Heng Tao Shen et al. (Eds.)

LNCS 6185

Web-Age Information Management

WAIM 2010 International Workshops: IWGD 2010, XMLDM 2010, WCMT 2010 Jiuzhaigou Valley, China, July 2010 Revised Selected Papers

Springer

Lecture Notes in Computer Science 6185

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Heng Tao Shen Jian Pei
M. Tamer Özsu Lei Zou Jiaheng Lu
Tok-Wang Ling Ge Yu Yi Zhuang
Jie Shao (Eds.)

Web-Age Information Management

WAIM 2010 International Workshops:
IWGD 2010, XMLDM 2010, WCMT 2010
Jiuzhaigou Valley, China, July 15-17, 2010
Revised Selected Papers

Springer

Volume Editors

Heng Tao Shen
The University of Queensland, Brisbane, QLD, Australia
E-mail: shenht@itee.uq.edu.au
Jian Pei
Simon Fraser University, Burnaby BC, Canada,
E-mail: jpei@cs.sfu.ca
M. Tamer Özsu
University of Waterloo, Canada
E-mail: tamer.ozsu@cs.uwaterloo.ca
Lei Zou
Peking University, China
E-mail: zoulei@icst.pku.edu.cn
Jiaheng Lu
Renmin University of China, China
E-mail: jiahenglu@gmail.com
Tok-Wang Ling
National University of Singapore, Singapore
E-mail: lingtw@comp.nus.edu.sg
Ge Yu
North-East University, Shenyang, China
E-mail: yuge@mail.neu.edu.cn
Yi Zhuang
Zhejiang University, Hangzhou, 310058, China
E-mail: zhuang@zjgsu.edu.cn
Jie Shao
University of Melbourne, Australia
E-mail: jsh@unimelb.edu.au

Library of Congress Control Number: 2010937439

CR Subject Classification (1998): H.4, H.3, I.2, C.2, H.5, H.2

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

ISSN 0302-9743
ISBN-10 3-642-16719-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-16719-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

WAIM 2010 Workshop Chair's Message

WAIM is a leading international conference for researchers, practitioners, developers and users to share and exchange cutting-edge ideas, results, experience, techniques and tools in connection with all aspects of Web data management. The conference invites original research and industrial papers on the theory, design and implementation of Web-based information systems, as well as proposals for demonstrations, tutorials and panels. Previous WAIM conferences were held in Shanghai (2000), Xian (2001), Beijing (2002), Chengdu (2003), Dalian (2004), Hangzhou (2005), Hong Kong (2006), Huangshan (2007), Zhangjiajie (2008) and Suzhou (2009). Along with the main conference, WAIM workshops are intended to provide an international group of researchers with forum for the discussion and exchange of research results related to the conference topics.

This WAIM 2010 workshop volume comprises papers from three workshops, which were

1. The First International Workshop on Graph Database
2. International Workshop on Advanced Techniques on XML Data Management, and
3. The Second International Workshop on Web-Based Contents Management Technologies

The contents of these three workshops were selected from a public call-for-proposals process. The workshop organizers put a tremendous amount of effort into soliciting and selecting research papers with a balance of high quality and new ideas and new applications. We asked all workshops to follow a rigid paper selection process, including the procedure to ensure that any Program Committee members (including workshop Program Committee Chairs) be excluded from the paper review process of any papers they are involved with. A requirement about the overall paper acceptance ratio was also imposed to all the workshops.

We are very grateful to the main conference organizers. We would also like to take this opportunity to thank all workshop organizers and Program Committee members for their great effort in putting together the workshop program of WAIM 2010.

May 2010

Heng Tao Shen
Jian Pei

The First International Workshop on Graph Database (IWGD 2010) Chairs' Message

The growing popularity of graph databases has generated interesting data management problems, such as indexing techniques, query algorithms and graph mining. This workshop focused on issues related to graph databases. IWGD 2010 was held in conjunction with WAIM 2010 in JiuZhai Valley, China. IWGD 2010 aimed at bringing together researchers in different fields related to graph databases who have common interests in interdisciplinary research. The workshop provided a forum where researchers and practitioners could share and exchange their knowledge and experience.

April 2010

M. Tamer Özsu
Lei Zou

International Workshop on Advanced Techniques on XML Data Management (XMLDM 2010) Chairs' Message

This was the First International Workshop on XML Data Management. The workshop focused on the convergence of database technology with XML technology, and brought together academics, practitioners, users and vendors to discuss the use and synergy between these technologies.

This workshop attracted 18 submissions from many different schools and countries. The Program Committee accepted 10 papers, among which there are 6 long papers and 4 short papers. The topics of accepted papers include XML applications in semantic Web, XML data storage and indexing, XML query languages and optimization and so on. These proceedings will serve as a valuable reference for XML data management researchers and developers.

The paper "Effective XML Keyword Search Through Valid Lowest Information Unit" mainly discusses concepts and usages of the proposed concepts of LIU and VLIU, which can be used to resolve the problem that LCA-semantics may be incomplete or redundant in the results of keyword search in some cases. Ying Lou et al. first present related work, then present the concept of "information unit" and then introduce the notion of VLIU. Finally, an approach based on VLIU is also outlined in this paper.

The paper "Reducing Redundancy of Xpath Query over Networks by Transmitting XML Views" studies the problem of answering Xpath queries over networks. Xu et al. presented analysis on how to minimize communication cost by rewriting queries. This paper proposes keeping a balance between the network traffic and the complexity of the query, which is a good point.

In their paper "Structure and Content Similarity for Clustering XML Documents," Zhang et al. proposed a new model to represent the structure and content information in XML documents. Based on the model, they define similarity measures that can be used to cluster XML documents.

In the paper "*pq*-Hash: An Efficient Method for Approximate XML Joins," the authors propose a data structure called pq-array to support searching for approximate tree set matching. In addition to that, an efficient algorithm, pq-hash, is proposed to avoid nested loop joins where possible.

In their paper"SeCCX:Semantics-Based Fine Granular Concurrency Control for XML Data," Rong et al. provide a new locking protocol, named SeCCX, for the concurrent transactions of XML data. First, the authors introduce the SeCCX, including its lock modes, lock compatibility matrix, protocol rules and conflict detections and then the authors analyze the isolation of SeCCX.

In the paper "XRCJ: Supporting Keyword Search in XML and Relation Co-occurrence," Zhang et al. propose a new method that combines XML and RDB in the same platform.

In "Building Web Application with XQuery," the authors explore a XQuery-based framework for developing Web applications. This paper proposes that the suggested framework can simplify Web applications development and provides a flexible and scalable architecture.

"Functional Dependencies for XML" studies how to express the functional dependency for XML data based on the current standard XML schema language, i.e. XML Schema. In particular, the authors shows how to support complex values such as list and set, and propose the "match tree" to judge the value equality for complex typed elements, which is further used to decide the satisfaction of the given functional dependency constraints.

In "TwigLinkedList: Improvement of TwigList," the authors introduce the concept of "level list," a linked list, into TwigList to check the parent-child relationship between node and solve the out-of-orderness problem. The paper also conducts several tests to evaluate its performance.

The paper "Compression Algorithms for Structural Query Results on XML Data" provides a compression technique for the results of an XML query. Wang et al. propose to compress XML data in order to reduce XML query-result sizes. At the end, this paper extends bitmap indexes for compression.

These papers cover a variety of topics. We believe that they will provide researchers and developers with a brief glimpse into this exciting new technology, specifically from the perspective of XML databases. We hope you enjoy reading the papers.

April 2010

Jiaheng Lu
Tok Wang Ling
Ge Yu

The Second International Workshop on Web-Based Contents Management Technologies (WCMT 2010) Chairs' Message

The Web is ever-increasing in size and involves quite a broad range of technologies such as databases, XML and digital contents, data and Web mining, Web services, the Semantic Web and ontology, information retrieval, and others. The Second International Workshop on Web-Based Contents Management Technologies (WCMT 2010) was held in conjunction with the WAIM 2010 conference and invited original research contributions on Web and Internet content technologies. WCMT 2010 aimed at bringing together researchers in different fields related to Web information processing who have a common interest in interdisciplinary research. The workshop provided a forum where researchers and practitioners could share and exchange their knowledge and experience.

April 2010

Yi Zhuang
Jie Shao

The First International Workshop on Graph Database (IWGD 2010)

Honorary Chair

Jianguo Xiao	Peking University, China

Program Co-chairs

M. Tamer Özsu	University of Waterloo, Canada
Lei Zou	Peking University, China

Program Committee

K. Singh Ambuj	University of California at Santa Barbara, USA
Gutierrez Claudio	Universidad de Chile, Chile
Zhao Dongyan	Peking University, China
Francis Ilyas Ihab	University of Waterloo, Canada
Cheng James	Nanyang Technological University, Singapore
Xu Yu Jeffery	The Chinese University of Hong Kong, China
Cheng Jiefeng	The University of Hong Kong, China
Zaki Mohammed	Rensselaer Polytechnic Institute, USA
Varadarajan Ramakrishna	University of Wisconsin-Madison, USA
Giugno Rosalba	University of Catania, Italy
Jin Wei	North Dakota State University, USA
Li Wen-Syan	SAP Technology Lab, China
Lian Xiang	Hong Kong University of Science and Technology, China
Wan Xiaojun	Peking University, China
Yan Xifeng	University of California at Santa Barbara, USA
Xiao Yanghua	Fudan University, China
Zhuang Yi	Zhejiang Gongshang University, China
Peng Zhiyong	Wuhan University, China

International Workshop on Advanced Techniques on XML Data Management (XMLDM 2010)

Program Co-chairs

Tok Wang Ling	National University of Singapore, Singapore
Ge Yu	North-East University, China
Jiaheng Lu	Renmin University of China, China

Program Committee

Bao Zhifeng	National University of Singapore, Singapore
Bressan Stephane	National University of Singapore (NUS), Singapore
Boncz Peter	Centrum Wiskunde & Informatica, The Netherlands
Yong Chan Chee	National University of Singapore, Singapore
Du Xiaoyong	Renmin University of China, China
Feng Jianhua	Tsignhua University, China
Gao Jun	Peking University, China
Kitsuregawa Masaru	Tokyo University, Japan
Li Guoliang	Tsinghua University, China
Lin Xuemin	University of New South Wales, Australia
Meng Xiaofeng	Renmin University of China, China
Xu Liang	National University of Singapore, Singapore
Xu Yu Jeffrey	The Chinese University of Hong Kong, China
Vagena Zografoula	Microsoft Research, USA
Wang Bin	North-East University, China
Wang XiaoLing	Fudan University, China
Wang Hongzhi	Harbin Institute of Technology, China
Wood Peter	University of London, UK
Zhou Aoying	East China Normal University, China
Zhou Yongluan	University of Southern Denmark, Denmark
Zhang Rui	University of Melbourne, Australia
Zhang Xiao	Renmin University of China, China

The Second International Workshop on Web-Based Contents Management Technologies (WCMT 2010)

General Co-chairs

Dickson K.W. Chiu	Dickson Computer System, China
Hua Hu	Hangzhou Dianzi University, China

Program Co-chairs

Zhuang Yi	Zhejiang Gongshang University, China
Shao Jie	University of Melbourne, Australia

Program Committee

Yang Jun	Google, USA
Qian Weining	East China Normal University, China
Kalnis Panos	National University of Singapore, Singapore
Li Cuiping	Renmin University, China
Chen Yi	Arizona State University, USA
Wu Weili	University of Texas at Dallas, USA
Wang Wei	University of New South Wales, Australia
Yu Yi	New Jersey Institute of Technology, USA
Xu Jianliang	Hong Kong Baptist University, China
Chi-Wing Wong Raymond	Hong Kong University of Science and Technology, China
Hu Haiyang	Hangzhou Dianzi University, China
Gao Yunjun	Zhejiang University, China
Zou Lei	Peking University, China
Huang Zi	University of Queensland, Australia
Wu Ziang	Nanjing University of Science and Technology, China
He Bingsheng	Microsoft Research Asia, China
Cong Gao	Aalborg University, Denmark
Mehrotra Sharad	University of California, Irvine, USA
Li Feifei	Florida State University, USA
Shen Hong	University of Adelaide, Australia
Choi Byron	Hong Kong Baptist University, China
Wang Wei	University of New South Wales, Australia

Zhang Jia	Northern Illinois University, USA
Jeung Hoyoung	EPFL, Switzerland
Zhou Mingqi	East China Normal University, China
Zhang Jun	Nanyang Technological University, Singapore
He Jing	Victoria University, Australia
Liu Xiaoyan	University of Melbourne, Australia
Zhou Xiangmin	CSIRO, Australia

Table of Contents

The First International Workshop on Graph Database (IWGD 2010)

International Workshop on Advanced Techniques on XML Data Management (XMLDM 2010)

The 2nd International Workshop on Web-Based Contents Management Technologies (WCMT 2010)

Mining Graphs with Constraints on Symmetry and Diameter

Natalia Vanetik

Deutsche Telecom Laboratories and Computer Science department,
Ben-Gurion University, Beer-Sheva, Israel
orlovn@cs.bgu.ac.il

Abstract. The area of graph mining bears great importance when dealing with semi-structured data such as XML, text and chemical and genetic data. One of the main challenges of this field is that out of many resulting frequent subgraphs it is hard to find interesting ones. We propose a novel algorithm that finds subgraphs of limited diameter and high symmetry. These subgraphs represent the more structurally interesting patterns in the database. Our approach also allows to decrease processing time drastically by employing the tree decomposition structure of database graphs during the discovery process.

