashtan et al. 2003), CRUM-PET (Poslad et al. 2001), Cyber-Guide (Abowd et al. 1997), Guide (Cheverst et al. 2002), Gulliver’s Genie (O’Hare & O’Grady 2003) and TIP (Hinze et al. 2009). Most of these systems focus on providing structured information and recommendations. Additionally, most of these systems only provide the user with information that is directly maintained by the system itself. They do not provide information from sources that are maintained in external sources, such as digital libraries.

The first system to be proposed that integrates a mobile tourist information system with a external digital library is TIP. The TIP/Greenstone Interaction Sequence

Figure 1: TIP/Greenstone Interaction Sequence
Bridge (Hinze et al. 2006) provides the functionality for TIP to connect to and retrieve information from a Greenstone digital library (Witten et al. 2009). In addition to the information that is provided by TIP for a specific location, a user can obtain further information on items and features in a Greenstone digital library that are related to the same location. Some limitations of the TIP/Greenstone bridge include the following: 1) it only retrieves external documents from Greenstone collections, 2) it only works with geographically aware digital libraries (i.e., place names must be identified and marked up), 3) TIP locations are defined by place names, while a mobile device defines location using GPS co-ordinates, extra steps are required to obtain a place name and have the user select an appropriate one before the user is able to access a Greenstone collection, and 4) information needed to access internal TIP information and external Greenstone information are kept separate, and cannot be presented to the user for consideration at the same time.

Contributions of the paper. Based on our analysis of related work, this project provides the following contributions to improve upon the limitations of existing systems:

1. Incorporating a spatial index into TIP for managing access to both internal and external resources. We will organize known sources of information in a spatial index by their GPS co-ordinate location instead of their place name.

2. Access to any digital library collection. Our system is not restricted to only accessing digital library collections managed by Greenstone. Links to collections managed by other digital library software systems such as DSpace (Tansley et al. 2006) and Fedora (Lagoze et al. 2006) can be added to the index as well.

3. One-pass access to relevant collections. Because GPS co-ordinates can be obtained directly from a mobile device, and the information sources are organized by their GPS co-ordinates, we can directly access both internal and external sources directly by directly accessing the spatial index.

4. No requirement of a “geographical aware digital library”. In order for the TIP/Greenstone Bridge to work, a collection needed to be preprocessed to identify place names (i.e., locations) in its documents. This is no longer a strict requirement. Any resource (TIP or external) can now be added by specifying their GPS location. The GPS location can be obtained by the user from a World Gazetteer. However, place name mark-up provides an additional advantage. It allows for place names to be selected (as a hyperlink) and further index accesses to take place, although an extra translation step to obtain GPS co-ordinates is required. But it will not be required in all situations when the index is accessed.

5. Handling collections that reference multiple locations. If a digital library collection contains documents that mention multiple locations, the collection can be added to the index multiple times, once per every location that is mentioned.

6. Navigating the index for continuous information dissemination to the user. We base the TIP index on an efficient spatial index, which could be searched from the beginning every time a user moves along with their mobile device. However, we feel this is unnecessary. The spatial index can also be efficiently navigated to continually provide information to the user, and thus extra searching is avoided.

4 Usage Scenario 1 – Retrieval of Resources

In this section we demonstrate how TIP can be utilized to access both internal and external resources. Here, our user arrives in Hamilton, NZ, and wants to access information on the city using TIP. In addition, our user finds links to other locations and is interested in looking into these further. Figure 1 shows the sequence of interactions that will take place.

First, TIP determines the user’s current GPS co-ordinates from their mobile device. Using these co-ordinates, TIP searches its index to identify all resources – both internal and external to TIP – that are near the user’s current location. TIP presents three links to the user – the first is information on Founders Theatre that is maintained by TIP itself, the second is a link to a Greenstone collection on plants and gardens, and the third is a DSpace collection on the history of the Waikato region. Our user is interested in gardens, and decides to follow the link to the Greenstone collection. TIP uses the Greenstone service (i.e., TIP/Greenstone Bridge) to obtain documents from the Plant&Garden collection.

Next, the user follows the link for the Hamilton Chinese Garden document. In addition to information on the garden itself, there is a link to the location of Wellington. Our user plans to visit Wellington in the near future, and decides to click on the place name for more information. TIP obtains the GPS co-ordinates that corresponds to Wellington and searches the index for all internal and external resources that it has on the location of Wellington and displays these to the user for further follow up. This is depicted at the end of Figure 1.

5 Modified TIP System Architecture

In this section we present our modifications to the existing TIP architecture to facilitate location-based access to TIP and external resources.

Figure 2 shows the conceptual design of TIP with the required extensions. In particular, the following components need to be added: 1) a spatial index that organizes TIP and external resources using their location co-ordinates, and 2) services to handle the retrieval of collections and documents managed by different digital library software systems. Our extended architectures shows services to handle retrievals from external Greenstone, DSpace and Fedora digital libraries. TIP already has a Greenstone service that we can use here (Hinze et al. 2006). In addition, this architecture can be easily extended to provide access to digital library collections managed by other software systems by adding the appropriate retrieval services.

6 TIP Tree: Spatial Index with navigation

In this section we present the TIP tree, which extends with mpr-tree to handle navigation of the location context. The goals of the TIP tree are:

1. To provide location-based access to information from both TIP and external digital libraries in a uniform manner;
To provide efficient navigation of the location context so that up-to-date information is provided continuously to the user.

Each node in the TIP tree represents a bounded region in space. Each bounded region can contain a co-ordinate for a particular location and/or other bounded regions covered by child nodes. For lower-level and leaf nodes, these regions will be smaller than those represented by higher-level nodes, with the root node encompassing all regions covered by all lower-level nodes. Each co-ordinate in the TIP tree represents a location for which TIP information and/or collections in external digital libraries exists. Therefore, a leaf node stores information on each item, including its name, whether it is internal or external to TIP, and if it is external, the name of the digital library that manages it and instructions on how it is accessed (i.e. usually via a URL).

Figures 3 and 4 depict an example of the TIP tree. Here, the TIP tree is indexing resources from different locations in New Zealand. For example, the co-ordinate representing Auckland provides access to resources on All Blacks games (located in TIP), a Greenstone digital library on the Auckland Museum (external to TIP), and a DSpace collection containing documents from the University of Auckland (also external to TIP).

### 6.1 Tree Construction and Updating

The TIP tree is constructed in the following way. For each resource to be added (i.e. either TIP information or an external digital library) and corresponding co-ordinate\(^2\) that indicates its location, a search is performed to see if the co-ordinate already exists in the TIP tree. If so, then the resource is added to the corresponding list. If not, then a new point representing the co-ordinate is inserted into the tree, and a new leaf node is added that contains the resource.

Updates to a resource in the TIP-tree can be performed by deleting the existing resource record, and re-inserting it with the updated information and/or location co-ordinate. Updates to a location co-ordinate can be performed by first preserving the existing leaf node it is linked to, then deleting the existing co-ordinate, inserting a new co-ordinate, and linking the new co-ordinate to the existing leaf node. The insertion and updating algorithm used by TIP is adapted from the mqr-tree – more information on the details of insertion can be found in (Moreau et al. 2009).

Note that although the example index (Figure 4) organizes resources based on its actual physical location, it is possible to also add a collection based on a location being mentioned in a collection. For example, if a collection at the University of Auckland mentions Hamilton, this collection can be added to the corresponding list for Hamilton as well by specifying the co-ordinate that represents Hamilton.

### 6.2 Navigation

Figure 5 shows the pseudocode for navigating the TIP tree. Navigation begins by performing an initial search to find a starting point in the index. This puts the search point within a bounded region of space that is covered by (i.e. represented with) a node in the tree.

Within the bounded region, the search point will be located in a particular quadrant (NW, NE, SE, SW). If a co-ordinate exists in the current quadrant, the corresponding list of digital library collections and TIP information will be retrieved and sent to the user, along with the distance the user is from the co-ordinate. Otherwise, nothing is displayed back to the user - if no co-ordinate exists at this location, then TIP currently had no information containing to this subregion, and therefore has nothing further to present the user.

The user will then move, which causes the coordinates of the search point to be updated. This will result in one of the following situations:

1. The search point remains in the current quadrant. In this situation, nothing changes other

---

\(^2\)obtained from a world gazetteer
Figure 3: The TIP spatial index

Figure 4: The TIP Tree
than updating the distance from the existing coordinate in the quadrant, if any. Any digital library collections and TIP information that is currently fetched for the user is still available to them.

2. The search point moves to a different quadrant. If a co-ordinate exists, the corresponding collection and information are displayed to the user. If the quadrant references a child node and the search point is within the bounded region of the child node, then the search point is sent to the child node. Otherwise, nothing occurs until the search point is updated again - because no collections or information is indexed at this point, nothing is presented to the user.

3. The search point falls outside of the current bounded region. In this situation, the search point is sent to the parent node and re-evaluated using the above criteria.

This process repeats until the search point falls outside of the space covered by the index, at which point navigation is terminated.

7 Usage Scenario 2 – Navigation

We present an example that demonstrates TIP tree navigation. Here, the TIP user will arrive at Auckland Airport and make their way to Wellington. Along the way, TIP will notify our user of any nearby information of interest. Figure 6 shows the locations (location indicated by coloured points, referenced by numbers) where TIP will display information to the user, and the corresponding locations (using the same coloured points and numbers) in the TIP tree.

The user will begin receiving information from the TIP tree on arrival at Auckland Airport (Latitude -37.00806, Longitude 174.79167, 1 pink point). An initial search of the index will set the search point in the bounded region and quadrant (NW) containing the co-ordinate for Auckland. Corresponding to the co-ordinate for Auckland are the digital library collection for the Auckland Museum, a digital library collection from the University of Auckland, and information on the upcoming All-Blacks rugby game at Eden Park. The first two collections are managed by Greenstone and DSpace respectively, and both reside on servers that are external to TIP. These collections are fetched from their respective servers using the corresponding retrieval service (see Figure 2) and made available to the user. The third item is fetched from within TIP and also presented to the user.

As the user makes their way to Hamilton (2 green point), the quadrant that the search point is in will be updated (SE) but the search remains in the same bounded region. The information available for Hamilton will now be fetched and displayed to the user. As the user continues along the west side of the North Island (3 pink point), the search point will be transferred to the parent node and its NW quadrant. Since the TIP tree contains no information on this area, nothing is retrieved for the user. Finally, the user makes their way to Wellington (4 cyan point, SW quadrant, same bounded region) and information on Wellington is now displayed to them.

8 Evaluation

In this section we present our empirical evaluation of the TIP tree. We compared the performance of the TIP tree against the original mqr-tree. The TIP tree

```plaintext
traversing = true;
search point (x,y) = findStartingNode(x,y);
while (traversing) {
    (x,y) = update-co-ordinates(x,y);
    /* is search point within the bounded region covered by current node?*/
    if((x,y) within node_region(X)) {
        Quad = determineQuad(X);
        /* does quad contain a co-ordinate?*/
        if(has_co-ordinate(Quad)) {
            Sights = access_all_sites(Quad);
            display_all_sights(Sights);
        }
        /* should we traverse in subtree?*/
        else if(with_node_region_of_child(Quad)) {
            X = child_node(X);
        }
        /* otherwise, there is no information to display, so nothing is done until co-ordinates are updated */
    }
    /* if search point is outside the node region covered by current node, but within space covered by index, then traverse in the parent node.*/
    else if(has_parent_node(X)) {
        X = parent_node(X);
    }
    /* otherwise, outside of space covered by index, so terminate search */
    else {
        traverse = false;
    }
}
```

Figure 5: TIP tree Navigation Pseudocode

uses the navigation strategy presented in the paper to process the user’s trajectory, while the mqr-tree processes a user’s trajectory by performing a search on every co-ordinate in the trajectory.