Keywords: graph mining, symmetry, diameter.

1 Introduction

Graph is an intuitive representation format for many types of data, such as communication networks, social networks, chemical molecules, protein interaction maps, ecological food webs etc. The field of graph mining is rapidly growing area that is applied extensively in bio-informatics, chemical analysis, network analysis, XML indexing and other fields. General graph mining algorithms can be divided into two types. Algorithms of first type employ the Apriori principle of constructing candidate graph patterns from smaller frequent patterns stage by stage and then verifying them by computing support of each pattern. First algorithms of this type suggested were AGM [IWM1] and [KK1]. The second type of algorithms uses a principle of a depth-first traversal of the search space while skipping candidate generation phase in favor of growing patterns in-place. The first algorithm to do so was gSpan [YH1]; closed patterns optimization method of [YH2], list of embeddings methods [BMB1] and pattern automorphism optimization method [JK1] were proposed later. Graph mining algorithms that search for specific types of patterns include tree mining algorithms [Z1],[CYM1],[RK1] and generalized trees [LLZZ1].

The number of frequent subgraphs in the graph database can be exponential in the size of the database and therefore the problem of graph mining lies in PSPACE. Moreover, it often turns out that for high support values too few frequent patterns exist, while for lower support values it is hard to determine which pattern deserves special attention. Several papers (see [WZWWS1], [ZYHY1])

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 1–12, 2010.

deal with mining graphs with constraints, but they do not utilize the symmetry and diameter constraints in the way it is utilized in the current paper. In this work we focus on patterns that on one hand are general enough (i.e. we do not limit ourselves to trees or cliques or otherwise) and on other hand, are structurally interesting. We seek patterns that have symmetry high enough (many automorphisms) and diameter low enough (largest of the shortest paths between two vertices in a pattern). For example, two atoms in a molecule have a non-trivial interaction only if they are not located too remotely one from another, and a group of people within social network is more interesting if the people interact with each other in a non-trivial fashion rather than when they always interact through a common acquaintance. The importance of mining symmetric graphs in social networks is discussed for example in [MSA1], there they claim: "We consider the size and structure of the automorphism groups of a variety of empirical real-world networks and find that, in contrast to classical random graph models, many real-world networks are richly symmetric". Authors of [XWWXW1] show that symmetric subgraphs are shown meaningful in understanding function and mechanism of world trading system.

In this paper we propose an algorithm for mining graphs with constrains on diameter and symmetry. Our algorithm can be used as an extension for any unconstrained graph mining algorithm, such as [YH1, JK1]. The approach we propose relies on preprocessing graphs in the database by constructing their tree decomposition [RS1], a structure that gives great insight into various graph properties, including its symmetry and diameter. Precision of preprocessing depends on the choice of tree decomposition algorithm [B2, B1, BLW1], which can be either exact or approximate. After the initial stage, we cut down the search space used for producing candidate graph patterns by estimating their diameter and symmetry. As a result, we are able to generate all suitable frequent graph patterns in a fraction of time required for full generation.

This paper is organized as follows. Section 2 describes definitions and proofs of tree decomposition properties we use. Section 3 contains the algorithm for mining frequent graph patterns with constraints on symmetry and diameter, proof of algorithm's correctness and discussion on complexity. Section 4 describes results of experimental evaluation.

2 Definitions

2.1 General Definitions

In this paper, we deal with undirected labeled graphs (for unlabeled graphs, we assume all labels equal). In a graph $G = (V, E)$, V denotes the node set, $E \subseteq V \times V$ denotes the edge set and each node $v \in V$ has a label $l(v)$. Given two vertices $u, v \in V$, a *simple path* between u and v is an alternating sequence of nodes and edges $v_0 := v, e_0, v_1, e_1, ..., e_{n-1}, v_n := u$ where $e_i = (v_i, v_{i+1})$ are edges in G and no v_i appears in this sequence more than once. A path between u and v is also called a (u, v)-*path*. The *length of a path* in the number of edges in it, and the *shortest path* between u and v is the path between u and v of the smallest length. *Diameter* $d(G)$ of a graph G is a maximum length of a shortest (u, v)-path over all pairs $u, v \in V$.

For two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, a mapping $\phi : V_1 \to V_2$ is called an *isomorphism* if it is one-to-one and onto and preserves the adjacency relation, i.e. $(v, u) \in E_1 \iff (\phi(v), \phi(u)) \in E_2$. An isomorphism of a graph onto itself is called an *automorphism*. Set of all automorphisms of a given graph forms a group under the operation of composition, called the *automorphism group* of a graph and is denoted $Aut(G)$. *Size* of automorphism group is denoted $|Aut(G)|$ and is bounded by $|V|!$; equality is achieved when $Aut(G)$ is symmetric group on $|V|$ elements. We measure the symmetry $s(G)$ of graph G by the closeness of the size of its automorphism group to the size of symmetric group $S_{|V|}$ as follows.

$$s(G) := \frac{|Aut(G)|}{|S_{|V|}|} = \frac{|Aut(G)|}{|V|!}.$$

A graph $G_1 = (V_1, E_1)$ is called a *subgraph* of $G_2 = (V_2, E_2)$, denoted $G_1 \subseteq G_2$, if there exists a one-to-one mapping $\mu : V_1 \to V_2$ that preserves edges, i.t. $(v, u) \in E_1$ implies $(\mu(v), \mu(u)) \in E_2$.

In our setting, graph database is a set of transactions $D := \{G_1, ..., G_n\}$, and we are given a user-supplied support threshold $supp \in \mathbb{N}$. A subgraph g is called *frequent* if *count* of its appearances in D as a subgraph is at least supp. Count of a graph must satisfy the *downward closure property*: for all subgraphs g_1, g_2 of any database graph G such that $g_1 \subseteq g_2$ we always have $count(g_1) \geq count(g_2)$.

We are also given user-defined symmetry bound symm and maximal diameter diam. Our goal is to find

all frequent subgraphs of D with symmetry $\geq$ symm *and diameter* $\leq$ diam.

2.2 Tree Decomposition of a Graph

Let $G = (V, E)$ be an undirected graph and let $V_T = \{V_1, ..., V_n\}$ be subsets of V covering V, i.e. $\bigcup_i V_i = V$ (note that subsets may intersect).

Definition 1. *V_T is called a tree decomposition if there exists a tree $T = (V_T, E_T)$ where*

1. *If $u \in V_i \cap V_j$ for some i, j, then all nodes of T on the (unique) path from V_i to V_j contain u as well.*
2. *For every edge $(u, v) \in E$ there exists $1 \leq i \leq n$ s.t. $u, v \in V_i$.*

For a given graph, several tree decompositions may exist, for instance, a trivial tree decomposition with $n = 1$, $V_1 = V$ and $E_T = \emptyset$ always exists. The *width* of a tree decomposition is $\max_{i=1,...,n}(|V_i| - 1)$. The *tree width* of a graph is minimal width of its tree decompositions; in this case a tree decomposition is called *minimal*. Figure 1 shows two tree decompositions of the same graph: the first one is not minimal and the second one is.

The concept of tree width was first introduced in [RS1]. Since tree width of a clique of size n is $n - 1$, the tree width is bounded by the size of a largest clique in a graph. The general problem of determining the tree width of a graph and constructing its tree decomposition is NP-hard. For graphs with tree width bounded

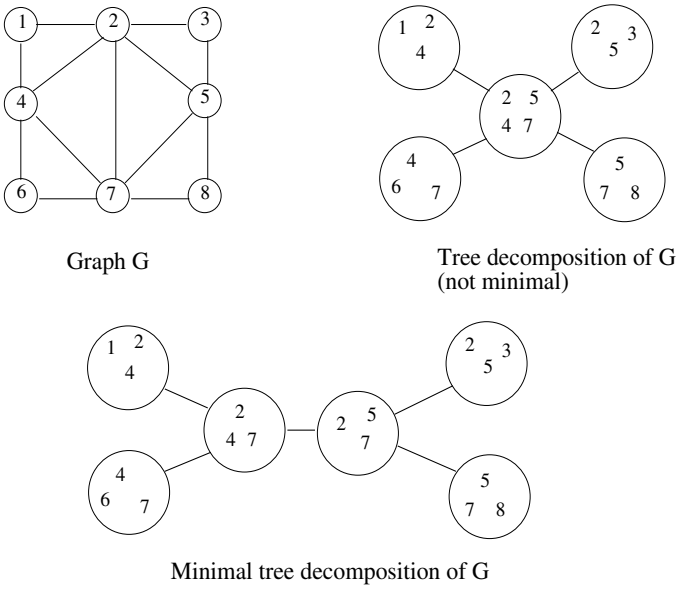


Fig. 1. Tree decompositions of a graph

by a constant, many problems that are otherwise NP-hard have an efficient solution. Families of graphs with bounded tree width include the cactus graphs, pseudoforests, series-parallel graphs, outerplanar graphs, and Halin graphs.

2.3 Properties of Tree Decompositions

Given a tree decomposition $T = (V_T, E_T)$ of a graph $G = (V, E)$, we can deduce some properties of G related to its diameter and symmetry. One should note that graph's diameter and symmetry are connected in the sense that a graph with lower diameter will have higher symmetry, although this dependency is far from trivial. For example, a graph will have symmetry 1 if and only if its diameter is 1, i.e. the graph is a complete graph. From now on, we always assume that the graph G we study is connected. Let $g = (V_g, E_g)$ be a connected subgraph of G.

Definition 2 (cover subtree). *Let $T_g = (V_{gT}, E_{gT})$ be a minimal subtree of T whose nodes cover V_g and contain all edges in E_g.*

Minimality of a subtree is inclusion-minimality of a subtree node set. The notion of cover subtree is well-defined because g is connected and thus the subgraph of T induced by V_g is a subtree of T. Figure 2 shows an example of a graph, its subgraph and minimal tree decomposition and the corresponding cover subtree.

Property 1. Let $g = (V_g, E_g)$ be a subgraph of a graph $G = (V, E)$ with a tree decomposition $T = (V_T, E_T)$. Then $d(g) \geq d(T_g)$.

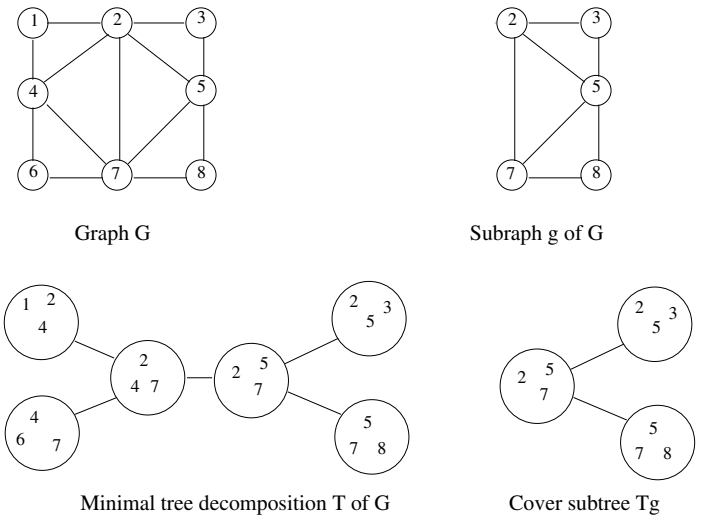


Fig. 2. Cover subtree

Proof. Follows from definition of an cover subtree and tree decomposition: every edge of g is contained within some node of T_g. □

Naturally, the closer T to a minimal tree decomposition, the better the estimate of Property 1 will be. If g is an induced subgraph of G and T is minimal tree decomposition, we would have $d(g) = d(T_g)$ because every intersection of g and a node of T is a clique.

Let us first define and prove a useful bound on the symmetry of a graph given its tree decomposition. We define labeling functions on nodes and edges of T:

$$l_V : V_T \to \mathbb{N} \text{ where } l(v) = |v| \text{ and } l_E : E_T \to \mathbb{N} \text{ where } l(u, v) = |v \cap u|.$$

Labeled automorphisms of T are automorphisms that preserve both label functions defined above. The labeled automorphism group of T is denoted $LAut(T)$. Let us define a bound on symmetry of g.