We first present an overview of our evaluation methodology. Then, we present and discuss the results of our tests, and finish with further discussion that arose from our evaluation.

8.1 Overview of Methodology

Here, we present the data sets, evaluation criteria and a brief overview of the tests that were performed.

Data sets. The evaluations used three sets of coordinates. Two of the three sets each contain 10,000 co-ordinates and consist of both real and synthetic points that represent various locations. The real co-ordinates represent locations of existing communities in New Zealand and were obtained from the World Gazetteer. The synthetic co-ordinates represent other locations in New Zealand and are randomly generated. The rationale for using additional randomly created locations is that a collection of documents may refer to a region that does not exist in the World Gazetteer (e.g. iceberg photos from off the coast of New Zealand) or may be located in a remote area that is not maintained in the World Gazetteer.

http://www.world-gazetteer.com
The third set of points represents a trajectory of 712 co-ordinates that represent a user’s 16km route from Hamilton to Cambridge, New Zealand. The trajectory was recorded using a mobile device while travelling by vehicle on the most direct route between the two cities (GPS trail capturing was done using the evertrail application on android4).

**Test criteria.** The primary performance criteria in all of our tests is the number of disk accesses required to locate whether data of interest exists at a specific location or not. We evaluate the worst case scenario – no caching is used. Therefore, every time a node is required, it must be fetched from secondary storage, which requires one disk access. A secondary performance criteria is the CPU time. However, we discovered that this value is negligible. We discuss the CPU results further in Section 8.5 below.

### Test setup
We performed four tests in our evaluations. The first test studies the effect of the number of points in the index (TIP tree and mqr-tree) on the number of disk accesses required to locate information. The second test is similar to the first, except the points are drawn over a smaller geographical region. This leads to an increased geographical density in points, while the same number of points is represented in the index. The third test studies the effect of the number of co-ordinates along the trajectory on the number of disk accesses required to locate information. The fourth test is similar to the third, but use the overall set of points drawn over a smaller geographical region. Each test is described in more detail and the results are presented below.

#### 8.2 #Overall Indexed Points
For the first test, we used a set of 10,000 points that consists of locations around the North Island of New Zealand. We performed ten test runs. Each run con-

---

4http://www.everytrail.com
structured both a TIP tree and an mqr-tree using a subset of the points. Then, both the TIP tree and mqr-tree processed the trajectory of 712 co-ordinates. The average number of disk accesses per co-ordinate was calculated for each data structure. Each of the ten test runs used 1000, 2000, 3000, up to 10,000 points to create the trees.

Figure 7 depicts the results of the test runs. From the graph, we observe that on average, the TIP tree only needs to execute one disk access per co-ordinate in the user’s trajectory. For the mqr-tree, the average number of disk accesses is significantly higher and increases as the number of points in the mqr-tree increases. For 1000 points, the average number of disk access is approximately 4.8 and for 10,000 points it is approximately 6.2. The reason for the improved performance of the TIP tree is because each “search” begins at the last node visited by the previous co-ordinate in the trajectory. If several subsequent co-ordinates are close to each other, then it is very likely that the same node will be visited repeatedly, and no further search to find the correct node is required. Because the original mqr-tree performs all searches from the root, this will result in a higher number of nodes being accessed.

8.3 Density of Points in a Region

For the second test, we evaluated the effect of density on the average number of disk accesses required to locate information. Here, we used a set of 10,000 points as in the first test. However, all points are locations from in and close to the Waikato Region, which is a subsection of the North Island of New Zealand. Therefore, we are looking at the effect of increasing density on the number of disk accesses required to locate information.

We perform the same test runs for the second test as performed for the first test. Figure 8 depicts the outcome of the second test. Again, we find that on average the TIP tree requires only one disk access per co-ordinate on the trajectory. However, we also find that, although the TIP tree has a lower average disk access over the mqr-tree, the improvement is not as significant as that found in the first test since the average number of disk accesses required by the mqr-tree is between 2 and 2.25.

This is a surprising result, since we had assumed that the higher the density of points, the more disk accesses that would be required. What is happening is the following. The mqr-tree and TIP tree achieve a significantly lower amount of coverage of space when a dense set of points are being indexed. This means that more points are being placed higher in the index, and fewer rectangles that represent coverage of space are being stored. This results in points (and therefore, information) being located much faster by the mqr-tree than in an mqr-tree that is indexing points that are more spread out geographically.

8.4 #Co-ordinates on Traversal Path

The last two tests we performed evaluated how the number of co-ordinates on a user’s trajectory affects the average number of disk accesses. Will too many or too few co-ordinates cause more disk accesses than necessary?

For the first of two tests, we used the set of 10,000 points that represents the North Island to construct the TIP tree and mqr-tree. For the second, we used the denser set of 10,000 points that represents the Waikato Region. For both tests, we performed ten test runs, using approximately 10%, 20%, 30% .... up to 100% of the co-ordinates in the Hamilton to Cambridge trajectory.

Figure 9 presents the results of the third test, while Figure 10 presents the results of the fourth test. We find that the TIP tree achieves a lower average number of disk accesses for each co-ordinate over the mqr-tree. However, in both cases, we also find that the average number of disk accesses is constant, regardless of the number of points on the trajectory that are used for locating information. This is a very significant result, especially with respect to efficiency of the TIP tree – fewer points does not result in an increase in the number of disk accesses, and therefore traversal remains efficient.

8.5 Further Discussion

Over all tests, we found that the TIP tree and its traversal algorithm performed better than the mqr-tree with respect to the average number of disk accesses per co-ordinate. We performed the worst-case evaluation here – specifically, that every time a node is checked, a disk access is required. We finish this section with related discussion on other issues that are related to this evaluation.

We also recorded the CPU execution time for each co-ordinate “search”. We found that the average execution time was less than 1ms. This time may be slightly higher if network transmission time was considered. However, given that minimal mobile web pages with a few links would be sent back to the user in the worst case, we feel that in many cases that network transmission time will be negligible too. Therefore, we feel that disk access time is the crucial factor in how well this system performs. However, further testing should be done to confirm this.

We also did not perform a best-case empirical evaluation at this time for a number of reasons. First, we found that the worst-case evaluation produced efficient results. Also, we found that in many cases, the same nodes were being accessed repeatedly. For the TIP tree, this would be one node. For the mqr-tree, this would be the same path of nodes to get to the one node that several co-ordinates will require. If the nodes were cached in memory, this would result in an average disk access of almost zero. Checking the cache may increase the execution time, depending on which cache management strategy is utilized. However, if caching takes place on the mobile device instead of on the server, this would eliminate most of the network transmission cost, the disk access cost and the execution time on the server. If the TIP tree is used for managing access to external resources, then most of the time only one node would be to be stored by the mobile device. If the mqr-tree is used, then cache size and cache management would be an important factor in how well the index performs in the “best case”. Caching is future work that we will be looking into.

9 Conclusion and Remaining Issues

In this paper, we present a framework for accessing both internal information and resources from external digital libraries from within a mobile information system. Using the TIP framework, we show how resources can be organized using the TIP tree so that uniform access to both TIP information and external information from collections managed by different digital library software packages can be achieved. We also present an algorithm for traversing the TIP tree.
to continually provide information to the user, so that repeated searching from the beginning is not necessary. An empirical evaluation shows the efficiency of the TIP tree and the traversal algorithm for handling a user’s trajectory as they fetch information from digital libraries based on their location.

Other contributions of this work include: 1) one-pass access to any digital library collection, 2) no requirement of a “geographically aware digital library collection, and 3) the ability to handle collections that reference multiple locations. In conclusion, our work provides a strategy for context-aware mobile systems to co-operate with digital libraries in a seamless and efficient manner.

From our current work, there are some outstanding issues that need to be addressed. First, navigation only works for co-ordinates. Two search additions that we want to include are:

- A region search where all information within a specified radius or rectangular region is fetched. An issue with the existing co-ordinate based search is the “oscillation” between two different result sets if the user crosses continuously over a node or quadrant boundary. For example, if the user crosses back and forth over the boundary that separates Auckland from Hamilton, the information that is presented to the user will also switch back and forth. Incorporating a region search will solve this problem by presenting either both sets of information to the user, or a subset of it. In either case, the information that is presented to the user in this situation will not be in constant change. Although the TIP tree supports region searching (which is inherited from the mq-trie), it currently does not support the navigation of a search region. This needs to be addressed.

- Incorporating searching on other forms of context, such as user interests. This could easily be incorporated by performing an addition search on the results of the location context search. However, it is more desirable to include other types of context into the indexing structure so that only one search is required.

Second, the ability to have more general to more specific searching (i.e. searching higher up versus lower down in the tree) is desired. Both of these require modifications to the current TIP tree structure so that all leaf nodes are on one level of the tree and all data resides in the leaf level.

Third, the use of node caching and the best-case scenario need to be explored and evaluated. As mentioned in the evaluation, several nodes appear to be fetched repeatedly, which if cached would lead to savings in disk access, CPU and communication costs. However, because a mobile device has less storage than a server, and the same device is also running other programs and services, we cannot assume that all nodes that are fetched can be cached on the device. Therefore cache management strategies for nodes needs to be explored.

These and other issues that may arise will be addressed in future work.
References


Witten, I. H., Bainbridge, D. & Nichols, D. (2009), How to Build a Digital Library, Elsevier Science Inc.
Optimized XPath evaluation for Schema-compressed XML data

Stefan Böttcher, Rita Hartel, Stefan Heindorf
University of Paderborn (Germany)
Computer Science
Fürstenallee 11, D-33102 Paderborn
{stb@, rst@, heindorf@mail.}uni-paderborn.de

Abstract
XML has become the de facto standard for data exchange in enterprise information systems. But whenever XML data is stored or processed, e.g. in form of a DOM tree representation, the XML markup causes a huge blow-up of the memory consumption compared to the data, i.e., text and attribute values, contained in the XML document. In this paper, we present an optimized XPath query evaluation for XSDS, an XML compression approach based on removing information that is obsolete as this information can be derived from the existing XML Schema definition (XSD). Thereby, XSDS allows for storing and exchanging XML data in a space efficient and still queryable way. While previous papers have shown that XSDS generally reaches stronger compression ratios than other approaches like gzip, bzip2, and XMILL and that XPath queries can be evaluated on XSDS compressed data, we show in this paper that when optimizing the query evaluation on XSDS compressed data by using the given schema information, we can speed up query evaluation by a factor of 13 reaching evaluation times that are more than 5 times faster than those of JAXP – the standard Java XPath evaluator. The speed up was reached by avoiding the decompression of large parts of the structure while evaluating the query.

Keywords: XML, XPath evaluation, XML compression

1 Introduction

1.1 Motivation
XML gains more and more popularity not only as a data exchange format but also as a storage, archive or data management format and XPath is the main standard to express path queries on XML data.

Whenever XPath query evaluation is a bottleneck of an application, a fast XPath query evaluator is desired. In addition, XML documents may become larger than the available main memory space. Therefore, it may be a significant advantage when the fast XPath query evaluator can process XPath queries on compressed XML documents that can still fit into main memory. Previous work ((Böttcher, Hartel & Messinger 2009), (Böttcher, Hartel & Messinger 2010), (Böttcher, Hartel & Messinger 2011)) has shown that XSDS shows stronger compression facilities than other generic data compressors and than other XML-specific compressors while at the same time allowing for query evaluation and direct search on the compressed data at a speed that is comparable to that of XPath query evaluation on the original uncompressed XML data. This can be achieved by using a generic XPath evaluation approach (e.g. (Böttcher & Steinmetz 2007)) that reduces query evaluation to navigation via the binary axes first-child, next-sibling and parent. By replacing this generic approach with a specific one that at the same time regards the additional knowledge that can be gained by the provided XML Schema Definition, we provide in this paper an XPath evaluator for XSDS compressed XML data that allows for evaluation times that are 13 times faster than the previously used generic approach and 5 times faster than query evaluation on uncompressed XML data using JAXP.

1.2 Contributions
In this paper, we present an optimized approach to XPath evaluation on XSDS compressed data that combines the following advantages:
- The approach allows for XPath evaluation directly on the compressed data, i.e., unnecessary decompression of the compressed data is avoided whenever possible.
- The approach uses the provided schema information to reduce the amount of compressed data to be tested during the evaluation process and allows for skipping large irrelevant parts of the document, whereas our previous approach had to parse at least the compressed document structure that comes in document order before the current context node.
- The approach suggests an additional index on the compressed data that does not significantly blow up the compressed data, but that allows for skipping complete sub-trees of the compressed XML document.

Combining all these properties, the presented approach allows for an efficient XPath evaluation on XSDS compressed data that is more than 13 times faster than the previously used generic approach for XPath evaluation on XSDS compressed XML data and up to 5 times faster than XPath evaluation on uncompressed XML data.

1.3 Query language
The subset of XPath expressions supported by our approach conforms to the set of core XPath as defined in (Gottlob, Koch & Pichler 2005). This set is defined by the following EBNF grammar:

```
<xml>
locationstep ::= locationpath
locationpath ::= locationstep ('/' locationstep)*
```

Copyright (c) 2012, Australian Computer Society, Inc. This paper appeared at the 23rd Australasian Database Conference (ADC 2012), Melbourne, Australia, January-February 2012. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 124. R. Zhang and Y. Zhang, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.
locationstep ::= x '::' t | x '::' t ['[ pred ']]
pred ::= locationpath

“exp” is the start symbol, “x” represents one of the axes (self, child, descendant-or-self, descendant, following-sibling), and “t” represents a “node test” (either an XML node name test or “*”, meaning “any node name”).

1.4 Paper organization
This paper is organized as follows: Section 2 summarizes the basic ideas followed by a description of how XSDS is being used for compressing XML data. Section 3 describes the fundamental concepts used by our approach to evaluate XPath queries consisting of a single path or XPath queries with filters. Furthermore, this section describes the index that allows for skipping sub-trees and the evaluation of XPath queries with filters. The fourth section outlines some of the experiments that compare our prototype with other XPath evaluators. Section 5 gives an overview of related work and is followed by the Summary and Conclusions.

2 Compressing the data
2.1 The basic idea
The main compression principle of XML schema subtraction (XSDS) is to remove all information that is strictly defined by the XML schema definition from a given XML document, and to encode only those parts of the XML document in the compressed format that can vary according to the XML schema. In this paper, we only provide a short overview of XSDS compression, as details are described in (Böttcher, Hartel & Messinger 2009), (Böttcher, Hartel & Messinger 2010), and (Böttcher, Hartel & Messinger 2011).

2.2 This paper’s example

Fig. 1 shows a visualization of our example schema S called rule graph of S. The number in the lower part of each node represents the node ID of each node; the upper part represents the node’s label.

2.3 Compressing the Document

Compressing the structure. Within the structure of an XML document, i.e., within the element tags, there are only three different concepts that allow for variant parts within an XML document defined by a given schema: First, when the schema (XSD) requires the choice of one out of different given alternatives. Second, when the XSD element ‘all’ requires the occurrence of all elements declared by children of the ‘all’ element, but they can occur in any order. Third, when the XSD requires a repetition of elements, which allows for a varying number of elements (including all its descendant elements).

The compression of these variant parts within an XML document works as follows. Each compression step assumes that we consider one current position in the XML document at a time for which the XSD allows variant parts. For each current position in the XML document for which the XSD allows a choice, we only store the alternative chosen at this current position. For each XSD element ‘all’, we only encode the order of the elements required by the children of the ‘all’ element in the XSD. Finally, for each repetition of elements starting at a given position within an XML document, we only store the number of occurrences of this element found at the current position of the XML document.

Compressing the Textual Data. Beneath the structure, an XML document contains textual data. We store the text data in document order in a text container and apply gzip on top of the container at the end of the document.

3 Optimized query evaluation
3.1 The basic idea
The goal of our optimized query evaluator for schema-compressed XML data is to avoid considering and thereby decompressing parts of the compressed document that cannot contribute to the query evaluation, i.e., we want to skip those parts of the compressed document D that do not need to be read to answer the given query.

As for XSDS the existence of an XML schema is required, we use the information given by the schema’s rule graph to optimize the evaluation of a given XPath query Q as follows. For each rule graph node, we determine whether or not it is relevant to Q, in such a way that those parts of D that correspond to non-relevant rule graph nodes can be skipped.

Let us consider for example the query //meta. Without additional knowledge, the complete document would have to be read, as an element node with label ‘meta’ could occur at any depth within the document. But if we consider the schema shown in Fig. 1, we know for example that there does not exist an element node with label ‘meta’ that is a descendant of a node with label ‘content’. Therefore, we can skip the sub-tree below the content node and continue the query evaluation with the next-sibling of the content-node.

In order to benefit from the XML schema definition given in the schema’s rule graph within the query evaluation, intuitively, we determine for each location...
step LS of a given query which paths through the schema’s rule graph are needed to generate a minimal XML document fragment that is sufficient to answer LS. We collect these relevant paths in a so called navigation guide, such that restricting the evaluation of location steps in the compressed XML document to paths in the navigation guide skips not only irrelevant parts in the rule graph, but also in the compressed XML document.

3.2 An optimization step extending the rule graph

Whether or not a next rule graph node is relevant to a location step of an XPath query depends not only on the current rule graph node, but on the path through the rule graph to the current rule graph node taken by previous location steps. That is, in general, relevance of a next rule graph node cannot be decided locally depending on a current rule graph node, and therefore, whether or not a next XML node has to be read cannot be decided locally on the current XML node and current rule graph position.

In order to further reduce the relevant rule graph nodes and thereby reduce the parts of the compressed XML document to be read, we do the following preprocessing step before building the navigation guide. We extend the rule graph, such that whether or not a rule graph node is relevant to a location step of an XPath query can be determined locally on the extended rule graph (ERG) node, i.e., independent of the path via which the ERG node was reached when evaluating the XPath query.

The extended rule graph ERG then fulfills the unique successor set property that each element node en of ERG uniquely defines the set ENAX of ERG nodes that can be reached via any of the axes ax ∈ {self, child, descendant-or-self, descendant, following-sibling} of en, i.e., ENAX=\{enax | enax ∈ en/ax\} is not larger than the set of nodes PENAX that can be reached by a certain path P from the root of ERG via en using the axis ax with PENAX=\{penax | penax ∈ /P/en/ax\}.

For example, this unique successor property does not hold for the element-node <meta> of the rule graph given in Fig. 1, as this element has different following-sibling nodes depending on whether it has the parent node <article> or the parent node <news>, i.e. //article/meta/next-sibling::* is a <headline> element or a <meta> element, however //news/meta/next-sibling::* is an <article> element or a <meta> element. Note that the axes self, child, descendant-or-self and descendant already fulfill the unique successor property, and only the axis following-sibling does not always fulfill the unique successor property, i.e., the rule graph is extended to have the unique successor property fulfilled for the following-sibling axis.

Whenever a rule graph node n does not fulfill the unique successor set property, we split the node n and its incoming paths in RG into a minimum number of nodes n1,...,np and a minimum number of incoming paths to n1,...,np in ERG as follows, such that each of the nodes n1, ..., np fulfills the unique successor set property.

Splitting a node n and its incoming paths in RG means that n in RG is replaced with n1,...,np in ERG, and the incoming paths to n are replaced by incoming paths to n1,...,np in ERG, such that each path in ERG to n1 or ... or np has a corresponding path in RG and vice versa, where a corresponding path has the same number of nodes and edges and the same sequence of node labels. Splitting a node n in RG into nodes n1,...,np in ERG includes that for each node nk in ERG, each edge (n, ns) from n to a successor node ns in RG is copied to an edge (nk, ns) in ERG. The remaining rule graph (e.g. c.f. Fig. 2) then fulfills the unique successor set property.

Fig. 2. Extended rule graph (ERG) of the rule graph of Fig. 1 after the preprocessing

Note that the ERG is not necessarily a tree, but is a graph that still may contain cycles. Fig. 3 shows on the left-hand side an example of a (simplified) graph that contains cycles and does not fulfill the unique successor set property for the nodes with label ‘e’ and ‘f’, and it shows on the left-hand side an extended graph that fulfills the unique successor set property although it still contains cycles.

Fig. 3. Example of a recursive graph and the corresponding extended graph fulfilling the unique successor set property

3.3 Constructing the Navigation Guide

The navigation guide shall contain all the paths through ERG that correspond to an answer to the query for at least one legal XML document. In order to construct the navigation guide, we “simulate” the query evaluation on ERG location step by location step, but without any access to or knowledge about the real compressed data.

Within a pre-processing step using the looking forward approach, each XPath query is transformed into an equivalent XPath query \( Q \) that does not contain any backward axes. Although \( Q \) does not contain any backward axes, for simulating on ERG the execution of a following-sibling axis contained in \( Q \), we have to navigate backwards and forward again on ERG. In order to illustrate and to execute the required navigation steps, we construct a data structure called navigation guide from the ERG. The navigation guide \( NG \) for a given XPath query \( Q \) is constructed from \( Q \) and the ERG in two consecutive blocks of steps.

Computing the nodes being reachable by the query.

In the first part of our approach, we copy paths of nodes
from ERG to corresponding paths of nodes in NG for each location step LSi of the query Q. We say that a path to a node n in NG is corresponding to a path to a node e in ERG and we call the node n corresponding to e, if both paths have the same sequence of node names, ignoring the direction of edges in the graphs of ERG and NG. Paths through the ERG are copied to corresponding paths in NG for only those ERG nodes which define the XML nodes that are accessed when LSi is evaluated.

For example, if we consider the ERG given in Fig. 2 and the query //meta, we find the paths 1-2-3-4 and 1-2-6-7-8-9-4’ through ERG for the location step //meta. These paths are copied to the NG.