Definition 3 (symmetry bound). *Let T be a tree decomposition of a graph $G = (V, E)$ and let $g = (V_g, E_g)$ be a subgraph of G. Let $T_g = (V_{gT}, E_{gT})$ be a cover tree of g in T. A symmetry bound of g in T is:*

$$\sigma(g, T_g) := |LAut(T_g)| \times \Pi_{v \in V_{gT}} |(v \setminus E_g) \cap V_g|! \times \Pi_{e \in E_{gT}} |e|!$$

Property 2. $s(g) \leq \sigma(T_g)/|g|!$

Proof. Every automorphism ϕ of g induces a labeled automorphism λ of T_g, since replacing each subset $U \subseteq V$ with $\phi(U)$ preserves the structure of a tree decomposition. An edge of T_g must be mapped to an edge by λ, and node to a node. An edge e of T_g as a set admits at most $|e|!$ permutations when λ is identity,

and a node v admits at most $|v \setminus E_g|!$ permutations. Therefore, $|Aut(g)| \leq \sigma(T_g)$ and the estimate follows. □

Property 3. $\sigma(g, T_g)/|g|!$ is monotonically non-increasing w.r.t adding edges and nodes to g.

Proof. Note that adding an edge to g does not change $\sigma(g, T_g)$. If adding a node x to g gives us a graph g' with $T_{g'} = T_g$, it means that $x \in v \in T_g$ and $\sigma(g', T_g) = \sigma(g, T_g) \times (|v| + 1)$. Since $|v| + 1 \leq |g| + 1 = |g'|$, we have $\sigma(g', T_g)/(|g| + 1)! \leq \sigma(g, T_g)/|g|!$. If $T_{g'} \neq T_g$, $T_{g'}$ contains exactly one node, say u, that is not in T_g. Then $u \setminus E_{g'} = \{x\}$. Adding a node to T_g can increase $|LAut(T_g)|$ by a factor of at most $|g| + 1$, since x can be mapped to at most $|g'| = |g| + 1$ vertices of g'. Then we have $\sigma(g', T_g)/(|g| + 1)! \leq \sigma(g, T_g) \times (|g| + 1) * 1/|g|! \times (|g| + 1) \leq \sigma(g, T_g)/|g|!$ as required. □

Based on the above two properties, we can significantly cut the search space for frequent subgraphs (induced or not) with required symmetry and diameter. We choose the pattern growth approach of gSpan [YH1] in our description and experiments, but one can use any other general graph mining algorithm instead. Search space reduction is then performed at each step of pattern growing by observing the parameters of pre-computed tree decompositions of database transactions. In general, one does not need to compute minimal tree decomposition and may settle for an easily computable but not optimal decomposition instead; this consideration should be made especially when the database is a single large graph. If, however, database consists of multitude of small transactions, it may be better to compute exact tree decomposition since it will decrease the number of generated candidates significantly.

3 Diameter and Symmetry Mining Algorithm

In this section, we present algorithm for mining frequent subgraphs with symmetry and diameter constraints. The search can take advantage of any tree decomposition, and is most efficient when the decomposition is minimal.

For Algorithm 1 to produce frequent subgraphs, we need to define the extension function of a frequent graph pattern appropriately. As a basis, we may use any function that builds next generation of frequent subgraphs by extending frequent subgraphs by adding some edges and/or nodes (for example, minimal DFS extension used in gSpan [YH1]). To determine which extensions are legal, we need to take into account the (minimal) tree decomposition structure of database transaction where the extension takes place. Function Extend() (see Algorithm 2) enlarges a frequent subgraph as long as its optimistic diameter and symmetry estimates are within the required boundaries *for every database transaction*. If candidate diameter and symmetry do not meet these estimates for some transaction, the candidate is immediately removed from the set. Since the estimate is based on cover subtree T_g of T for g, Extend() may produce frequent subgraphs with larger than needed diameter and smaller than needed symmetry. However, these subgraphs are required in order to find all frequent subgraphs we

Algorithm 1. DS-search

Input: Graph database $D = \{G_1, ..., G_n\}$.
Support threshold supp $\in \mathbb{N}$.
Maximal diameter diam.
Minimal symmetry symm.
Output: Frequent subgraphs with diameter $\leq$ diam and symmetry $\geq$ symm.

1. $i \leftarrow 0$;
2. Preprocessing:
   (a) $F_0 \leftarrow$frequent nodes in D.
   (b) Remove all non-frequent nodes from D.
   (c) Build tree decomposition T_i for each transaction G_i.
3. While $F_i \neq \emptyset$ do:
   (a) $F_{i+1} \leftarrow \emptyset$;
   (b) For each subgraph $f \in F_i$ set $F_{i+1} \leftarrow F_{i+1} \cup Extend(g_i, D, \text{diam}, \text{symm}, \text{supp})$.
   (c) $i \leftarrow i + 1$;
   (d) go to step 3.
4. Leave in $\cup F_i$ all subgraphs of symmetry $\geq$ symm and diameter $\leq$ diam only;
5. Output $\cup F_i$;

Algorithm 2. Extend$(f, D, \text{diam}, \text{symm}, \text{supp})$

Input: Frequent subgraph f of size k.
Graph database $D = \{G_1, ..., G_n\}$.
Maximal diameter diam and minimal symmetry symm.
DFS-extend() function.
Support threshold supp $\in \mathbb{N}$.
Output: frequent extensions of f.

1. Extensions $\leftarrow$DFS-extend(f, supp)
2. For each $h \in$ Extensions do:
   (a) For each subgraph $h_i \subset G_i$ isomorphic to h do:
      i. Let T^i be tree decomposition of G_i.
      ii. If $d(T^i_{h_i}) >$ diam or $\frac{\sigma(h_i, T^i_{h_i})}{|V_{h_i}|!} \leq$ symm, remove h from Extensions.
3. Return Extensions.

seek - a small-diameter subgraph can be a supergraph of a subgraph with larger diameter, and a subgraph with high symmetry can be a supergraph of a subgraph with low symmetry. Figure 3 shows how the subgraph extension is done. At first step, g is transformed into g' by adding a node and two edges. Since cover subtree stays the same, so are the estimates for subgraph's diameter and symmetry. At the second step, g'' is formed by adding a node and an edge and the corresponding intersection subtree now consists of two nodes. Now diameter estimate increases and symmetry estimate decreases.

3.1 Correctness Proof

In this section we prove that every frequent subgraph g of G with symmetry $\geq$ symm and diameter $\leq$ diam is produced DS-search algorithm (completeness).

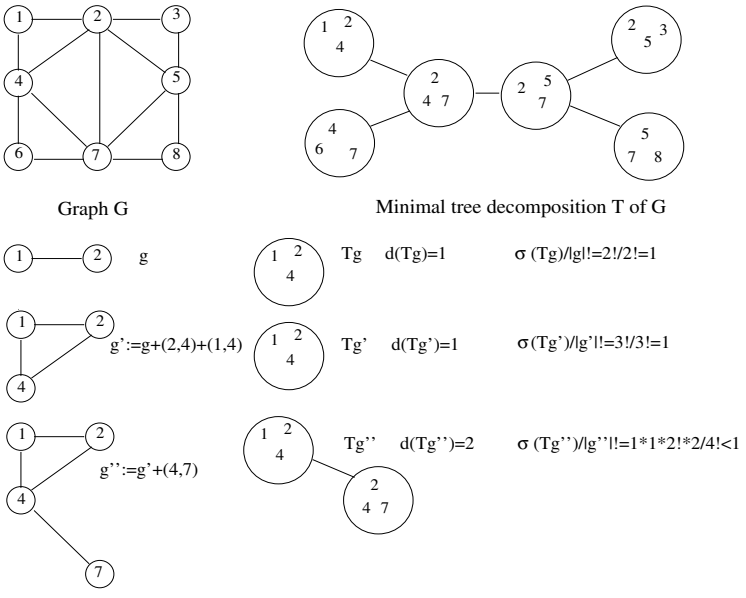


Fig. 3. Two steps of subgraph extension

If completeness is proved, then step 4 of DS-search algorithm ensures that the algorithm is sound, i.e. no frequent subgraph with too big diameter or not enough symmetry appears in the output.

Claim. DS-search algorithm is complete.

Proof. Let g be the smallest frequent subgraph of G with symmetry $s(g) \geq$ symm and diameter $d(g) \geq$ diam not found by the algorithm. Let T^i be a tree decomposition of a transaction $G_i \supset g$ and let $e \in g$ be an edge such that graph $g \setminus e$ is connected (e always exists as g is connected). Then $d(T^i_{g\setminus e}) \leq d(T^i_g)$ because $T^i_{g\setminus e} \subseteq T^i_g$. By Property 3 we have $\sigma(g \setminus e, T^i_{g\setminus e}) \geq \sigma(g, T^i_g) \geq$ symm. Therefore, $g \setminus e$ was found by the algorithm and thus g is kept in Extensions of $g \setminus e$ in step 2(a)ii of the Extend() function. □

3.2 Complexity

The graph mining problem in general lies in PSPACE, and so does its limitation w.r.t diameter and symmetry (take, for instance, the case when all database transactions are complete unlabeled graphs where the search space cannot be successfully decreased by any means). The problem of computing minimal tree decomposition of a graph is NP-hard (see, e.g., [AP1]). However, this task needs to be performed only once and can be done off-line. If one is willing to sacrifice some of the search space reduction, it is enough to compute some tree decomposition, not necessarily the minimal one. Our method is suitable for the case

of transactional graph database rather than for the case of a single large graph dataset (real-life molecular datasets are of this form). When minimal tree decomposition of database transactions is available, some tasks become trivial. For instance, finding all maximal frequent subgraphs of minimal diameter and maximal symmetry (i.e., maximal cliques) can be done by simply counting the nodes of minimal tree decompositions of database transactions. Computing T_g, given a subgraph g and tree decomposition T, is linear in the size of g, and both $d(T_g)$ and $\sigma(g, T_g)$ can be computed during the computation. Labeled automorphisms of a tree are easy to compute, for example by using linear-time and linear-space method of [DIR1].

4 Experimental Evaluation

DS-search algorithm was implemented in C and used gSpan optimization suitable for molecular mining developed by K. Jahn and S. Kramer described in [JK1, JK2] as an underlying graph mining algorithm. Exact tree decompositions of graph transactions were computed off-line with LibTW Java software package [LibTW]. The algorithm was tested on a machine with Intel Xeon 2.60GHz CPU and 3Gb of main memory running Linux OS. We have run the optimized graph mining algorithm of [JK1] without symmetry and diameter constraints for running time comparison. Two different types of databases were tested. The first two sets contain molecular data from the Predictive Toxicology Evaluation Challenge [SKB1]. These sets consists of labeled graphs and contain 300+ transactions with each transaction containing tens of nodes. The next two sets are synthetic graph databases from VFLib graph matching library. Each dataset contains 100 unlabeled graphs. The si4_m2D database contains 2-dimensional meshes and the si4_r001 database contains Erdös-Renyi graphs.

In charts, X axis denotes support in % while the Y axis denotes time that took the algorithm to produce all frequent subgraphs for given support value, in seconds. We use log-scale for the Y axis in order to show the difference in small running times better. We have tested DS-search algorithm with maximal diameter constraint set to 1, 3 and 5 in each case (denoted $D = 1, 3, 5$), and minimum symmetry constraint 0.1, 0.01, 0.001 (denoted $S = 0.1, 0.01, 0.001$) separately. *All* series in charts denotes results of algorithm run without diameter

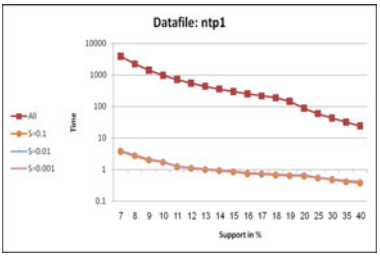


Fig. 4. PTDB 1: symmetry constraints

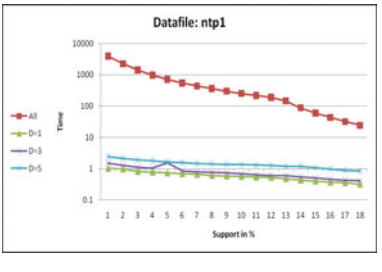


Fig. 5. PTDB 1: diameter constraints

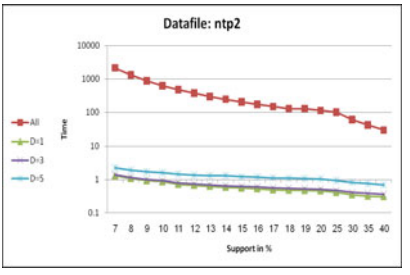


Fig. 6. PTDB 2: symmetry constraints

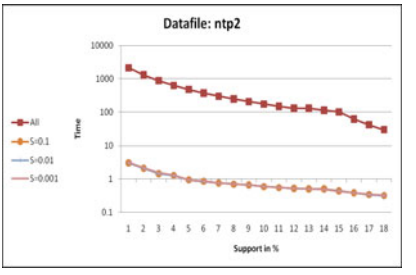


Fig. 7. PTDB 2: diameter constraints

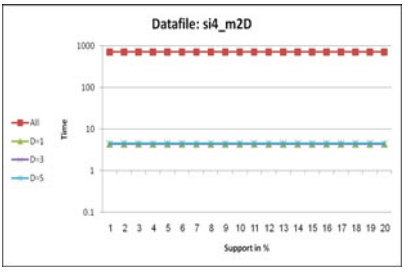


Fig. 8. 2D meshes: symmetry constraints

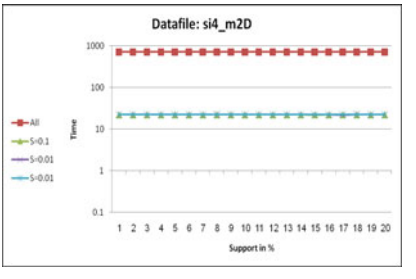


Fig. 9. 2D meshes: diameter constraints

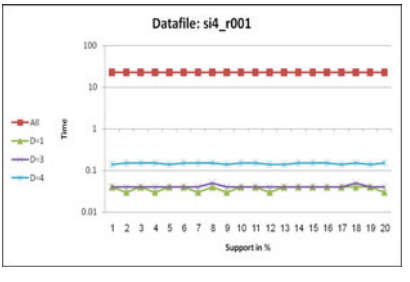


Fig. 10. Erdös-Renyi graphs: symmetry constraints

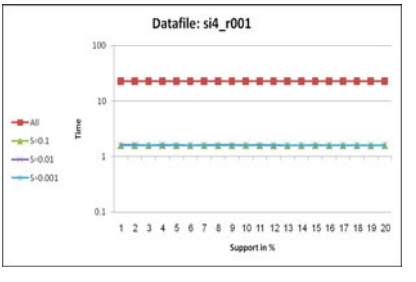


Fig. 11. Erdös-Renyi graphs: diameter constraints

and symmetry constraints. In some cases, especially when testing symmetry constraints, results for different symmetry values are very close (within 0.01 sec) and therefore several series appear on a chart as a single line.

Figures 4 and 5 show how DS-search algorithm performs when diameter or symmetry constraints are set for first Predictive Toxicology Evaluation Challenge molecular database, denoted ntp1. Figures 6 and 7 performance of DS-search for the second Predictive Toxicology Evaluation Challenge molecular database we have used, denoted ntp2. Results testing DS-search on si4_m2D synthetic database are shown in Figures 8 and 9, and results of testing DS-search on si4_r001 database are shown in Figures 10 and 11. Results demonstrate that using

tree decomposition-based estimate method allows to produce frequent subgraphs with required diameter or symmetry much faster.

5 Conclusions

In this paper we presented DS-search algorithm for mining frequent subgraphs of limited diameter and symmetry. Our algorithm can use various graph mining algorithms as a basis, thus taking advantage of any optimizations suggested in the field of unconstrained data mining. DS-search relies on two things: off-line database preprocessing that constructs tree decomposition for every database transaction and search space reduction performed at run-time. Space reduction depends on the nature of tree decomposition: if minimal tree decomposition is chosen, one would spend more time on preprocessing but produce less candidate patterns while approximate tree decomposition will save preprocessing time and give us less efficient search space reduction. We have evaluated our algorithms on two types of databases, real-life molecular data and synthetic graph data and have shown that DS-search algorithm produces frequent subgraphs of required symmetry and diameter much faster than the underlying general approach.

Acknowledgments

The author wishes to express her deepest gratitude to Prof. Ehud Gudes for inspiration, comments and suggestions.

References

[AP1] Arnborg, S., Proskurowski, A.: Linear time algorithms for NP-hard problems restricted to partial k-trees. Discrete Applied Mathematics 23(1), 11–24 (1989); 51–58 (2002)

[BLW1] Bern, M., Lawler, E., Wong, A.: Linear-time computation of optimal subgraphs of decomposable graphs. Journal of Algorithms 8(2), 216–235 (1987)

[B1] Bodlaender, H.L.: Dynamic programming on graphs with bounded treewidth. In: Lepistö, T., Salomaa, A. (eds.) ICALP 1988. LNCS, vol. 317, pp. 105–118. Springer, Heidelberg (1988)

[B2] Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. SIAM Journal on Computing 25(6), 1305–1317 (1996)

[BMB1] Borgelt, C., Meinl, T., Berthold, M.: Advanced pruning strategies to speed up closed molecular fragments. In: Proc. IEEE Conf. on Systems, Man and Cybernetics (SMC 2004), The Hague, Netherlands. IEEE Press, Piscataway (2004)

[CYM1] Chi, Y., Yang, Y., Muntz, R.R.: Indexing and mining free trees. In: Xindong Wu, A.T. (ed.) Proceedings of the 3rd IEEE International Conference on Data Mining. IEEE Computer Society, Los Alamitos (November 2003)

[DIR1] Dinitz, Y., Itai, A., Rodeh, M.: On an Algorithm of Zemlyachenko for Subtree Isomorphism. Inf. Process. Lett. 70(3), 141–146 (1999)

[IWM1] Inokuchi, A., Washio, T., Motoda, H.: An apriori-based algorithm for mining frequent substructures from graph data. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 13–23. Springer, Heidelberg (2000)

[JK1] Jahn, K., Kramer, S.: Optimizing gSpan for Molecular Datasets. In: Proceedings of the Third International Workshop on Mining Graphs, Trees and Sequences 2005, vol. 13(5), pp. 509–523 (2002)

[JK2] Optimized gSpan algorithm for molecular databases implementation, http://wwwkramer.in.tum.de/research/pubs/jahn_mgts05

[KK1] Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: Proceedings of the 2001 IEEE International Conference on Data Mining, pp. 313–320 (2001)

[MSA1] MacArthur, B.D., Snchez-Garca, R.J., Anderson, J.W.: Symmetry in complex networks. Discrete Applied Mathematics 156(18), 3525–3531

[LibTW] LibTW tree decomposition SW library, http://www.treewidth.com

[LLZZ1] Lin, X., Liu, C., Zhang, Y., Zhou, X.: Efficiently Computing Frequent Tree-Like Topology Patterns in a Web Environment. In: Proceedings of 31st Int. Conf. on Tech. of Object-Oriented Language and Systems (1998)

[RS1] Robertson, N., Seymour, P.D.: Graph minors III: Planar tree-width. Journal of Combinatorial Theory, Series B 36, 49–64

[RK1] Rckert, U., Kramer, S.: Frequent Free Tree Discovery in Graph Data. In: Proceedings of the ACM Symposium on Applied Computing, pp. 564–570 (2004)

[SKB1] Srinivasan, A., King, R.D., Bristol, D.W.: An assessment of submissions made to the Predictive Toxicology Evaluation Challenge. In: Proc. 16th Int. Joint Conf. on Artificial Intelligence, IJCAI 1999, Stockholm, Sweden, pp. 270–275. Morgan Kaufmann, San Francisco (1999)

[WZWWS1] Wang, C., Zhu, Y., Wu, T., Wang, W., Shi, B.: Constraint-Based Graph Mining in Large Database. In: Zhang, Y., Tanaka, K., Yu, J.X., Wang, S., Li, M. (eds.) APWeb 2005. LNCS, vol. 3399, pp. 133–144. Springer, Heidelberg (2005)

[XWWXW1] Xiao, Y., Wu, W., Wang, H., Xiong, M., Wang, W.: Symmetry-based Structure Entropy of Complex Networks. Physica A 387, 2611–2619 (2008)

[YH1] Yan, X., Han, J.: gSpan: Graph-Based Substructure Pattern Mining. In: ICDM 2002, pp. 721–724 (2002)

[YH2] Yan, X., Han, J.: Closegraph, Mining closed frequent graph patterns. In: Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2003), Washington DC, USA, pp. 286–295. ACM Press, New York (2003)

[Z1] Zaki, M.J.: Efficiently mining frequent trees in a forest. In: Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM Press, New York (2002)

[ZYHY1] Zhu, F., Yan, X., Han, J., Yu, P.S.: gPrune: A Constraint Pushing Framework for Graph Pattern Mining. In: Zhou, Z.-H., Li, H., Yang, Q. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4426, pp. 388–400. Springer, Heidelberg (2007)

Graph Partitioning Strategies for Efficient BFS in Shared-Nothing Parallel Systems*

Victor Muntés-Mulero, Norbert Martínez-Bazán, Josep-Lluís Larriba-Pey, Esther Pacitti, and Patrick Valduriez

DAMA-UPC, Departament d'Arquitectura de Computadors,
Universitat Politécnica de Catalunya, Campus Nord-UPC 08034 Barcelona
{vmuntes,nmartine,larri}@ac.upc.edu
http://www.dama.upc.edu
INRIA and LIRMM, Montpellier, France
pacitti@lirmm.fr,Patrick.Valduriez@inria.fr

Abstract. Traversing massive graphs as efficiently as possible is essential for many applications. Many common operations on graphs, such as calculating the distance between two nodes, are based on the Breadth First Search traversal. However, because of the exhaustive exploration of all the nodes and edges of the graph, this operation might be very time consuming. A possible solution is distributing the graph among the nodes of a shared-nothing parallel system. Nevertheless, this operation may generate a large amount of inter-node communication. In this paper, we propose two graph partitioning techniques and improve previous distributed versions of BFS in order to reduce this communication.

Keywords: Distributed Graphs, Graph Partitioning, Distributed BFS, Graph Databases.

1 Introduction

The increasing interest for analyzing graph-like information in many areas has reinforced the need for handling massive graphs very efficiently. Analyzing a social network [3] or studying the complex mechanisms inside a living cell [11] are just a couple of examples where the massive storage of graphs and the efficient exploration of large subsets of these graphs is becoming a real challenge. Queries such as finding all genes whose expression is directly or indirectly influenced by a given molecule [12], finding the shortest distance between two members in a social network, or measuring the worst-case communication time of a network interconnecting the nodes of a distributed system, have become necessary for many of these applications. Despite the diversity of scenarios, a comprehensive analysis of the great majority of the queries performed in these areas reveals

* Work partially funded by the Picasso research program between France and Spain, the COLOR VLGDB project at INRIA Sophia-Antipolis Méditerranée, the Ministry of Science and Innovation of Spain (grant numbers TIN2009-14560-C03-03 and JC2009-00305) and Generalitat de Catalunya (grant number GRC-1087).

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 13–24, 2010.

a subset of fundamental common operations upon which most complex queries rely. Among these, the Breath-First Search algorithm (BFS) is widely used in numerous typical structural queries such as finding the shortest distance between two vertices or finding the set of vertices at a certain distance of a given source.

The need for traversing massive graphs has given rise to the developement of efficient graph storage and management techniques, where data is considered to be a large graph containing objects in the form of vertices and edges which, in turn, may contain additional data such as weights or other attributes. Nevertheless, two different limitations are still evident: (i) graphs may not fit in memory and graph traversals may imply an intensive usage of secondary storage, thus degrading performance, and (ii) even if the graph fits in memory, exhaustive and repeated traversals to answer complex queries might be very time consuming. Recent approaches, such as DEX [8], try to favor performance by keeping in memory the most frequently traversed vertices and edges in the graph in a compact way, based on the use of bitmaps. This improves on the classical graph representations (adjacency or incidence matrices) in terms of memory requirements. However, handling such graphs in a single machine is still inefficient and requires exploring solutions beyond current proposals.

Data distribution and query parallelization arises naturally as a possible solution to this problem. In [13], authors propose to arbitrarily partition the graph into several parts and distribute them into the nodes[1] of a shared-nothing parallel system in order to solve BFS more efficiently. The traversal starts from a source vertex and keeps visiting neighbors iteratively, generating communication with other nodes when any of these neighbors is not assigned to the current node. However, these previous attempts to parallelize BFS ignore the possible effect that different partitionings of the graph might have on the amount of inter-node communication. This could unnecessarily degrade the overall performance. This problem becomes critical if we consider queries that need to traverse the graph using BFS repeteadly from multiple source vertices. For instance, computing the maximum communication time between two nodes in a network of computers requires computing all-pairs shortest paths, making an intensive use of operations based on BFS traversals. In these situations, it is easier and even more necessary to exploit parallelization. Note that, since BFS requires traversing the entire graph which does not fit in the single memory of a node, it is not possible to centralize data in a node and solve the query locally. It is necessary to distribute the query over many nodes.

Much work has been devoted to the problem of balanced graph partitioning. Given a graph, the objective is to divide it into k partitions such that each partition contains about the same number of vertices and the number of edges connecting vertices from different partitions is minimum. Even for $k = 2$, this problem is NP-Hard [6] and is also known as finding the minimum edge-separator. Important efforts have been made on developing polynomial-time heuristic algorithms that give good sub-optimal solutions [5,7], and their extensions to find a near-optimum multiple-way separator that splits the graph in k parts [10]. Some

[1] Note that we refer to the entities in the data graph as *vertices* and the machines in the shared-nothing parallel system as *nodes*.

effort is also devoted to the analogous problem of finding minimal size vertex separators, although in general this problem is harder than the corresponding problems on edge separators [1]. Data partitioning strategies have also been deeply studied for other database models. For instance, in ODBMSs, objects could be considered vertices in a graph and the relationships between objects the edges. A comprehensive survey of partitioning techniques for ODBMSs may be found in [9]. Unfortunately, graph databases are subject to queries that are completely different from those in ODBMSs. For instance, structural queries such as finding the shortest distance between two vertices in a very large graph might be crucial in graph databases but irrelevant in ODBMSs, making previous partitioning strategies unsuitable for these new requirements.