Furthermore, although NG may contain cycles for recursive schemas, NG grows like a tree, i.e., common prefixes of paths in NG are stored only once, and paths copied to NG are appended to NG’s leaf nodes. The common prefix is 1-2, and the leaf nodes of NG are 4 and 4’.

More precisely, at the beginning, NG consists only of a single node that is copied from the root node of ERG, the node 1 in the example.

Then, repeatedly for each location step LSi, NG is modified depending on the axis of the current location step LSi=axis::nnt as follows.

If the axis is the self-axis, the leaf nodes n in NG not corresponding to an ERG node e that conforms to the node name test nnt and the incoming edges to n are deleted from NG.

Except for self-axis location steps, NG is extended by copying ERG paths to corresponding NG paths and attaching the copies of paths starting in an ERG node e to the corresponding leaf node n of NG.

If the axis is the child-axis, for each leaf node n in NG the corresponding node e in ERG is determined, and each path in ERG from e to a next element node e2 in ERG not containing any other element node except of e and e2 is determined. For each element node e2 that fulfills the given node name test nnt, the path from e to e2 is copied to NG, i.e. appended to n.

If the axis is the descendant-axis, for each leaf node n in NG the corresponding node e in ERG is determined, and each path in ERG from e to any further element node e2 in ERG is determined. For each element node e2 that fulfills the given node name test nnt, the path from e to e2 is copied to NG, i.e. appended to n.

If the axis is the following-sibling-axis, for each leaf node n in NG the corresponding node e in ERG is determined. Starting from e, those element nodes e2 in ERG are determined that define a following-sibling of the XML elements defined by e. To find a connection from e to these element nodes e2 in ERG, we have to consider the semantics of the “( , , )”, “*” and “+” nodes. Then, connections from e to e2 are copied to paths in NG for each element node e2 that fulfills the given node name test nnt, even if the connection uses edges of ERG in reverse direction.

For example, to find a connection in ERG to a next-sibling <headline> element e2 when starting in the <meta> element e represented by node 4’ in Figure 2, we have to pass the nodes 4’-9-8-10, i.e. we have to go from 4’ via 9 to 8 in opposite direction of the edges found in ERG. Nevertheless, the path copied to NG contains copies of the nodes 4’, 9, 8, and 10 in this order.

To extend the example, if we consider the rule graph given in Fig. 2 and the query //meta/following-sibling::headline, we find the paths 1-2-3-4 and 1-2-6-7-8-9-4’ for the location steps //meta. Starting from the node 4, we do not find a result node for the next location step following-sibling::headline, but starting from the node 4’, we find the path 4’-9-8-10. Note that the corresponding edges in the NG go from 4’ to 9 and from 9 to 8 as navigation in ERG to the following-sibling follows this path, although the corresponding edges in the ERG shown in Fig. 2 take the opposite direction.

Deleting nodes that cannot contribute to the query result: In a second block of steps, we do a depth-first search backwards from the NG nodes representing XPath query result nodes to the root node of NG and mark all visited nodes. Thereafter, we eliminate from NG all those nodes that are not marked as they lead to a result of one of the intermediate location steps that cannot contribute to a result of the query. The paths remaining after the second step form the final graph structure of the navigation guide.

Continuing the example, the input of our second block of steps consists of the sub-graph containing the paths 1-2-3-4 and 1-2-6-7-8-9-4’-9-8-10. The result of the depth first search backwards that uses the edges of the NG in opposite direction starting at node 10 (the single result node) does not visit the nodes 3 and 4, i.e. the nodes 3 and 4 are deleted. Thus, the final NG shown in Fig. 4 only contains the path 1-2-6-7-8-9-4’-9-8-10. Here, the sub-path 1-2-6-7-8-9-4’ conforms to location step //meta and the sub-path 4’-9-8-10 conforms to location step following-sibling::headline.

Note that more than one copy of the same ERG node e might occur in the navigation guide as e is visited by different location steps, e.g. two copies of each of the nodes 8 and 9 occur in NG.

3.4 Evaluation of path queries without filters
The evaluation of a query based on the extended rule graph (ERG) and on the navigation guide (NG) is similar to the decompression of the document combined with a skipping of irrelevant sub-trees. This means, while we sequentially process compressed XML data, we walk synchronously in pre-order through the ERG and through
the NG. Whenever we pass a ‘*’-node in ERG, we determine how often to walk through its ERG sub-tree, and whenever we pass a ‘|’-node in ERG, we determine which path in ERG to follow. But in contrast to the decompression of XSDS compressed XML data based on ERG alone, we follow only those ERG paths that have a corresponding path in NG, and thereby skip large, irrelevant parts of the compressed document.

In addition to the graph structure of the navigation guide, we maintain a stack S that contains sets of navigation guide nodes and that represents the walk through the navigation guide and is handled synchronously to the recursion stack of the pre-order walk through the ERG.

We start the query evaluation at the beginning of the compressed data and in the root fn of NG (where fn corresponds to fe), and the stack S contains only one set containing the first node fn of the navigation guide NG.

Whenever the recursive pre-order walk through the ERG requires following an edge from a node e1 to a node e2, where e1 corresponds to a node n1 in the set on top of the stack S, we check, whether there exists an edge in NG from n1 to an NG node n2, such that e2 corresponds to n2. If at least one such node n2 ∈ NG exists, we proceed to n2 in the ERG and push to S the set NG2 that contains all nodes n2 ∈ NG to which n2 corresponds and that are children of any node n1 ∈ NG that was on top of stack. If there is no such edge in the navigation guide, we skip the sub-tree beyond this edge in the ERG.

For example, consider that we are currently in node 2 of ERG shown in Fig. 2, and the stack S contains the two sets [{//meta:1}, {/meta:2}], i.e., the set at top of S contains only the node with ID 2 shown in Fig. 4 as one of the NG nodes generated for the location step //meta. The first outgoing edge of node 2 in ERG is the edge E1=(2,3). But as there is no corresponding edge (/meta:2,x) in NG, we can skip the sub-tree rooted by node 3 in ERG. Instead, we check the next outgoing edge E2=(2,6) of node 2 in ERG. As we find the corresponding edge (/meta:2,meta:6) in NG, we follow the edge E2 in ERG and push to S a set containing the node /meta:6 to the stack S, i.e., we get S= [{//meta:1}, {/meta:2}, {/meta:6}].

Whenever the recursive walk through ERG requires a backtracking step (i.e. a pop-operation on the recursion stack) such that we go back from a node en2 to a node en1 in the ERG, we perform a pop-operation on S as well. In addition, we add the set NG1 to the new top-of-stack that contains all nodes n1 that correspond to such a node en1 and that are children of any node n2 that was on top of stack.

If we perform for example the backtracking from node 4’ to node 9 in the ERG, we pop the entry //meta:4’ (for the node of location step //meta with ID 4’) from the stack, such that the entry //meta:9 becomes the new top-of-stack. In addition, we add the entry /following-sibling::headline:9 to the current top-of-stack, as there is an edge from //meta:4’ to /following-sibling::headline:9 in the navigation graph, such that the new top-of-stack consists of the entries //meta:9 and /following-sibling::headline:9.

Whenever the result node of the navigation guide (i.e. the node /following-sibling::headline:10 in our example) is stored on the stack, the current position within the compressed data is a result to the query.

3.5 Skipping sub-trees
As the compressed data generated by XSDS does not allow for skipping sub-trees of a compressed XML document without additional information, we have extended the XSDS compression in such a way that it inserts a small skip-index into the compressed data. This index stores the address within the compressed data of the end of each repeated XML sub-tree structure that conforms to a sub-tree of the ERG that has a repetition node with label ‘*’ as root. Such an address consists of a position in the compressed XML structure and a position in the compressed text values of the XML document. Whenever the navigation guide allows for skipping a sub-tree with root node rn in the ERG, we walk through the ERG and the compressed representation recursively until we reach rn a second time (within the backtracking). Whenever we find a repetition node R in ERG within this skipping process, we read the address of the end of the compressed data belong to R and directly jump to that end position in the compressed structure and in the compressed text data, thereby skipping large parts of the compressed document.

3.6 Evaluation of queries with filters
Concerning the evaluation of predicate filters, we follow the idea of (Böttcher & Steinmetz 2007). Whenever a location step contains one or more predicate filters, we generate and activate a navigation guide FNG for the filter path. At the same time, we add a so-called reservation to the current entries of the set CS on top of the stack. This reservation is connected to FNG and ensures, that the entries of CS are considered valid, if and only if FNG reaches a final state, i.e., if and only if the filter can be evaluated to true for these nodes. Each filter navigation guide has the same principal design and functionality as the navigation guide of the main path of Q as described in the previous sections. Only if the execution of the filter yields true for a given stack entry, i.e., if the result node of the filter navigation guide is reached, the reservation is deleted and the entry is considered valid. If the execution of the filter navigation guide yields false for this entry, the entry itself is deleted and considered invalid.

The location-steps of the filter navigation guide can be connected to other filter navigation guides, such that nested XPath filter expressions can be evaluated by this concept as well.

4 Evaluation of our Prototype Implementation

4.1 Evaluation environment
We have compared the performance of our optimization to the performance of the previously used generic XPath evaluator (Böttcher & Steinmetz 2007), called XMLFramework, to the Java standard evaluator JAXP and to the state of the art approaches SAXON (SAXON...
The XSLT and XQuery Processor n.d.) and eXist (eXist: An open source native XML database n.d.). For this purpose, we used the following queries from the XPathMark performance benchmark (Franceschet 2005):

Q1: /site/closed_auctions/closed_auction/annotation/description/text/keyword
Q2: //closed_auction//keyword
Q3: /site/closed_auctions/closed_auction//keyword
Q4: /site/closed_auctions/closed_auction[annotation/description/text/keyword]/date
Q5: /site/closed_auctions/closed_auction[descendant::keyword]/date
Q6: /site/open_auctions/open_auction/bidder[following-sibling::bidder]

The following documents generated with XMark (Schmidt et al. 2002) served as our test documents (each document name is of the form d.<factor>.xml, where ‘factor’ is the /f parameter passed to the XMark generator):

Our tests were executed on an Intel Core 2 Duo 1.8 GHz with 2 GB of RAM. We used Microsoft Windows XP SP 3 with Java SE 6 Update 23. The java heapsize was set to 1024 MB. We used version 9.3 of SAXON and 1.4.0 of eXist. We ran at least 5 consecutive executions of each query.

XMLFramework and XE were executed directly on the XSDS compressed documents - the XMLFramework on the version without the skip index and XE on the version with skip index. JAXP and SAXON were executed on the original, uncompressed documents, and eXist uses its own dedicated database with indices.