In this paper, we focus on reducing the inter-node communication during the parallelized execution of the BFS operation on massive graphs containing up to millions of vertices and edges. For this purpose, we assume graphs to be non-attributed since attributes are not relevant for the BFS operation. We study the effect of different partitioning methods on the overall communication during the execution of this algorithm. For this, we propose two new heuristic partitioning strategies which benefit from the fact that they work on compact collections of vertices and edges, such as those proposed for DEX [8]. We show that our techniques reduce communication from 74% to 99% depending on the dataset, compared to previous proposals.

The remainder of this paper is organized as follows. In Section 2, we present the state-of-the-art and formally define the problem. Section 3 presents our two new partitioning techniques as well as a modification of previous distributed versions of BFS. Section 4 presents some results and, finally, we draw some conclusions and outline our future research.

2 Preliminary Definitions and Previous Work

In order to understand the effect of graph partitioning on the amount of inter-node communication, this section introduces some preliminary definitions and previous work. At the end, as a conclusion of the analysis of this previous work, we formally define the problem that motivates this research.

Let $G = (V, E)$ be an undirected graph where each vertex $v \in V$ is defined by a unique object identifier. We assume the graph to be connected without any loss of generality, since disconnected subgraphs can be handled separately as smaller connected graphs. As proposed in [13], we also assume that G is distributed in a shared-nothing parallel system as in Figure 1. The system contains p independent nodes (different computers containing their own separate memory and disk spaces), connected through a high-speed interconnect.

Let us call $\mathcal{P}(V) = \{V_1, \cdots, V_p\}$ the p subsets into which V is partitioned, such that $V_1 \cup \cdots \cup V_p = V$ and partition V_i is assigned to a node N_i. Ideally, partitions need to be balanced in order to balance the workload in each node. In practice, this restriction is not strict and a certain degree of unbalancing may be admissible.

Definition 1. *We say that a vertex $v \in V_i$ is a* boundary *if it is connected to at least one vertex $u \in V_j$ and $j \neq i$.*

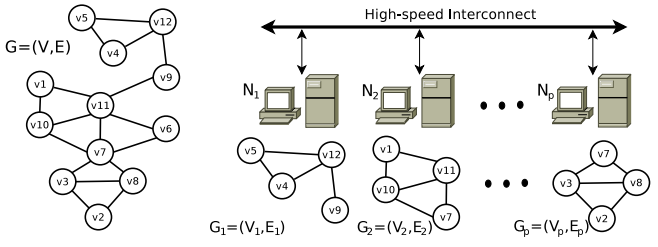


Fig. 1. Graph distribution in a shared-nothing system

Given a graph partitioning, we call B the set of boundary vertices. For instance, $v7$, $v9$ and $v11$ in Figure 1 are boundaries, for this partitioning of G.

2.1 Distributed BFS with 1D and 2D Partitioning

The work in [13] presents and compares two approaches for distributing BFS: BFS with 1D and 2D partitionings, which propose two different ways of partitioning the adjacency matrix of a graph. Note that they can handle huge graphs using this representation only because it is implemented on a BlueGene/L system with a large number of nodes over 30000. Following, we describe the distributed BFS algorithm (DBFS) used for their work.

In DBFS with 1D partitioning (1D), matrix rows are randomly assigned to the p nodes in the system (Figure 2.a). Therefore, each row contains the complete list of neighbors of the corresponding vertex. DBFS traverses all the vertices in the graph ordered by their distance to a source vertex v_s. Because of this, the algorithm is synchronized by levels, i.e. nodes have to be synchronized after traversing the vertices at a certain distance to v_s, in order to guarantee that vertices that are closer to the source vertex are visited first. DBFS proceeds as follows (see Algorithm 1). Each node N_i executes the same code on its partition V_i and receives as a parameter the source vertex v_s. Given a node N_i, three subsets of vertices are kept per node: the subset of visited vertices $V_{visited} \in V_i$, the vertices to be visited during the current iteration $V_{current} \in V_i$ (i.e. vertices located at the same minimum distance from the source vertex in the local node), and the vertices to be visited in the next iteration $V_{next} \in V_i$. These subsets are initialized at lines 2-5. For the sake of clarity, we have rewritten the algorithm presented in [13], preserving the same functionality, so as it is easier to explain the improvements of our techniques and compare them with previous work.

The outer loop is executed until there are not more vertices to exploit in any of the nodes in the system (line 6). For each vertex $v \in V_{current}$, its set of neighbor vertices $\mathcal{N}_v$ is found. For each vertex $u \in \mathcal{N}_v$, if u is assigned to the local node and has not been previously visited, it is added to V_{next} (line 12), otherwise the vertex identifier is sent to N_x, being $u \in N_x$ (lines 9-11). Authors, implicitely assume that each node has the information on which vertices have been assigned to which partitions. Once all the vertices in $V_{current}$ have been processed, if the node receives vertex identifiers from other nodes (line 14), these are also

Algorithm 1. Distributed BFS (DBFS) with 1D partitioning

```
1:  procedure DBFS(v_s)                                   ▷ Executed in each node N_i
2:      V_visited = ∅; V_next = ∅;                        ▷ Subsets of vertices are initiallized
3:      if v_s ∈ V_i then V_current = {v_s};
4:      else V_current = ∅;
5:      end if
6:      while at least 1 node has V_current ≠ ∅ do
7:          for all v ∈ V_current do
8:              N_v = neighbors(v) \ (V_visited ∪ V_current);   ▷ Neighbors of v not found yet
9:              for all u ∈ N_v ∩ B do                    ▷ Boundaries are sent to their nodes
10:                 send u to N_x such that u ∈ N_x;
11:             end for
12:             V_visited = V_visited ∪ {v}; V_next = V_next ∪ (N_v ∩ V_i);
13:         end for
14:         for all u ∈ B sent by other nodes N_x do
15:             receive u from N_x;          ▷ Receive boundaries assigned to the current node
16:             V_next = V_next ∪ {u} \ (V_visited ∪ V_current);
17:         end for
18:         wait until all nodes have processed their corresponding V_current;
19:         V_current = V_next; V_next = ∅;
20:     end while
21: end procedure
```

added to V_{next}, in order to be processed during the next iteration. Otherwise, the node waits until all nodes in the system have finished processing the vertices corresponding to the current distance (line 18).

Note that there exist several possibilities in terms of sending data. On one side, a single message can be sent immediately for each vertex to be sent to other nodes. On the opposite side, communication can be deferred until all nodes in $V_{current}$ have been processed in order to send a single (probably very large) message containing the vertex identifiers of all the vertices to be transmitted. In this paper, we ignore this implementation issue by counting the number of vertex identifiers sent through the interconnect during the execution, which is always the same independently on how they are sent to other nodes.

In DBFS with 2D partitioning (2D), the idea is to benefit from the fact that nodes are organized in $R \times C = p$ processor meshes in the architecture of the system that they are using, i.e. in a two-dimensional grid such that each node is connected to its four immediate neighbors. The adjacency matrix is divided into $R \cdot C$ blocks of rows and C blocks of columns $\{\mathrm{CB}_1, \cdots, \mathrm{CB}_C\}$ such that each node is assigned those values in the adjacency matrix corresponding to the intersection between C blocks of rows and a block of columns, as shown in Figure 2.b. Therefore, the list of adjacencies of a given vertex is divided into R nodes, and the set of vertices in the graph is assigned to C groups of R nodes. In order to explode a vertex v from one node in CB_i, a message is sent to the other $R-1$ nodes in CB_i to get the complete list of adjacencies of that vertex and each of these nodes communicate with one of the other $C-1$ nodes in the same row of the mesh, if the adjacent vertices of v belong to another CB_j. Thanks to this, the number of nodes to which a certain node sends data can be reduced drastically, since each node only communicates with at most $R+C-2$ instead of p nodes as in the case of $1D$ partitioning. We do not explicitely present the

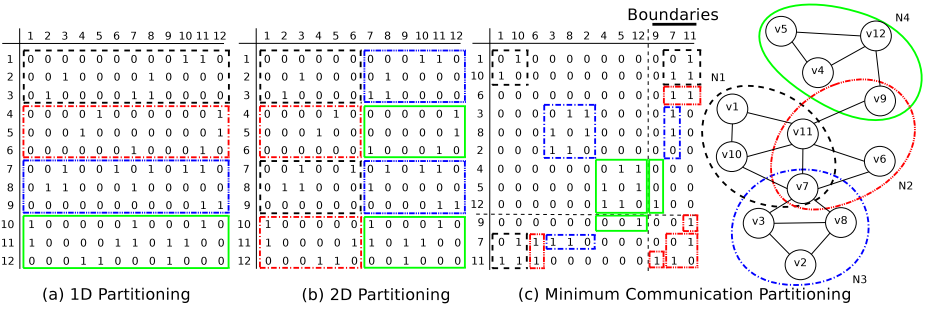


Fig. 2. 1D, 2D and minimum-φ partitionings into 4 partitions

adaption of Algorithm 1 to this new scenario due to a lack of space. The details on this algorithm may be found in [13].

2.2 Problem Definition

As we have seen in this section, previous work has largely ignored the problem of minimizing the amount of information sent through the interconnect. In this subsection, we formally define this problem.

Definition 2. *Given a set of nodes $N = \{N_1, \cdots, N_p\}$ in a shared-nothing parallel system, a graph $G = (V, E)$ and a partitioning $\mathcal{P}(V) = \{V_1, \cdots, V_p\}$ on V, such that partition V_i is assigned to node N_i, we define the* inter-node communication cost function of one BFS execution *$f(i, j)$ as the function that computes the total number of vertex identifiers transmitted from node N_i to node N_j during the execution of a BFS traversal. We define the* communication cost function of a BFS execution *as $\varphi(N, G, \mathcal{P}(V)) = \sum_{i=1}^{p} \sum_{j=1}^{p} f(i, j)$.*

Note that $f(i, j) \neq f(j, i)$, $f(i, i) = 0$. For short, we will use φ instead of $\varphi(N, G, \mathcal{P}(V))$ henceforth. Let us call $en(v) = \{u | (u, v) \in E \wedge u \in V_i \wedge v \in V_j \wedge i \neq j\}$ the set of neighbors located in other partitions. From Algorithm 1, we define $\varphi_{1D} = \sum_{v \in B} |en(v)|$. Let us call $en_{CB}(v) = \{u | (u, v) \in E \wedge u \in \mathrm{CB}_i \wedge v \in \mathrm{CB}_j \wedge i \neq j\}$. For the 2D approach, we define $\varphi_{2D} = |V| \cdot R + \sum_{v \in B} |en_{CB}(v)|$. Note that both functions depend heavily on the number of boundaries $|B|$.

Problem Definition. *Given a set of nodes $N = \{N_1, \cdots, N_p\}$ in a shared-nothing parallel system and a graph $G = (V, E)$, we define the problem of* finding the minimum inter-node communication during a BFS traversal *as the problem of finding a balanced partitioning $\mathcal{P}(V)$ such that $\varphi(N, G, \mathcal{P}(V))$ is minimum.*

3 Partitioning for Reducing Inter-node Communication

In this section, we propose two new partitioning techniques. Additionally, we analyze DBFS and propose a set of modifications. Both contributions aim at reducing inter-node communication during the execution of a BFS traversal.

3.1 Heuristics for Communication Reduction

From previous work, we learn two important lessons. First, reducing the number of boundary vertices may reduce the amount of vertices sent through the interconnect during the execution of DBFS. And second, it is well-known that the number of vertices in a minimum vertex-separator is always smaller or equal to the number of boundary vertices created by choosing the minimum edge-separator. Since we are interested in reducing the amount of boundaries, in this section we propose two techniques that try to find a vertex-separator that divides the graph into k partitions and minimize φ. Each vertex in this set will be assigned to all the partitions containing at least one of its neighbors. This problem differs from previous research on graph partitioning since we are not looking for a minimum edge- or vertex-separator, but for a vertex-separator that minimizes φ. Note that this requires a small modification of Algorithm 1 by changing line 10 to "send u to all nodes N_x such that $u \in N_x$" since each vertex identifier u has to be sent now to as many nodes where u is assigned. Also, we reduce the amount of inter-node communication by generating partitions that are as compact as possible, thus favoring that a large number of vertices are only connected to vertices assigned to the same node and, therefore, reducing the number of vertices assigned to multiple partitions and diminishing communication between nodes.

From these initial premises, we can deduce that, during the execution of DBFS, each node will send a vertex identifier to other nodes each time it finds a boundary vertex, which is assigned to at least another partition. Let us call $p(v)$ the set of partitions where a boundary vertex v is assigned. Thus, $p(v) \equiv \{V_i | v \in V_i\}$. The approximate number of vertex communicated in DBFS for this partitioning is $\varphi \approx \sum_{v \in B}(|p(v)| - 1)$ (where B is the set of boundary vertices).