<table>
<thead>
<tr>
<th>Name</th>
<th>Original Size</th>
<th>Size XSDS (w/o index)</th>
<th>Size XSDS + index</th>
<th>eXist</th>
</tr>
</thead>
<tbody>
<tr>
<td>d0.001.xml</td>
<td>0.11 MB</td>
<td>0.04 MB</td>
<td>0.04 MB</td>
<td>0.16 MB</td>
</tr>
<tr>
<td>d0.010.xml</td>
<td>1.13 MB</td>
<td>0.36 MB</td>
<td>0.36 MB</td>
<td>1.04 MB</td>
</tr>
<tr>
<td>d0.100.xml</td>
<td>11.32 MB</td>
<td>3.48 MB</td>
<td>3.59 MB</td>
<td>18.21 MB</td>
</tr>
<tr>
<td>d1.000.xml</td>
<td>113.06 MB</td>
<td>34.79 MB</td>
<td>35.94 MB</td>
<td>183.55 MB</td>
</tr>
</tbody>
</table>

Table 1. XMark documents and representation sizes used in our evaluation

4.2 Evaluation results

The evaluation results are shown in Fig. 5 and 6.

XXE outperforms the XMLFramework, JAXP and SAXON, whereas the advantage of using XXE gets larger with increasing document size (although all approaches scale linearly). Only eXist is faster by using larger indices and thus, causing higher storage requirements. Furthermore, XXE is clearly faster than completely decompressing the document, e.g. the decompression of d1.000.xml takes 37 seconds on our test machine.

5 Related Works

There exist several different approaches to the evaluation of XPath queries on XML data streams. Nearly all of them are based on automata (X-scan(Ives, Halevy & Weld 2002), YFilter (Diao, Rizvi & Franklin 2004), (Green et al. 2004), (Gupta & Suciu 2003), AFilter (Candan et al. 2006)) or parse trees ((Bar-Yossef, Fontoura & Josifovski 2007),(Barton et al. 2003), (Chen, Davidson & Zheng 2006)). All of them support the axes child and descendant-or-self and most of them support predicate filters and wildcards, but besides (Onizuka 2010) none of them support the sibling-axes as our solution does. All of these approaches require plaintext XML as input, typically in form of a SAX event stream. Applying these approaches to compressed XML would lead to a complete decompression of the compressed XML document into SAX events.

(Böttcher & Steinmetz 2007) and (Benter, Böttcher & Hartel 2011) are generic approaches that can be applied to XML representations that allow – similar as in this approach - navigation along the basic axes first-child,
Proceedings of the Twenty-Third Australasian Database Conference (ADC 2012), Melbourne, Australia

next-sibling and self plus parent in case of (Böttcher & Steinmetz 2007) and the inverse axes of first-child and next-sibling in case of (Benter, Böttcher & Hartel 2011). Therefore they can be applied to plain text XML as well as to compressed XML without forcing a complete decompression of the compressed XML data. (Arroyuelo et al. 2010) and (Maneth & Nguyen 2010) present a compressed representation for XML together with an XPath evaluator that is based on tree automata and that allows to skip irrelevant parts of the compressed XML document during the evaluation process. They allow selecting a single start point and follow the path to the root bottom-up and the path to the “leafs” of the query top-down.

Bisimulation (Buneman et al. 2005) and XQueC (Arion et al. 2007) are further approaches that allow the evaluation of XQuery instructions on top of compressed XML data. XQueC (Arion et al. 2007) proposes an XML representation that is optimized for an efficient transformation via XQuery. The structure compression as well as the data compression is chosen in such a way that path queries can be evaluated efficiently on it, but in return, the compression ratio reached by XQueC (Arion et al. 2007) is not as strong as by other compressors. The approaches being used in XQueC (Arion et al. 2007) appear to be non-applicable to other compression techniques. A document that is compressed via Bisimulation can be transformed with the help of XQuery into a compressed target document. In contrast to our approach, Bisimulation (Buneman et al. 2005) supports XPath expressions consisting of child axis steps only, whereas our approach supports all forward axes except the following axis.

6 Summary and Conclusions

Whenever the combination of XPath query evaluation with high storage costs or with low data transfer rates is a bottleneck of an application, using XSDS compressed data instead of the original XML data might be a solution to overcome this problem. We propose an optimized query optimization for XSDS compressed data that allows for XPath evaluation directly on the compressed data, i.e., unnecessary decompression of the compressed data is avoided whenever possible. Furthermore, the approach uses the provided schema information to reduce the amount of compressed data to be tested during the evaluation process and allows for skipping large irrelevant parts of the document. In order to allow the skipping of these irrelevant parts, the approach suggests an additional index on the compressed data that does not significantly blow up the compressed data.

Our experiments have shown that our approach outperforms previous approaches that allow the query evaluation on XSDS compressed data, and furthermore, it even outperforms the Java standard evaluator JAXP and the state of the art approach SAXON. In our experiments it was outperformed only by eXist. But in contrast to eXist, which needs more than 150% storage space of the original data, our approach only requires 32%, i.e., XXE decreases the storage and transfer costs, whereas eXist even would increase them.

7 References


Diao, Y, Rizvi, S & Franklin, MJ 2004, 'Towards an Internet-Scale XML Dissemination Service', *Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB)*, Morgan Kaufmann, Toronto, Canada.


Onizuka, M 2010, 'Processing XPath queries with forward and downward axes over XML streams', *EDBT 2010*, 13th International Conference on Extending Database Technology, Lausanne, Switzerland.


Efficient String-Based XML Stream Prefiltering

Stefan Böttcher, Rita Hartel, Steffen Weber
University of Paderborn (Germany),
Computer Science, Fürstenallee 11,
D-33102 Paderborn
{stb, rst}@uni-paderborn.de, steffen.weber@gmail.com

Abstract
Whenever huge XML documents have to be evaluated according to a given XPath or XQuery query, parsing the whole document in form of e.g. SAX events is the baseline that is common to all evaluators. But typically only few parts of the document are really relevant and can contribute to the query evaluation. We propose an approach to String-based prefiltering of an XML document D that outputs a smaller document D’ that contains the relevant parts of the document, such that the query Q evaluated on D yields the same result as Q evaluated on D’. In contrast to previous approaches, our approach extends the idea of efficient String-based XML prefiltering with support for XML Schema instead of DTDs, recursive schemata, and attribute filters. Our experiments on a 1 GB XMark document, taking the average over 22 queries, have shown that our approach outperforms previous prefiltering approaches and that it reaches an average speed-up factor of 8 compared to XQuery evaluation without prefiltering.

Keywords: XML, XML filtering, XML prefiltering, XPath, XQuery, XML data streams, XSD.

1 Introduction

1.1 Motivation
In many scenarios – as e.g. when filtering an XML-based news ticker – there is a huge amount of data that has to be filtered efficiently. Traditionally, approaches that filter XML streams or evaluate path queries on the XML data either generate a main-memory representation of the XML data in form of a DOM tree and work on that or they are based on XML events as they are provided by a SAX parser. All these approaches that are based on DOM trees or XML events have a common baseline run-time: the time needed to parse the XML text into the DOM tree or into XML events. In order to be able to outperform this baseline run-time, we propose a String-based prefilter that does not require parsing the data as XML tokens but can be applied to the textual representation of the XML document. This String-based approach restricts the prefilter’s expressiveness. However, this disadvantage is outweighed by the higher efficiency in filtering and separating the relevant parts of the document compared to an approach based on XML events.

Such a first String-based prefilter called SMP was introduced in (Koch, Scherzinger & Schmidt 2008). But SMP mainly has the following weaknesses: It cannot handle schema information in terms of XML Schema, but only in terms of DTDs. But XML Schema is much more widely used, and it allows more restrictive document definitions as DTD and therefore allows for a more accurate prefiltering. But of even more importance is that the approach introduced in (Koch, Scherzinger & Schmidt 2008) does not consider attribute filters, e.g. /a[@att='value'], although these filters provide a very high selectivity and therefore allow to reduce the relevant part of the document even stronger. Furthermore, the SMP prototype does not allow processing XML documents that are based on a recursive schema, as it is e.g. used by the XMark benchmark (Schmidt et al. 2002) or by XHTML files.

1.2 Contributions
In this paper, we present an approach that

- allows to filter a document D according to a given set of projection paths P derived from a query Q into a document D’ in such a way that Q applied to D yields the same result as Q applied to D’,
- reduces path search in XML documents to String matching problems for which efficient approaches already exist, i.e., our approach works on the String representation of the XML data, and no parsing of the String representation into XML tokens (as e.g. performed by a SAX parser) is necessary,
- considers schema information given as a DTD or as an XML Schema,
- allows to speed up query evaluation of existing XQuery or XPath evaluators up to a factor of 13 in our experiments,
- can handle recursive schema information, and
- considers attribute filters (e.g. /a[@att='value']) that allow to exclude large irrelevant parts from the source document.

In comparison to SMP (Koch, Scherzinger & Schmidt 2008), not only the scope of operations was increased by supporting XML Schema, recursive schemas, and attribute filters, but the String search algorithms were improved, such that our Java-based prototype even outperforms the C++-based SMP prototype.

1.3 Considered XPath syntax
The syntax for the projection paths that describe a superset of the parts of the document that are relevant to
answer the given query is defined by the following grammar with \texttt{ProjectionPath} as start symbol.

\begin{verbatim}
ProjectionPath ::= LocationStep+ Modifier?
LocationStep ::= Axis TagName AttributeFilter*
Axis ::= '/' | '//'
TagName ::= QName | '*'
Modifier ::= '@' | '#'
AttributeFilter ::= '[@' QName Operator Operator ::= '=' | '!=' | '<' | '<=' | '>' | '<='
''' String ''']'
\end{verbatim}

The projection paths only comprise the axes child and descendant as well as attribute filters. All query paths QP containing other axes have to be rewritten to a set of projection paths PP comprising the query result, such that QP applied on a document D yields the same result as QP applied to a document D' that was generated by filtering D according to PP. In order to eliminate all backward axes an approach as presented in (Olteanu et al. 2002) could be used.

The modifier ‘#’ defines, that the whole sub-tree rooted by the result node of this projection path has to be copied to the output, whereas the modifier ‘@’ defines that the result node including all its attributes has to be copied to the output. If the location step does not contain any modifier, only those nodes that are result nodes of the projection path (or of a prefix of the projection path) are copied to the output.

1.4 Example

The goal of our approach is to project a given XML document D to a smaller XML document D’ according to a given set of projection paths in order to speed up the query evaluation. Therefore, based on a given schema definition, we determine the relevant parts of the document using efficient String matching techniques.

\begin{verbatim}
1 <!ELEMENT a (b|c)>  
2 <!ELEMENT b (#PCDATA)>  
3 <!ELEMENT c (b,b?)>  
\end{verbatim}

(a)

\begin{verbatim}
1 <a>  
2  
3 <b><c><b>One</b></c></b>  
4 </a>  
\end{verbatim}

(b)

Fig. 1. (a) DTD and (b) XML document for our example

For example, we consider the projection path //a/b, the XML document given in Fig. 1 (a), and the DTD given in Fig. 1 (b). First of all, we have to search for an ‘a’-element at any depth of the document. As the XML specification allows the start-tag of an ‘a’-element to be written e.g. as \texttt{<a>} or as \texttt{a>} where ‘\texttt{>}’ denotes a space character, we search for matchings of the String ‘<a>’. We find a matching at the beginning of the first line of our document. From now on, three different XML events are relevant: the start of a ‘b’-element, the start of a ‘c’-element, and the end of the ‘a’-element. Therefore, we now search simultaneously for matchings of ‘<b>’, for matchings of ‘<c>’, and for matchings of ‘</a>’. The first matching found is the start-tag of the ‘b’-element in line 2 of our document. We continue our search with searching a matching of ‘</b>’, which is found at the end of line 2. Now, we have to search again for matchings of ‘<b>’, for matchings of ‘<c>’, and for matchings of ‘</a>’.