Figure 2.c shows the optimal balanced partitioning of the graph in our example. We also show the associated adjacency matrix. Note that we have reordered rows and columns symmetrically in order to clarify the example. In our case, we break the rationale of sharing rows and columns of the adjacency matrix, but we are more selective on the information that we take from such matrix. We do that by considering the use of compact data structures such as those presented in [8]. This is an important difference with previous approaches which keep the whole adjacency matrix.

Each edge (v, u) is only kept in the partition that contains v and u. Note that if $v, u \in B$ they might be contained together in more than one partition. In this case it is not necessary to duplicate edges but just assign them to one of these partitions. For instance, in Figure 2.c, boundary vertices $v7$ and $v11$ are assigned to both $N1$ and $N2$. However, the edge between $v7$ and $v11$ has been assigned to $N2$ following the criteria of keeping partitions as balanced as possible in terms of nodes and edges. Just as an example, starting DBFS from node $v4$ generates 26 vertex identifiers through the interconnect for 1D, 54 for 2D and 4 for the minimum-φ partitioning. In order to find this minimum φ-partitioning, we propose the following two heuristic alternatives.

The greedy approach (GA). Our first proposal is called Greedy Approach heuristic. Similar approaches for obtaining quasi-minimum edge-bisectors can be

found in the literature as the Min-Max Greedy algorithm [2]. We extend the main ideas of that work in order to adapt it to the communication cost function (Definition 2) and allow it for a k-way partitioning. The basic idea of this algorithm is to take a different seed vertex for each of the k partitions and repeteadly adding a new neighbor vertex to each partition such that it produces the minimum possible increase $\Delta\varphi$. When the next vertex v to include in partition V_i also belongs to one or more other partitions, it is considered a boundary vertex and then: (i) the neighbors of v are excluded from election for partition V_i and (ii) the neighbors of v in V_i are removed from the list of neighbors of the first partition V_j that included v. In other words, only the first partition to include a vertex can continue traversing its neighbors. The following partitions will just consider this node to be a boundary and will not continue the exploration. Since the first partition traverses the neighbors, this guarantees that each vertex will eventually be included in the list of neighbors of some partition and therefore that all vertices will be included in at least one partition. Also, it may happen that one or more partitions make a third partition starve, i.e. the list of neighbors of this partition becomes empty. In this situation, we choose a vertex that has not been included yet in any partition and continue the exploration. Note that this may generate disconnected subgraphs in a partition.

The enriched greedy approach (EGA). GA is aimed at optimizing φ. However, in particular at the beginning of the execution, there are many vertices whose addition produces the minimum possible increase $\Delta\varphi = 0$. Therefore, we propose to refine this heuristic by adding vertices that not only minimize $\Delta\varphi$ first, but also minimize a second function f. Given a vertex v and a partition V_i, we define this function as $f(v, V_i) = \frac{|\{(v,u)\in E | u \notin V_i\}|}{\delta_v}$ where δ_v is the degree of v. The rationale behind function f is that vertices that are more linked to the current partition must be included first. This will probably benefit the inclusion in the partition of vertices that are uniquely connected to vertices in the same partition, reducing the number of boundaries. Thus, when two or more vertices produce the same $\Delta\varphi$, those among them with lower Δf are preferred.

Note that, these two algorithms do not guarantee partitions to be completely balanced. As mentioned before, although balanced partitions are desirable, we will admit a certain degree of unbalancing. In general, by partitioning the graph in this way, we are favouring locality increasing the probability of finding neighbors in the same node.

3.2 φ-Aware DBFS

A detailed study of DBFS reveals that it performs an important number of unnecessary communications. Each partition V_i does only keep information about the nodes assigned to V_i, but never about the vertices in other partitions connected to vertices in V_i. This means that two vertices $v_1, v_2 \in V_i$ connected to another vertex $u \in V_j$ where $i \neq j$, imply sending the vertex identifier of u twice from V_i to V_j. This is unnecessary, since once u has been visited in a DBFS execution it must not be visited again. In order to solve this problem, we propose a simple yet effective modification of the algorithm to reduce the number of vertex identifiers

Table 1. Largest Connected Component Information

dataset	vertices	edges	size(MB)	max degree
us-road-ny	264346	366923	13.3	8
scopus-authors	6972959	59857403	1481.3	8821
imdb	1965243	207096281	4555.0	62599

communicated between nodes. We propose to keep information about the vertices in other partitions connected to vertices in the current node. For this, we modify line 12 of Algorithm 1 and substitute it by the expression "$V_{visited} = V_{visited} \cap \{v\}$; $V_{next} = V_{next} \cup \mathcal{N}_v$". With this, we eliminate redundancy in the number of vertex identifiers sent through the interconnect. We call the improved versions of 1D and 2D, 1D-I and 2D-I, respectively.

4 Experimental Results

Now, we present the results of the experiments that show the impact of our partitioning techniques on the number of vertex identifiers sent through the interconnect. We first describe the datasets used for these experiments. Second, we study the reduction in the number of vertex identifiers sent during a BFS traversal from a randomly taken source. Finally, we comment on the workload of each node during the execution of the BFS. We run our experiments considering different number of nodes ranging from 2 to 64.

4.1 Datasets Description

We use three different real datasets to perform the experiments. We call the first dataset *USRoadNY* ($|V| = 264346$, $|E| = 366923$), extracted from [4]. It represents the road network of NY state, where each vertex is a crossroad and edges are roads. This graph is quasi-plannar. Second, we use *Scopus* ($|V| = 8307126$, $|E| = 61320009$), which has been provided by Scopus (www.scopus.org). It contains all the authors worldwide and their collaborations in the area of computer science, plus all the Spanish authors and their collaborations in any knowledge area. Vertices are authors and edges represent that two authors have a publication together. Finally, the largest dataset is called *IMDB* ($|V| = 1977972$, $|E| = 207128488$) and it has been obtained from the Internet Movie Database (www.imdb.com). Each vertex is an actor and an edge between two actors indicates that they have collaborated together in a film. Table 1 summarizes the information of their largest connected component.

4.2 Inter-node Communication Reduction

Now, we compare the number of vertex identifiers sent through the interconnect when computing DBFS and its variants from a randomly chosen source vertex, for our two proposals, GA and EGA, and also for 1D and its improved version 1D-I. Figure 3 shows the results for our techniques and those based on 1D using

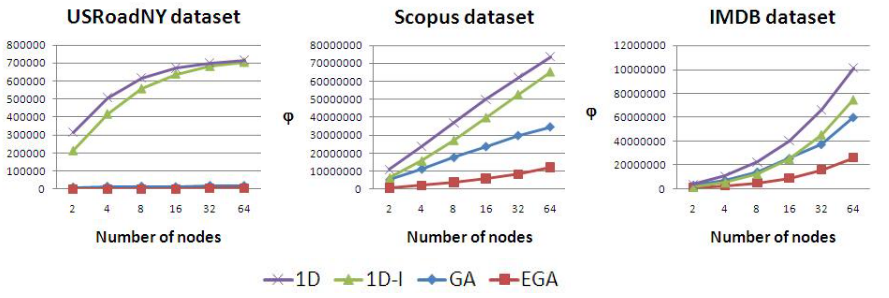


Fig. 3. Number of vertex identifiers sent through the interconnect for different datasets

Table 2. Number of vertex identifiers sent (φ) for 2D-I

	Processor Mesh Configuration [p (RxC)]					
Dataset	**4 (2x2)**	**8 (4x2)**	**8 (2x4)**	**16 (8x2)**	**16 (4x4)**	**16 (2x8)**
USRoadNY	479060	1007752	683868	2065136	1212560	823830
Scopus	13366312	27312230	22751867	55204066	36697785	34247275
IMDB	3920677	7851163	7730897	15712135	11661383	14855946

the three datasets presented for different number of processors. Results for 2D-I are presented in Table 2. Since there might be many different configurations for the processor mesh, we show a sample of them. However, results can be directly compared with those in the figure. For the sake of space we ommit results for 2D, since those for 2D-I are always better.

From these results, we observe that the improvements introduced by the φ-aware DBFS reduce the number of vertices sent from 5% up to 25% depending on the dataset (1D vs 1D-I). Also, we observe that our two techniques GA and EGA outperform previous approaches in terms of reduction of φ. In particular, EGA achieves significant reductions compared to the other approaches thanks to the fact that it takes into consideration a second function f, besides φ, to guide the construction of compact partitions, as explained in Subsection 3.1. Compared to 1D it obtain benefits that range from 74% to 99% depending on the dataset, for 64 nodes. Benefits vary for different types of graph. In particular, with plannar or quasi-plannar graphs (USRoadNY), the amount of communications can be minimized drastically even using GA (improvements ranging from 40% to 97% compared to 1D). Note also that our techniques preserve the gain ratio with other approaches independently on the number of nodes into which the graph is partitioned, making them more scalable. Finally, we can see that the number of vertex identifiers sent in 2D-I is in general similar to that in 1D-I. This make sense since 2D was devised to reduce the number of nodes to which one node sends vertex identifiers, but not the quantity of identifiers.

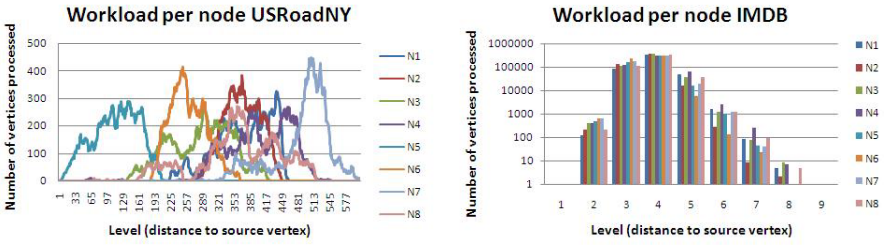


Fig. 4. Number of vertex processed per node for USRoadNY and IMDB (8 partitions)

4.3 Workload Distribution

Finally, we study the workload distribution in the different nodes during the execution of DBFS. We study the size of the subset of vertices $V_{current}$ for each node and each level, which represent the number of vertices processed at each iteration during the traversal. We run the experiments using USRoadNY and IMDB, since the former represents the most sparse graph among those in our datasets and it is quasi-plannar, and the latter is denser.

Figure 4 shows results after running DBFS using EGA and fixing the number of partitions to 8. First, note that the number of levels in which DBFS is solved is different depending on the dataset. This is caused by the fact that USRoadNY is quasi-plannar and its max degree is very low (equal to 8) and, therefore the diameter of the graph is very large. On the other side, IMDB is more dense and the maximum degree is very large, and therefore the diameter necessarily shrinks. Because of this, we have chosen two different types of plot to present results, one for each dataset. We can observe that load balancing depends on the graph topology. While the workload is balanced for graphs such as that for the IMDB dataset, it unbalances for plannar or quasi-plannar graphs with large diameters (USRoadNY). Fortunately, most graphs in all the areas mentioned above, such as social networks, are not plannar and usually have small diameters.

5 Conclusions and Future Work

Distributing a graph into several nodes to speed-up the traversal of vertices is crucial. We have shown that it is not only possibly to improve previous proposals for distributing BFS, but also that graph partitioning is essential for reducing the number of vertex identifiers that have to be sent between nodes, up to two orders of magnitude in the best cases. With this work, we open an important list of questions for future research, ranging from the impact on using other graph partitioning algorithms such as multiple-level graph partitioning algorithms and other improvements to extending these results to weighted, labelled and/or attributed multigraphs.