Although the next matching of ‘<b>’ can be found in line 3, this ‘b’-element is no result node of the query //a/b as ‘b’ is not a child of the ‘a’-element, but a a child of the ‘c’-element. In order to search for all relevant tags, we have to consider the information of the given schema. Form the schema definition for the ‘a’-element, we derive that whenever we are inside an ‘a’-element, we do not only have to search for the end-tag ‘</a>’ and for the start-tag ‘<b>’, but in addition, we have to search for the start-tag ‘<c>’ in order to skip the sub-tree rooted by the ‘c’-element.

1.5 Paper organization

This paper is organized as follows: Section 2 first gives a general overview of our approach and explains the construction of the element templates and of the filter automaton. Then, it describes the prefiltering process and discusses three optimizations of the filtering – jump offsets, recursive schemata and attribute filters. Section 3 contains a detailed analysis of the experiments performed on our prototype and Section 4 compares our approach with related works. Finally, Section 5 concludes the paper with a short summary and outlook.

2 Our Solution

2.1 Overview of our solution

The idea of our approach is to reduce the problem of optimized query evaluation for XML data to String pattern-matching, i.e., to search the next occurrence of one of a set of patterns within the textual representation of the XML document. Therefore, we first represent the given DTD or XML schema as a rule graph. Second, based upon the rule graph, we build so-called element templates that are the building-blocks for the filter automaton: each element-template represents the structure of an element as defined in the DTD or in the XML schema. Third, using the given projection paths, we combine these element templates to the filter automaton. This automaton decides for a given state (i.e., for a matched position within the XML document) what patterns to search for. It is independent of the XML document and only depends on the schema and the projection-paths.

2.2 Rule graphs

Fig. 2. Rule graph of the schema given in Fig. 1

We represent the schema information as a so called rule graph, a graphical representation of the schema. Such a
rule graph can be constructed for an XML schema definition in the same way as for a DTD. It contains the following types of nodes:

- element nodes (e.g. with label `<a>`) that represent a DTD element definition or an xsd:element-node,
- repetition nodes (with label `*`) that represent a repeated occurrence of sub-trees as e.g. defined by minOccurs and maxOccurs in XSD or as e.g. defined by the `*`, `+` or `?` operator in a DTD (with the values of `0` and `unbounded`, `1` and `unbounded` or `0` and `1` for minOccurs and maxOccurs respectively),
- choice nodes (with label `|`) that represent a choice operator in a DTD element definition or an xsd:choice-node,
- sequence nodes (with label `[,]`) that represent a sequence operator (,) in a DTD element definition or an xsd:sequence-node,
- value-nodes (e.g. with label `[STRING]`) that define the type of the `text`-value of an element, and
- nodes defining attributes.

Fig. 2 shows the rule graph of the example DTD given in Fig. 1.

### 2.3 Element templates

In order to generate a template of an automaton for an element el, called an element template, we first generate a pair of automaton states p1 and p1’ referring to the start-tag and the end-tag of el. That is, initially the element template contains the list $V=\{p1, p1\}$ of nodes and the empty list of edges $E=\{\}$. Then, we regard the element sub-graph of the element el in the rule graph, i.e., the sub-graph of the rule graph that contains each path P from the element-node en1 corresponding to el to an element-node en2 where P does not contain further element-nodes. The element sub-graph of the element ‘a’ is shown in Fig. 3(a).

![Fig. 3. (a) Element sub-graph of the rule graph for element a and (b) element template for element ‘a’](image)

Starting at the element subgraph’s leaf nodes, which are element-nodes, we compute bottom-up the sets StartStates[el] and FinalStates[el] for the corresponding element template as follows:

If the current node $cn$ is a

- leaf node (i.e., it is an element node ei), we add a new pair of states $pi$ and $pi'$ for ei, i.e., $V := V \cup \{pi, pi'\}$, and we set $\text{StartStates}[cn] := \{pi\}$ and $\text{FinalStates}[cn] := \{pi'\}$
- choice node, we compute the state lists as the union of the state lists of its children, i.e., $\text{StartStates}[cn] := \bigcup_{e \in \text{child}(cn)} \text{StartStates}[c]$ and $\text{FinalStates}[cn] := \bigcup_{e \in \text{child}(cn)} \text{FinalStates}[c]$
- repetition node, the state-lists are inherited from its (single) child node. Additionally, if the maxOccurs value given in the XSD is greater than 1 we add edges from the final-states of each child to the start-states of its next-sibling, i.e., $\text{StartStates}[cn] := \text{StartStates}[\text{firstChild}(cn)]$, $\text{FinalStates}[cn] := \text{FinalStates}[\text{lastChild}(cn)]$ and $E := E \cup \{(u, v)|c \in \text{child}(cn) \land u \in \text{FinalStates}[c] \land v \in \text{StartStates}(\text{nextSibling}(c))\}$
- root node (i.e., it is an element node), the state lists are inherited from its (single) child node, and we draw edges from the node p1 to the start states and from the final states to the node p1’, i.e., $\text{StartStates}[cn] := \text{StartStates}[\text{child}(cn)]$, $\text{FinalStates}[cn] := \text{FinalStates}[\text{child}(cn)]$ and $E := E \cup \{(u, v)\text{maxOccurs}(cn) > 1 \land u \in \text{FinalStates}(cn) \land v \in \text{StartStates}(cn)\} \cup \{(u, v)\text{minOccurs}(cn) = 0 \land \exists s, f; (u, s) \in E \land (f, v) \in E \land s \in \text{StartStates}(cn) \land f \in \text{FinalStates}(cn)\}$
- root node (i.e., it is an element node), the state lists are inherited from its (single) child node, and we draw edges from the node p1 to the start states and from the final states to the node p1’, i.e., $\text{StartStates}[cn] := \text{StartStates}[\text{child}(cn)]$, $\text{FinalStates}[cn] := \text{FinalStates}[\text{child}(cn)]$ and $E := E \cup \{(u, v)\text{maxOccurs}(cn) > 1 \land u \in \text{FinalStates}(cn) \land v \in \text{StartStates}(cn)\} \cup \{(u, v)\text{minOccurs}(cn) = 0 \land \exists s, f; (u, s) \in E \land (f, v) \in E \land s \in \text{StartStates}(cn) \land f \in \text{FinalStates}(cn)\}$

Nodes of type ‘ALL’ are treated similar to a repetition node with minOccurs=n and maxOccurs=n with a choice node as single child, where n is the number of children of the ‘ALL’-node.

If the element node might have empty content, we add an edge from p1 to p1’ to the element template. The element template for the element ‘a’ is shown in Fig. 3(b).

### 2.4 Building the filter automaton

**Overview of the filter automaton construction:** In the next part of our algorithm, we use a processing queue PQ in order to build the filter automaton from the element templates, depending on whether or not the elements are relevant for the location steps in the given projection paths. For this purpose, all projection paths (as defined in
the grammar given in Section 1.3) are processed in parallel and location step by location step.

The processing queue PQ contains pairs of states (p1, p1’) each of which has a label li and represents elements ei with the node name li of the XML document and has one or more projection paths associated to it that have to be evaluated against it. Before we describe the algorithm in general, we present an example.

Example: building the filter automaton for the projection-path //a/b#.

We start with an initial pair of states (p1, p1’) labelled ‘a’ (the document’s root element). The projection-path //a/b# is assigned to (p1, p1’), and (p1, p1’) enters the queue PQ.

Now the queue is processed. We extract (p1, p1’) and consider its only projection-path //a/b#. The tag-name of the first location-step LS1 ‘//a’ matches the label ‘a’ of (p1, p1’). Therefore (p1, p1’) is marked as being relevant and is assigned the action ‘copy tag’. We furthermore expand (p1, p1’) by inserting the element-template ‘a’, i.e. two new pairs of states are created: (p2, p2’) with label ‘c’ and (p3, p3’) with label ‘b’. Both are added to PQ and both are assigned the two projection-paths //a/b# and //a/b# (the latter because of the descendant-axis in the location-step LS1).

We extract (p2, p2’) with label ‘c’ from PQ. Its first projection-path ‘//b’ is skipped because the label ‘b’ of its first location-step does not match the label ‘c’ of (p2, p2’). Its second projection-path ‘//a/b#’ with a descendant-axis in its first location-step is skipped as well because the labels don’t match and because a ‘c’-element cannot contain an ‘a’-element according to the schema.

The pair of states (p2, p2’) stays irrelevant for now.

We continue with PQ’s last pair of states (p3, p3’) with label ‘b’. The label of its first projection-path ‘//b’ matches, and therefore (p3, p3’) is marked as being relevant. The modifier ‘#’ yields the action ‘copy on/off’. Furthermore, the algorithm notices that the pair (p3, p3’) that was marked as relevant has a sibling (p2, p2’) that might contain a ‘b’-element according to the schema. In this special situation, the sibling (p3, p3’) of the currently considered pair of states is marked as being relevant (but keeps its action ‘nop’ and is not expanded further) in order to skip this element and all its descendant nodes. The second projection-path ‘//a/b#’ of (p3, p3’) is skipped for the same reason that it was skipped at (p2, p2’).

Now PQ is empty. All three pairs of states have been marked as being relevant, and therefore all of them remain in the automaton.

Determining relevant nodes and node expansion:
The algorithm starts with a processing queue PQ containing only the pair (p1, p1’) of states that corresponds to the root element. All initial projection paths are assigned to this pair.

Then, repeatedly for the current top of queue (pi, pi’) of PQ and for each first location step LS1 of a projection path PP assigned to (pi, pi’), we check whether or not this pair of states is relevant for the location step as described in the remainder of this section. If this is the case, we mark (pi, pi’) as being relevant (i.e. it will not be removed from the final automaton). If, furthermore, LS1 is not the last location step of PP, the pair (pi, pi’) has to be expanded.

Let LP=LS1/LS2/…/LSn be a projection path attached to the pair (pi, pi’), currently on top of PQ, with li being the label of the element ei represented by the pair (pi, pi’) of states, and let LS1 be of the form LS1=axis::mnt.

Now, depending on the axis and the node-name test, the pair (pi, pi’) of states is marked as being relevant and is expanded depending on the following conditions.

If axis=child, the pair (pi, pi’) is marked as being relevant if the label li fulfills the node name test nnt. If furthermore n > 1, we have to expand the pair (pi, pi’) and thus replace it by the element template Eli for ei, add all other state pairs of Eli to PQ, and attach the projection path LS2/…/LSn to all other state pairs of Eli.

If axis= descendant-or-self, the pair (pi, pi’) is marked as being relevant if the label li of ei or the label of any of its possible descendants according to the schema fulfills the node name test nnt. Irrespective from the existence of further location-steps, we have to expand the pair (pi, pi’) and thus replace it by the element template Eli for ei, add all other state pairs of Eli to PQ, and attach the projection path LS1/LS2/…/LSn to all other state pairs of Eli. If additionally li fulfills the node name test nnt and if n > 1, we add the projection path LS2/…/LSn in addition to all state pairs of Eli including (pi, pi’).

In one particular case, siblings of (pi, pi’) might be marked as being relevant as well: If the pair (pi, pi’) has been marked as being relevant and if it has a sibling (pj, pj’) that represents an element which might have a descendant labelled li according to the schema, then the pair (pj, pj’) has to be marked as being relevant, too (c.f. the node <c> in line 3 of Fig. 1 that is a sibling of node with label ‘b’ and might have a descendant with label ‘b’).

If a pair (pi, pi’) remains relevant, but unexpanded after the algorithm has processed all projection-paths, we insert an edge from pi to pi’.

Assigning actions to pairs of states: During the construction of the automaton, we assign different actions to the pairs of states. The action ‘copy on’ defines that we start copying the XML document to the output until we read the action ‘copy off’. The action ‘copy tag’ defines that we copy the current tag to the output and the action ‘copy tag + attr’ defines that we copy the current tag including all its attributes to the output. The action ‘nop’ means no action or copying is done at all. Finally, there is an action ‘expand’ which is needed for recursive schemata and which is explained in Section 2.7. The actions have a precedence which is (copy on/off > copy tag + attr > copy tag > nop > expand).

Whenever the label li of the current state pair (pi, pi’) does not fulfil the node name test nnt of LS1, the action ‘nop’ is assigned to (pi, pi’). Otherwise, actions are assigned as follows: (pi, pi’) gets the action ‘copy tag’ if LS1 does not contain any modifier. Furthermore, (pi, pi’) gets the action ‘copy tag + attr’ if LS1 contains the modifier ‘@’. Finally, pi gets the action ‘copy on’ and pi’ gets the action ‘copy off’ if LS1 contains the modifier ‘#’.

148
An action can only be replaced by actions of higher precedence. Whenever a pair gets the action ‘copy on/off’, we do not have to expand it any more, as the corresponding sub-tree including all its descendant nodes will be copied to the output anyway.

Fig. 4 shows the filter automaton for the query //a/b# including the assigned actions.

### 2.5 Executing the filter automaton

The String-based prefiltering of an XML document using the above-described filter automaton works as follows.

The execution starts in the state p1 that corresponds to the start-tag of the root element of the XML document and with a pointer pointing to the position of the root element within the document. Starting at the current state pi, we search for all element labels (eli, ... , eln) that are attached to states that are successor states of pi. That is, we search for the String ‘<a’ in order to search for a start tag of an element with label ‘a’. Whenever we search for a single element label only, we use the algorithm of Boyer, Moore and Horspool (Boyer & Moore 1977), (Horspool 1980), and we use the Commentz-Walter algorithm (Commentz-Walter 1979), whenever we search for more than one element label at a time. Whenever we find an occurrence of a searched element label elj, the state pj carrying the label elj becomes the active state, and the process is continued with the state pj and the input stream position at which the label elj has been found.

At each state-transition the action of the new state is executed (usually something is copied to the output). The execution terminates when we have reached the state p1’ and have thus found the end-tag of the root element.

### 2.6 Skipping parts of the document

Each edge in the automaton represents the search for a new start-tag or end-tag and thus allows skipping parts of the document. For example, if we have found the start-tag of the element ‘c’ and search for the end-tag of the element ‘c’, the given schema defines that the ‘c’-element contains at least on ‘b’-element (c.f. Fig. 1). As the shortest representation of the ‘b’-element is ‘<b/>’, which has a length of 4 characters, we can skip the next 4 characters within our String search. This minimal number of characters that can be skipped are called jump offset.

The jump offset for each state is defined to be the shortest path to another relevant state where the edge-weight for that tag is chosen depending on whether an empty-element tag representation is possible (min-length of content = 0) or not.

The jump offset can be calculated during the construction of the automaton and does not need to be calculated during runtime. Jump offsets can further reduce the number of positions to be searched and therefore accelerate the filtering process. XML Schema allows the computation of larger jump-offsets than DTD because many of its data-types have a minimum length. This may especially forbid the representation of an element as a short empty-element-tag as in the example above.

### 2.7 Recursive schemata

If we consider recursive schemata, i.e., schemata whose graph contains cycles, the construction of the automaton as described above would not terminate. If there is a projection-path with a descendant-axis location-step the node-name of which matches a label on a cycle within the schema graph, this would lead to an infinite process of replacing the current pair of states by its element template without changing the attached projection path.

But as the XML document only has a limited depth, we do only need to expand the automaton up to a fixed size limited by the document depth. As the depth of the document is not known at the time of the construction of the filter automaton, we expand the recursive parts of the automaton at runtime only if the current document depth requires a further expansion.

Whenever during the construction of the automaton the expansion of a pair of states (pi2, pi2’) is required that is included in a second pair (pi1, pi1’) belonging to the same element (i.e., there exists a path from pi1 to pi2 and a path from pi2’ to pi1’ in the schema graph), we do not expand the pair of states (pi2, pi2’), but instead attach the action ‘expand’ to pi2. Whenever during the execution of the filter automaton a state with action ‘expand’ is reached, the pair of states is replaced by the corresponding element template, i.e., the expansion is performed during runtime.

Whenever such a pair of states (pi2, pi2’) that is part of a recursion is marked as being relevant, but is unexpandable, the edge from pi2 to pi2’ is assigned a special search algorithm for locating the end tag specified by pi2’. As the current element with label eli might contain nested elements with the same element label eli, it is not sufficient to search for the pattern ‘</eli’. Instead, we have to respect the proper nesting, i.e. while searching for the end-tag ‘</eli’, an equal number of start-tags ‘<eli’ and end-tags ‘</eli’ must be skipped.
2.8 Attribute filters

The syntax of the projection paths allows for attribute filters, i.e., predicate filters of the form `@att=const` that compare attribute values to a given constant.

For each step location that contains attribute filters, we duplicate the pair of states (pi1, pi1’) to (pi2, pi2’) with all its incoming edges and outgoing edges. One copy (e.g. (pi1, pi1’)) corresponds to elements for which the filter evaluates to true, whereas the second copy (e.g. (pi2, pi2’)) corresponds to elements for which the filter evaluates to false. Both are marked as relevant and the copy (pi1, pi1’) is assigned an action different from ‘nop’ (i.e. ‘copy tag’, ‘copy tag+attr’ or ‘copy on/off’), whereas the copy (pi2, pi2’) is assigned the action ‘nop’. Whenever an element corresponding to these pairs of states is found, it is checked whether or not the attribute filter can be evaluated to true and the state belonging to the current case (either true or false) is set active.

In general, N attribute-filters could apply to the same node-pair in the automaton. Only in this case, N copies of this node-pair are not sufficient, and instead, we need one copy for each possible combination of filter results.

Although a simple satisfiability-check is performed to avoid obviously conflicting filter combinations, this handling of attribute-filters leads to an exponential duplication of pairs of states. Nevertheless, the approach as a whole profits strongly as attribute filters typically are highly selective.

3 Evaluation of our prototype implementation

We have developed an implementation called StringFilter that features the algorithm presented in this paper.

3.1 Experimental setup

We have chosen XMark documents of sizes up to 2 GB for our measurements. In case of the XMark queries Q1 and Q4, we have created two different sets of projection-paths – one with, the other without attribute filters.

All measurements have been conducted on an HP EliteBook 8730w (Intel Core 2 Duo T9400, 4 GB RAM) running Gentoo Linux 64 Bit, Java 6.0.23 and GCC 4.5.2. In order to prevent the HDD from impacting the results, all input-documents were buffered in memory and the results were not written back to disk but to ‘/dev/null’.

The projection paths used in our evaluation are shown in Table 2.

3.2 Comparison to other filters

The runtime of StringFilter and SMP-Tool (Koch, Scherzinger & Schmidt 2008) on a 2 GB XMark document is depicted in Figure 5. On average, StringFilter needs 2.9 seconds to filter the document. Using attribute-filters (‘AF’) in the projection-paths slightly increases the runtime (see ‘Q1 AF’ and ‘Q4 AF’ in the diagram).

Compared to SMP-Tool, StringFilter is 16% faster on average. Since SMP-Tool lacks support for recursive schemas it cannot filter a document for the evaluation of Q15 and Q16. It furthermore does not support attribute filters.

![Table 2: Projection paths used in our evaluation](image-url)
Evaluation as a prefilter

On average, for the queries used in our evaluation, the size of the projected document is only 3.1% of the XMark document’s size. Therefore the runtime of an XQuery engine like Qizx or MonetDB can be reduced by adding StringFilter to the query-evaluation process.

Figure 6 compares the runtime of Qizx and MonetDB on unfiltered XMark documents with the combined runtime of StringFilter and Qizx/MonetDB on filtered XMark documents. The results are averaged over all XMark queries.

The memory-usage of the combined approach is dominated by the XQuery engine – StringFilter’s memory usage of about 4 MB + JVM is independent from the document size.

In addition to Qizx and MonetDB we did experiments with Saxon. Saxon ran out of memory when evaluating queries on the unfiltered 1 GB XMark document but succeeded when using the filtered documents (with the exception of Q8-Q12 where we aborted the evaluation after 10+ minutes).

For queries that use attribute-filters, our experiments have shown that on a 1 GB stream, the usage of those attribute-filters in the projection-paths reduces the combined runtime of StringFilter + Qizx by 43% in case of Q1 and by 12% in case of Q4 respectively.

4 Related Works

XPath or XQuery evaluators that focus on a fast evaluation of huge XML documents or of XML data streams typically require the tokenization of the String representation of the XML data into SAX events, StAX events or into a DOM tree. Therefore, they all require the same baseline run-time: at least the time for the tokenization is required in order to evaluate queries on the compressed representation.

One example for a recent approach that uses an automaton to process the XML token and thus has to work on the tokenized XML structure, but that has exactly this tokenization as baseline run-time is the approach presented in (Onizuka 2010). It supports the axes self, child, descendant, following and following-sibling but does not support backward axes. It translates the queries into expressions over the binary axes first-child and next-sibling and then constructs a two-layered NFA that consumes the SAX events start-element, end-element and character. The first layer evaluates the main path of the query, whereas the second layer is responsible for the evaluation of the predicate filters.

Mainly there are two different approaches to overcome this problem. The first one is to store the XML data in form of an indexed representation instead of storing it as a text file. This is typically done in form of additional index data or in form of a relational database representation. The approaches presented in (Boncz et al. 2006), (Halverson et al. 2004), (Pal et al. 2004), and (Sidirougos & Boncz 2009) follow this idea.

The other idea to overcome this problem is to add a String-based prefilter before the query evaluation itself. By using such a prefilter the tokenization is still required, but the document to be tokenized becomes much smaller such that the overall runtime decreases. The advantage of such a prefilter is that it does not pose any requirements on the query engine and can be combined transparently with any query engine working on the textual representation of XML data.

The approach presented in (Huang, Chuang & Lee 2005) and (Huang et al. 2006) forms a mixture of both approaches. It uses a small index that allows to efficiently determine the set of so called “candidate documents”, i.e., of document fragments on which the query is evaluated to
the same result as on the original data. These candidate documents can then be passed to any query evaluator to retrieve the original query result. In contrast to our approach, this index prevents it from being applied to XML data streams.