References

1. Ashcraft, C., Liu, J.W.H.: Using domain decomposition to find graph bisectors. BIT Numerical Mathematics 37(3), 506–534 (1997)
2. Battiti, R., Bertossi, A.A.: Greedy, prohibition, and reactive heuristics for graph partitioning. IEEE Trans. Comput. 48(4), 361–385 (1999)
3. Chakrabarti, D., Faloutsos, C.: Graph mining: Laws, generators, and algorithms. ACM Computing Surveys (CSUR) 38(1), 2 (2006)
4. Demetrescu, C., Goldberg, A.V., Johnson, D.S.: 9th dimacs challenge 9 benchmark platform, version 1.1(october 30, 2006), http://www.dis.uniroma1.it/~challenge9
5. Fiduccia, C.M., Mattheyses, R.M.: A linear-time heuristic for improving network partitions. In: DAC 1982: Proceedings of the 19th Design Automation Conference, pp. 175–181. IEEE Press, Piscataway (1982)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
7. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. The Bell system technical journal 49(1), 291–307 (1970)
8. Martínez-Bazan, N., Muntés-Mulero, V., Gómez-Villamor, S., Nin, J., Sánchez-Martínez, M.-A., Larriba-Pey, J.-L.: Dex: high-performance exploration on large graphs for information retrieval. In: CIKM 2007: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, pp. 573–582. ACM, New York (2007)
9. Özsu, M.T., Valduriez, P.: Principles of distributed database systems, 2nd edn. Prentice-Hall, Inc., Upper Saddle River (1999)
10. Sanchis, L.A.: Multiple-way network partitioning. IEEE Trans. Comput. 38(1), 62–81 (1989)
11. Trissl, S., Leser, U.: Fast and practical indexing and querying of very large graphs. In: SIGMOD 2007: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, pp. 845–856. ACM, New York (2007)
12. van Helden, J., Naim, A., Mancuso, R., Eldridge, M., Wernisch, L., Gilbert, D., Wodak, S.: Representing and analysing molecular and cellular function using the computer. Biological Chemistry 381(i9–10), 921–935 (2000)
13. Yoo, A., Chow, E., Henderson, K., McLendon, W., Hendrickson, B., Catalyurek, U.: A scalable distributed parallel breadth-first search algorithm on bluegene/l. In: SC 2005: Proceedings of the, ACM/IEEE conference on Supercomputing, p. 25. IEEE Computer Society, Washington (2005)

HyperGraphDB: A Generalized Graph Database

Borislav Iordanov

Kobrix Software, Inc.
http://www.kobrix.com

Abstract. We present HyperGraphDB, a novel graph database based on generalized hypergraphs where hyperedges can contain other hyperedges. This generalization automatically reifies every entity expressed in the database thus removing many of the usual difficulties in dealing with higher-order relationships. An open two-layered architecture of the data organization yields a highly customizable system where specific domain representations can be optimized while remaining within a uniform conceptual framework. HyperGraphDB is an embedded, transactional database designed as a universal data model for highly complex, large scale knowledge representation applications such as found in artificial intelligence, bioinformatics and natural language processing.

Keywords: hypergraph, database, knowledge representation, semantic web, distributed.

1 Introduction

While never reaching widespread industry acceptance, there has been an extensive body of work on graph databases, much of it in the 90s. Various data models were proposed, frequently coupled with a complex object representation as a natural, practical application of graph storage. More recently, several developments have contributed to a renewed interest in graph databases: large-scale networks research, social networks, bioinformatics as well as the semantic web and related standards. Part of that interest is due to the massive amounts of graph-oriented data (e.g. social networks) and part of it to the inherent complexity of the information that needs to be represented (semantics and knowledge management). This body of work has been thoroughly reviewed in [1]. In this paper, we present the implementation of a generalized hypergraph model independently proposed by Harold Boley [4] and Ben Goertzel [5]. The model allows for n-ary hyperedges that can also point to other edges, and we add an extensible type tower [6] to the mix. Two generalizations of graphs related to our work have been the Hypernode model [2] and the GROOVY model [3], both focused specifically on representing objects and object schemas. While hypergraphs have been extensively used as an analytical tool in database research, we are not aware of any other implementation of general hypergraphs as a native database.

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 25–36, 2010.

The main contributions of HyperGraphDB[1] lies in the power of its structure-rich, reflexive data model, the dynamic schema enforced by an extensible type system, and open storage architecture allowing domain specific optimizations. It must be noted however that the representational flexibility also leads to performance gains when fully exploited. While a hypergraph can be represented as a regular graph, frequently the opposite is also true in a non-trivial way. Many graphs will have repetitive structural patterns stemming from the restrictions of the classical graph model. Such patterns can be abstracted via a hypergraph resentation, leading to much fewer nodes and database operations. For example, flow graphs where edges represent multi-way input-output connections can be stored much more compactly using a hypergraph-based model.

Furthermore, reducing the complexity of a representation is not the only benefit of a hypergraph model. As illustrated in the context of cellular networks, "transformation to a graph representation is usually possible but may imply a loss of information that can lead to wrong interpretations afterward" ([7]). In fact, biological networks are replete with multilateral relationships that find a direct expression as hypergraphs. Another example of how the ability to represent higher-order relations can improve algorithms can be found in [8], where the authors present a learning algorithm for Markov Logic Networks based on what they call "hypergraph lifting" which amounts to working on higher-order relations.

One common criticism of RDF ([9]) stores is the limited expressiveness of binary predicates, a problem solved by HyperGraphDB's n-ary relationships. Two other prominent issues are contextuality (scoping) and reification. A popular solution of the scoping problem has been proposed in the form of *Named Graphs* ([11]). Reification is represented through a standardized reification vocabulary, by transforming an RDF graph into a reified form ([10]). In this transformation a single triplet yields 4 triplets, which is unnatural, breaks algorithms relying on the original representation, and suffers from both time and space inefficiencies. A similar example comes from the Topic Maps standard [12] where reification must be explicitly added as well, albeit without the need to modify the original representation. Those and other considerations from semantic web research disappear or find natural solutions in the model implemented by HyperGraphDB.

The paper is organized as follows: first, we review some variations of hypergraphs and describe the particular one adopted by HyperGraphDB. In subsequent sections, we detail the system's architecture: storage model, typing, indexing, querying and its P2P distribution framework. Lastly, we describe in some detail one particular application in natural language processing.

2 Hypergraphs and the HyperGraphDB Model

The standard mathematical definition of a hypergraph is a family of sets over a universal set of vertices V, or equivalently an undirected graph where an edge

[1] The system is open-source software, freely available at `http://code.google.com/p/hypergraphdb`.While the current implementation is in the Java programming language, we note that the architecture is host-language-agnostic and we make very few references to Java constructs in the exposition below.

can connect any number (> 0) of vertices. Such hypergraphs are studied in the context of set combinatorics, where they are also called finite set systems [13]. In representational problems directed hypergraphs have also been proposed, where a hyperedge acquires an orientation in one of two ways: (1) partitioning it into a head and a tail set yields several interesting applications as reported in [14]; or (2) treating the edge as a tuple (ordered, with repetition) yields a very powerful representational language [16].

In basic set theory a hypergraph essentially defines an incidence structure over the universe of vertices V. Such a hypergraph is isomorphic to a bipartite graph where one set represents the hypergraph's vertices and the other its hyperedges. If one includes hyperedges in the vertex universe as well, a set theoretic formalization becomes harder because the foundation axiom no longer holds. Such hypergraphs are isomorphic to general (i.e. allowing cycles) directed graphs and they have been studied from a set-theoretic perspective by P. Azel [17]. But in a computational setting such generalized hypergraphs are a much more natural construct as they directly allow the recursive construction of logical relationships while easily avoiding dangerous circularity in a manner similar to Russel's type theory. This is the model adopted by HyperGraphDB. The representational power of higher-order n-ary relationships is the main motivation behind its development.

In the HyperGraphDB data model, the basic representational unit is called an *atom*. Each atom has an associated tuple of atoms called its *target set*. The size of the target set is called the atom's *arity*. Atoms of arity 0 are called *nodes* and atoms of arity > 0 are called *links*. The *incidence set* of an atom x is the set of atoms that have x as a member of their target set (i.e. the set of links pointing to x). The use of tuples instead of sets for an atom's target set is not the only possible choice; both can be supported, but we have focused on the former as the most practical by far.

Moreover, each atom has an associated strongly typed *value*. Values are of arbitrary *types*, and types themselves are atoms (see below for a detailed discussion of the type system). Atoms and values represent two orthogonal aspects of the information content stored in a HyperGraphDB instance. Atoms are the semantic entities that form the hypergraph structure, while values are typed data that may be structured or not. The value of an atom can be changed or replaced with a value of a different type while preserving an identical linkage structure. Several atoms may share the same value. On the other hand, values are immutable entities.[2]

Note that this model does not need to impose any particular semantics on the data being stored. It is a semi-structured, general purpose model incorporating graph, relational, and object-oriented aspects. The HyperGraphDB database schema is a type system that can evolve dynamically, where data integrity constraints can be enforced by type implementations. As exemplified by RDF, such a

[2] One could directly overwrite a value in storage or change its runtime representation and HyperGraphDB has no control over that, but there are no means in the API to modify a value.

flexible architecture is called for on the open web, where fixed database schemas can easily break.

3 Two-Layered Architecture

A key aspect of HyperGraphDB's architecture are the two levels of data organization which we now describe. We note that the actual physical storage is mostly irrelevant with the notable requirement that an efficient key-value indexing must be available, such as the BerkeleyDB storage system [18] which is currently being used.

The two-layered architecture defines a primitive hypergraph storage abstraction, the primitive storage layer and a model layer. The primitive storage layer is partitioned into two associative arrays (i.e. key-value stores):

$LinkStore : ID \rightarrow List < ID >$
$DataStore : ID \rightarrow List < byte >$

In other words, the primitive layer consists of a graph of pure identifiers and raw data. Each identifier maps either to a tuple of identifiers or to a plain byte array. The default representation of identifiers is a cryptographically strong type 4 UUID [19]. This makes it easier to manage them in a large distributed environment while virtually eliminating the chance of collision - each data peer can make up new IDs without a central authority. This layer is augmented with a set of indices some of which form an essential part of the data organization while others can be added by users or extension implementations. The main requirement from the key-value store perspective is that the indices support multiple ordered values per single key.

The model layer is where the hypergraph atom abstraction lives, together with the type system, caching, indexing and query facilities. The model layer is implemented by formalizing the layout of the primitive storage layer in the following way:

$AtomID \rightarrow [TypeID, ValueID, TargetID, ..., TargetID]$
$TypeID \rightarrow AtomID$
$TargetID \rightarrow AtomID$
$ValueID \rightarrow List < ID > | List < byte >$

In other words, each identity is taken to represent either a hypergraph atom, or the value of an atom. In the former case, the atom is stored as an identity tuple where the first argument is the (identity of the) type of the atom, the second its value, and the rest of the arguments form the target set of the atom. In the latter case, the value can be directly stored as a byte array or it can be an aggregate stored as an identity tuple as well. The layout of atom identities is managed by the core framework while the layout of data values is delegated to type implementations. Note that ValueIDs also form a recursive structure enabling the storage of arbitrarily complex structures that are orthogonal to the graph atom abstraction. The $TypeID \rightarrow AtomID$ production will be detailed in the next section.

The core indices needed for the efficient implementation of the model layer are:

$IncidenceIndex : UUID \rightarrow SortedSet < UUID >$
$TypeIndex : UUID \rightarrow SortedSet < UUID >$
$ValueIndex : UUID \rightarrow SortedSet < UUID >$

The IncidenceIndex maps a hypergraph atom to the set of all links pointing to it. The TypeIndex maps a hypergraph type atom to the set of all its instance atoms. The ValueIndex maps (the ID of) a top-level value structure to the set of atoms carrying that value as a payload.

4 Typing

HyperGraphDB has the ability to store arbitrary types of data as atoms in a programming-language-neutral way. Specifically, as an embedded database it maps data values in the host language to and from permanent storage. Therefore, it must cover typing constructs found in various languages in a way that integrates seamlessly with a given language's runtime environment. This is achieved via a general, extensible typing mechanism that interacts very closely with the storage layer, based on the foundational notion of a *type* in computer science.

In computational type theory, types are formalized by saying what can be done with them. A defining notion is equality: when are two elements of a given type equal? Another fundamental concept is compounding (or constructing) new types out of simpler ones. To make a new type, one needs type constructors, that is types of types. Finally, the notion of sub-typing, which is akin to subsumption or substitutability, is usually introduced independently. On the other hand, equality can be conceived as bi-directional subsumption: two instances of the same type are equal iff they subsume each other. The essence of types has been nicely summed up by the logician Jean-Yves Girard as "...plugging instructions. A term of a given type T is both something that can be plugged somewhere as well as a plug with free, typed variable" ([20]). This duality, together with the considerations above lead to the following minimalistic design:

- A type is an atom capable of storing, constructing and removing runtime representations of its instances to and from the primitive storage layer.
- A type is capable, given two of its instances, to tell whether one of them can be substituted for the other (subsumption relation).

In other words, types are just atoms assigned a special role that allows the system to maintain the assocation between types and instances as well as super-subtype relationships. This is formalized in the following Java interface that each type atom must implement:

```
public interface HGAtomType
{
  Object make(ID valueId,
              List<ID> targetSet,
              Set<ID> incidenceSet);
```

```
 ID store(Object instance); // return valueId for runtime instance
  void release(ID valueId);
  boolean subsumes(Object general, Object specific);
}
```

In addition to the `subsumes` relation, the interface simply defines CRUD methods needed to manage values in the storage layer. Because values are by definition immutable, there is no update method. Note that the `make` method takes both the target and incidence sets as extra parameters (besides the identifier of the value in primitive storage). Thus, the representation of an atom is a function of both those sets. This means that an atom can be annotated with various relationships that define what it is at runtime. The `make` method makes a type into an object factory. A type constructor is simply a type atom that returns an *HGAtomType* instance from its make method.

Now, the type system is bootstrapped by a set of (configurable) predefined types covering standard simple values such as numbers and strings as well some standard type constructors for record structures, lists and maps. Such predefined types all have the same type called *Top*, which is its own type. For example, the type constructor for record structures is managing record types which in turn manage concrete records. A compound type such as a record or a list can store its components by relying recursively on other HyperGraphDB types. That is, *HGAtomType* implementations can handle values both at the atom level and the pure (composite) value level.