The approach presented in (Takeda et al. 2002) presents a String search algorithm that not only handles single-byte characters but also handle multi-byte symbols. When regarding each tag as a multi-byte symbol, this approach could be used for finding matchings of query location steps within an XML document. As the proposed approach does not consider the document structure as defined by given schema information, it might yield false positives as they are motivated in Section 1.4 (e.g. the ‘b’ element in line 3).

The SMP-Tool (Koch, Scherzinger & Schmidt 2008) follows the same idea as our approach. Although it was implemented in C, the implementation performs less efficient than our Java implementation. In contrast to our approach, SMP cannot handle schema information in terms of XML Schema, but only in terms of DTDs. But XML Schema allows more restrictive document definitions as DTD and therefore allow for a more accurate prefiltering. In addition, SMP does not consider attribute filters, e.g. /a[@att='value'], although these filters provide a very high selectivity and therefore allow to reduce the document even stronger. Furthermore, the SMP prototype does not allow to process XML documents that are based on a recursive schema.

5 Summary and Conclusions

Whenever huge XML documents have to be evaluated according to a given XPath query or XQuery query, parsing the whole document in form of e.g. SAX events is the baseline that is common to all conventional query evaluators. But typically only few parts of the document are really relevant and can contribute to the query evaluation. We propose an approach to String-based prefiltering of XML documents D that does not require to parse D into SAX events, into StAX events, or into a DOM tree and that outputs a smaller document D’ that contains all relevant parts of D, such that the query Q evaluated on D yields the same result as Q evaluated on D’.

Our experiments have shown, that our approach outperforms previous approaches and that it reaches a speed-up factor of up to 13 compared to XQuery evaluation using Qizx on a 2 GB document.

Our approach not only leads to a speed-up during query evaluation, but for some documents, it might even allow for the query evaluation for DOM-based query evaluators. In many cases the complete DOM tree of the document does not fit into main memory, whereas the DOM tree of the filtered document fits into main memory, such that a DOM-based query evaluation can only be performed on the prefiltered document, but not on the original document.

Furthermore, we assume that our approach is not limited to XPath queries using child and descendant axes, but can be extended to queries using the axes following and following-sibling as well.

6 References


Onizuka, M 2010, 'Processing XPath queries with forward and downward axes over XML streams', EDBT 2010, Lausanne, Switzerland.


Sidirourgos, L & Boncz, P 2009, 'Generic and updatable XML value indices covering equality and range lookups', Proceedings of the 2009 EDBT/ICDT Workshops, ACM.

### Author Index

<table>
<thead>
<tr>
<th>Author</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alahmari, Fahad</td>
<td>89</td>
</tr>
<tr>
<td>Böttcher, Stefan</td>
<td>137, 145</td>
</tr>
<tr>
<td>Cao, Jinli</td>
<td>81</td>
</tr>
<tr>
<td>Dietrich, Jens</td>
<td>31</td>
</tr>
<tr>
<td>Dobbie, Gillian</td>
<td>21</td>
</tr>
<tr>
<td>Feng, Boliang</td>
<td>61</td>
</tr>
<tr>
<td>Gu, Yu</td>
<td>41</td>
</tr>
<tr>
<td>Hartel, Rita</td>
<td>137, 145</td>
</tr>
<tr>
<td>Heindorf, Stefan</td>
<td>137</td>
</tr>
<tr>
<td>Hinze, Annika</td>
<td>127</td>
</tr>
<tr>
<td>Islam, Md. Saiful</td>
<td>11</td>
</tr>
<tr>
<td>Lin, Shukuan</td>
<td>41</td>
</tr>
<tr>
<td>Liu, Chengfei</td>
<td>11</td>
</tr>
<tr>
<td>Lu, Jiaheng</td>
<td>61</td>
</tr>
<tr>
<td>Magee, Liam</td>
<td>89</td>
</tr>
<tr>
<td>McCartin, Cathe</td>
<td>31</td>
</tr>
<tr>
<td>Naeem, M. Asif</td>
<td>21</td>
</tr>
<tr>
<td>Osborn, Wendy</td>
<td>127</td>
</tr>
<tr>
<td>Qi, Jianzhong</td>
<td>51</td>
</tr>
<tr>
<td>Qian, Weinong</td>
<td>99</td>
</tr>
<tr>
<td>Sattar, Abdul</td>
<td>117</td>
</tr>
<tr>
<td>Stantic, Bela</td>
<td>117</td>
</tr>
<tr>
<td>Tao, Yufei</td>
<td>3</td>
</tr>
<tr>
<td>Terry, Justin</td>
<td>117</td>
</tr>
<tr>
<td>Thom, James A.</td>
<td>89</td>
</tr>
<tr>
<td>Vines, Phil</td>
<td>71</td>
</tr>
<tr>
<td>Wang, Hua</td>
<td>5</td>
</tr>
<tr>
<td>Wang, Yanqiu</td>
<td>41</td>
</tr>
<tr>
<td>Webb, Geoffrey I.</td>
<td>7</td>
</tr>
<tr>
<td>Weber, Gerald</td>
<td>21</td>
</tr>
<tr>
<td>Weber, Steffen</td>
<td>145</td>
</tr>
<tr>
<td>Wen, Zeyi</td>
<td>51</td>
</tr>
<tr>
<td>Wong, Wilson</td>
<td>89</td>
</tr>
<tr>
<td>Xu, Chuanfei</td>
<td>41</td>
</tr>
<tr>
<td>Xu, Zhenghua</td>
<td>51</td>
</tr>
<tr>
<td>Xue, Yuan</td>
<td>51</td>
</tr>
<tr>
<td>Yang, Nan</td>
<td>61</td>
</tr>
<tr>
<td>Yang, Yang</td>
<td>81</td>
</tr>
<tr>
<td>Yu, Ge</td>
<td>41</td>
</tr>
<tr>
<td>Zhang, Rui</td>
<td>iii</td>
</tr>
<tr>
<td>Zhang, Xiuzehe</td>
<td>71</td>
</tr>
<tr>
<td>Zhang, Yanchun</td>
<td>iii</td>
</tr>
<tr>
<td>Zhang, Zhao</td>
<td>99</td>
</tr>
<tr>
<td>Zhao, Bin</td>
<td>99</td>
</tr>
<tr>
<td>Zhong, Hao</td>
<td>109</td>
</tr>
<tr>
<td>Zhou, Aoying</td>
<td>99</td>
</tr>
<tr>
<td>Zhou, Rui</td>
<td>11</td>
</tr>
<tr>
<td>Zhou, Ying</td>
<td>109</td>
</tr>
<tr>
<td>Zhou, Yonghuan</td>
<td>61</td>
</tr>
<tr>
<td>Zhou, Zhixin</td>
<td>71</td>
</tr>
</tbody>
</table>
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 113 - Computer Science 2011
Contains the proceedings of the Thirty-Fourth Australasian Computer Science Conference (ACSC 2011), Perth, Australia, 17-20 January 2011.

Volume 114 - Computing Education 2011
Contains the proceedings of the Thirteenth Australasian Computing Education Conference (ACE 2011), Perth, Australia, 17-20 January 2011.

Volume 115 - Database Technologies 2011
Contains the proceedings of the Twenty-Second Australasian Database Conference (ADC 2011), Perth, Australia, 17-20 January 2011.

Volume 116 - Information Security 2011
Edited by Colm Boyd, Queensland University of Technology, Australia and Josef Pieprzyk, Macquarie University, Australia. January 2011. 978-1-920682-96-5.
Contains the proceedings of the Ninth Australasian Information Security Conference (AISC 2011), Perth, Australia, 17-20 January 2011.

Volume 117 - User Interfaces 2011
Contains the proceedings of the Twelfth Australasian User Interface Conference (AUIC2011), Perth, Australia, 17-20 January 2011.

Volume 118 - Parallel and Distributed Computing 2011
Contains the proceedings of the Ninth Australasian Symposium on Parallel and Distributed Computing (AusPDC 2011), Perth, Australia, 17-20 January 2011.

Volume 119 - Theory of Computing 2011
Edited by Alex Potain, Victoria University of Wellington, New Zealand and Tony Saha, Queensland University of Technology, Australia. January 2011. 978-1-921770-00-5.

Volume 120 - Health Informatics and Knowledge Management 2011
Contains the proceedings of the Fifth Australasian Workshop on Health Informatics and Knowledge Management (HIKM 2011), Perth, Australia, 17-20 January 2011.

Volume 121 - Data Mining and Analytics 2011
Contains the proceedings of the Ninth Australasian Data Mining Conference (AusDM'11), Ballarat, Australia, 1-2 December 2011.

Volume 122 - Computer Science 2012
Edited by Mark Reynolds, The University of Western Australia, Australia and Bruce Thomas, University of South Australia. January 2012. 978-1-921770-04-3.
Contains the proceedings of the Thirty-Fifth Australasian Computer Science Conference (ACSC 2012), Melbourne, Australia, 30 January – 3 February 2012.

Volume 123 - Computing Education 2012
Edited by Michael de Raadt, Moodle Pty Ltd and Angela Carbone, Monash University, Australia. January 2012. 978-1-921770-03-6.
Contains the proceedings of the Fourteenth Australasian Computing Education Conference (ACE 2012), Melbourne, Australia, 30 January – 3 February 2012.

Volume 124 - Database Technologies 2012
Contains the proceedings of the Twenty-Third Australasian Database Conference (ADC 2012), Melbourne, Australia, 30 January – 3 February 2012.

Volume 125 - Information Security 2012
Contains the proceedings of the Tenth Australasian Information Security Conference (AISC 2012), Melbourne, Australia, 30 January – 3 February 2012.

Volume 126 - User Interfaces 2012
Edited by Haiying Shen, Flinders University, Australia and Ros T. Smith, University of South Australia. January 2012. 978-1-921770-07-4.
Contains the proceedings of the Thirteenth Australasian User Interface Conference (AUIC2012), Melbourne, Australia, 30 January – 3 February 2012.

Volume 127 - Parallel and Distributed Computing 2012
Edited by Jianjun Chen, University of Technology, Sydney, Australia and Rajiv Ranjan, CSIRO ICT Centre, Australia. January 2012. 978-1-921770-08-1.
Contains the proceedings of the Tenth Australasian Symposium on Parallel and Distributed Computing (AusPDC 2012), Melbourne, Australia, 30 January – 3 February 2012.

Volume 128 - Theory of Computing 2012
Contains the proceedings of the Eighteenth Computing: The Australasian Theory Symposium (CATS 2012), Melbourne, Australia, 30 January – 3 February 2012.

Volume 129 - Health Informatics and Knowledge Management 2012
Contains the proceedings of the Fifth Australasian Workshop on Health Informatics and Knowledge Management (HIKM 2012), Melbourne, Australia, 30 January – 3 February 2012.

Volume 130 - Conceptual Modelling 2012
Contains the proceedings of the Eighth Asia-Pacific Conference on Conceptual Modelling (APCCM 2012), Melbourne, Australia, 31 January – 3 February 2012.

Volume 131 - Advances in Ontologies 2010
Contains the proceedings of the Sixth Australasian Ontology Workshop 2010 (AGOW 2010), Adelaide, Australia, 7th December 2010.