Record-style structures with named parts are so common that we have defined an abstract interface for them called *HGCompositeType* that views complex values as multidimensional structures where each dimension is identified by a name and has an associated *HGProjection* implementation which is able to manipulate a value along that dimension. Such types that deal with complex structures are free to split them into separate identities in storage (i.e. as value links in the low-level graph) and provide functions like indexing, querying or reference counting for their parts. Alternatively, they can store them as compact blobs accessible only at the atom level.

In sum, types in HyperGraphDB aggregate many aspects commonly found in type systems in various computer languages. They play the role of object factories like classes in object-oriented languages. They are classes in the set theoretic sense, having a well-defined extension. They also define the structural properties of atoms and hence are intensional. Finally, they play a semantic role in constraining atom usage and enforcing certain consistency and integrity properties as typical in both typing and database systems. The combination of those aspects enable HyperGraphDB to naturally integrate many different formalisms without the need to single out any one in particular. Meta-models such as Sowa's conceptual graphs, RDF, combinatory logic expressions, object-orientation and even the standard relational model can be implemented "on their own terms". Furthermore, it allows for storage customization and optimization all the way to the very primitive values.

5 Indexing

Keeping in the spirit of the reflexive, open architecture of HyperGraphDB, indexing facilities are also an extensible feature. The implicit indexing of atoms described in section 3 is not optional as it is essential for an efficient support of the model layer semantics. For example, the incidence set indexing is crucial for the performance of both graph traversals and in general for querying the graph structure. Additional indices are being maintained at both the primitive and model layers.

At the primitive layer custom indices are employed by type implementations. Such indices are obtained directly from the storage layer and managed internally by a type. For instance, the default implementation of primitive types maintains an index between the primitive data and its value identifiers, thus enforcing value sharing between atoms at the storage level with reference counts. [3] Value sharing and reference counting is not done by default for complex types because there is no universal way to implement it efficiently unless the notion of primary key (i.e. a given dimension or a combination thereof) is introduced by implementing an appropriate complex type constructor.

At the model layer, indices can work both on the graph structure and value structure of atoms. However, they are always associated with atom types. This association is less constraining than it seems since an index will also apply to all sub-types of the type it is associated with. Indices are registered with the system by implementing the *HGIndexer* interface which must produce produce a key given an atom. *HGIndexer* instances are stored as atoms as well, and can thus have associated persistent meta-data for query planning, application algorithms etc. Some of the predefined indexers are listed in the following table:

ByPartIndexer	Indexes atoms with compound types along a given dimension.
ByTargetIndexer	Indexes atoms by a specific target (a position in the target tuple).
CompositeIndexer	Combines any two indexers into a single one. This allows to essentially precompute and maintain an arbitrary join.
LinkIndexer	Indexes atoms by their full target tuple.
TargetToTargetIndexer	Given a link, index one of its target by another.

Though we have not done that in the current implementation, an indexer could detect and index local graph structures to be used in more sophisticated graph mining algorithms.

6 Querying

Querying a graph database may take one of several forms. A graph traversal is a form of query yielding a sequence of atoms. Retrieving a set of not necessarily

[3] This is not necessary conceptually or technically; it is transparent at the model layer and could be changed in case different performance characteristics are required.

linked atoms matching some predicate, as in relational databases, is yet another kind. Finally, a third prominent category is pattern matching of graph structures.

Traversals are defined in terms of iterator APIs. Evidently, the more general hypergraph model entails a corresponding generalization of traversals.[4] There are several aspects to that generalization, some of which are already outlined in [16]. Firstly, the notion of node adjacency is generalized. Adjacent atoms are found as in a standard graph, by examining the links pointing to a given atom, i.e. its incidence set. Given an atom x_i and a set of typed link tuples pointing to it, one might want to examine only a subset of those links. Then given a particular link of interest $l = [x_1, ..., x_i, ..., x_n]$, the adjacent atoms could be a subset of $\{x_{i+1}...x_n\} or \{x_{i-1}...x_1\}$ depending on both atom types and properties and the direction of the traversal. Furthermore, in a case of uniform structural patterns in the graph, one might be interested in a notion of adjacency where more than on link is involved - that is, y is adjacent to x if it is reachable in a certain way. All of those cases are simply resolved at the API level by providing the traversal algorithm (breadth-first or depth-dirst) with an *adjacency list generator* - an interface that returns a list of atoms adjacent to a given atom. The second generalization of traversals adds an additional dimension to the process by allowing one to follow incident links as well as adjacent atoms. While handled equally trivially at the API, such hyper-traversals model a conceptually different phenomenon: jumps in abstractions levels within the representational structure. Because such jumps are highly domain dependent, they haven't been formalized within the system. It must be noted that both flat and hyper-traversals depend on the incidence index and the efficient caching of incidence sets.

Set-oriented queries are implemented through a simple, but efficient querying API where data is loaded on demand with lazy join, bi-directional (whenever possible) cursors. Query expressions are built out of a set of primitives[5]:

$eq(x), lt(x), eq("name", x), \ldots$	Compare atom's avalue.
$type(TypeID)$	Type of atom is $TypeID$.
$typePlus(TypeID)$	Type of atom is a subtype of $TypeID$.
$subsumes(AtomID)$	Atom is more general than $AtomID$.
$target(LinkID)$	Atom belongs to the target set of $LinkID$.
$incident(TargetID)$	Atom points to $TargetID$.
$link(ID_1, \ldots, ID_n)$	Atom's target set includes $\{ID_1, \ldots, ID_n\}$.
$orderedLink(ID_1, \ldots, ID_n)$	Atom is a tuple of the given form.
$arity(n)$	Atom's arity is n.

as well as the standard *and*, *or* and *not* operators. Much of the querying activities revolve around performing join operations on the various indices available for the

[4] In hypergraphs as finite set system, traversals are generalized to the notion of transversals which are sets having non-empty intersection with every edge of the graph. Computing minimal transversals has applications in various combinatorics algorithms, but HyperGraphDB has been mainly applied to tuple-oriented representational problems and instead provides facilities for computing directed n-ary traversals.

[5] For brevity, we list only the most fundamental ones.

query. Small to moderately sized atom incidence sets are cached as ordered ID arrays, which makes scanning and intersecting them extremely efficient. Wholes or portions of larger storage level indices get cached at the storage layer as well, but remain in B-Tree form. Joins are mostly performed on pairs of sorted sets (usually of UUIDs) via a zig-zag algorithm, when the sets support key-based random access, or a merge algorithm, when they don't. Intersections are implemented as bi-directional iterators, which allows for backtracking during more sophisticated search algorithms.

Because of the generality of the HyperGraphDB data model, querying for patterns would most conveniently be done through a special purpose language in the style of [21]. Such a query language, accounting for the various structural possibilities of typed atoms with directed or undirected hyperedges and complex values, is currently a work in progress.

7 Peer-to-Peer Distribution

Data distribution in HyperGraphDB is implemented at the model layer by using an agent-based, peer-to-peer framework. Algorithms are developed using communication protocols built using the Agent Communication Language (ACL) FIPA standard [22]. ACL is based in turn on speech act theory [23] and it defines a set of primitive communication performatives such as *propose*, *accept*, *inform*, *request*, *query* etc. Each agent maintains a set of conversations implementing dialog workflows (usually simple state machines) with a set of peers. All activities are asynchronous where incoming messages are dispatched by a scheduler and processed in a thread pool.

The P2P architecture was motivated by the most common use case of HyperGraphDB as a distributed knowledge representation engine. It is assumed that total availability of all knowledge at any particular location will not be possible, or in many cases not even desirable. For example, an eventually consistent data replication schema is based on the following algorithm:

1. Upon startup each agent broadcasts interest in certain atoms (defined by a boolean predicate) to its peers by sending a *subscribe* performative.
2. Each peer then listens to atom events from the database and upon addition, update or removal of an atom, it notifies interested peers by sending an *inform* communication act.
3. To ensure consistency, local transactions are linearly ordered by a version number and logged so that they can eventually reach all interested peers.
4. A peer that receives a notification about an atom transaction must acknowledge it and decide whether to enact the same transaction locally or not. There is no two-phase commit: once an event is acknowledged by a receiver, the sender can assume that it was processed.

This approach pushes the balance of replication vs. partitioning of data to the user level. We felt this was the appropriate choice since simple partitioning schemas, such as by type or by value, are easily configured on top of the base

algorithm while application-dependent graph structures can only be distributed meaningfully at the domain level. A fully transparent alternative that attempts to adapt the precise location of atoms based on local usage and meta-data would result in unpredictable and frequent remote queries, in addition to requiring a significant amount of additional local storage.

8 Example Application

One area where HyperGraphDB has been successfully applied is Natural Language Processing (NLP). We developed a distributed NLP processing system called Disko where the representational power of the data model considerably simplified the task. Disko uses a relatively standard NLP pipeline where a document is split into paragraphs and sentences, then each sentence is parsed into a set of syntactic dependency relations. Finally semantic analysis is applied to those relations yielding a logical representation of their meaning.

In this application, several domains are naturally represented independently while interacting in a clean and meaningfull way. First, a manually curated OWL ontology is stored as a hypergraph representing OWL classes as types that create runtime instances of OWL individuals based on their graph linkage. The OWL classes themselves are typed by an $OWLTypeConstructor$. Prior to parsing, an entity detection algorithm maps named entities (such as people and organizations) into individuals of that ontology. Second, the WordNet lexical database [24] is represented by storing synonym sets as undirected links between word atoms and semantic relationships between those sets (e.g. part-whole) as 2nd order directed links. A parser based on a dependency grammar [25] combined with a postprocessing relational extractor [26] yields strongly typed syntactic relationships between word atoms while a WSD (Word Sense Disambiguation) phase yields semantic relationships between synonym sets, again 2nd order, n-ary links. Further semantic analysis is performed by Prolog rules operating directly on the database instance via a mapping of logic terms to HyperGraphDB atoms, yielding further semantic relations. Both syntactic and semantic relations are of varying arity ranging from 1 (e.g. verb tense) to sometimes 4 (e.g. `event(buy, time, buyer, seller, object)`). The tree-like structure of the document is also recorded in HyperGraphDB with scoping parent-child binary links between (a) the document and its paragraphs, (b) a paragraph and its sentences, (c) a sentence and each linguistic relationship inferred from it. Finally, an implementation of the dataflow-based programming model ([27]) also relies on HyperGraphDB to store its distributed dataflow network.

A semantic search based on the NLP pipeline output makes heavy use of HyperGraphDB's indexing capabilities. First, a user question is translated into a set of relationships using the same pipeline. Ontological and lexical entities are quickly resolved by various $ByPartIndexer$s on their object-oriented representation. Non-trivial relationships (the ones with $arity > 1$) between those entities are matched against the database using $orderedLink(x_1, x_2, ...)$ queries. Such queries are translated to joins between the incidence sets of x_1 through

x_n and possibly including $ByTargetIndexer$s on some particularly large types of relationships. The matches thus accumulated are grouped according to their sentence/paragraph scopes, where the scopes themselves are efficiently resolved via a $TargetToTargetIndexer$, which is usually applied when one wants to efficiently navigate a tree sub-graph of a large hypergraph. The groups are scored and sorted into a search result set.

Note that both the ability of links to hold more than two targets and to point to other atoms is essential in this representation. Furthermore, the type system simplifies the programming effort by (1) making the mapping between persistent data and runtime representation transparent and (2) enforcing schema constraints on the representation. An approach relying on most other database models (including ODBMs, RDBMs, RDF stores, classical graph stores etc.) would force a contrived mapping of the several disjoint domains involved to a representation motivated solely by storage.

9 Conclusion

The data model adopted by HyperGraphDB generalizes classical graphs along several dimensions (reified n-ary links, strong typing) and as such it doesn't lend itself to a pointer structure based storage layout. On the other hand, an open layered architecture and an extensive type system allow the model to easily subsume many different formalisms without necessarily incurring a performance penalty in the process. Further development of the query language for the HyperGraphDB model, bindings for other programming languages such as C++, support for nested graphs and distributed algorithms based on the established foundation will hopefully fuel the adoption of this data model also in domains outside AI/NLP/Semantic Web research.

References

1. Angles, R., Gutierrez, C.: Survey of Graph Database Models. ACM Computing Surveys 40(1), Article 1 (2008)
2. Levene, M., Poulovassilis, A.: The Hypernode model and its associated query language. In: Proceedings of the 5th Jerusalem Conference on Information technology, pp. 520–530. IEEE Computer Society Press, Los Alamitos (1990)
3. Levene, M., Poulovassilis, A.: An object-oriented data model formalised through hypergraphs. Data Knowl. Eng. 6(3), 205–224 (1991)
4. Boley, H.: Directed recursive labelnode hypergraphs: A new representation-language. Artificial Intelligence 9(1) (1977)
5. Goertzel, B.: Patterns, Hypergraphs & Embodied General Intelligence. In: IEEE World Congress on Computational Intelligence, Vancouver, BC, Canada (2006)
6. Sheard, Tim: Languages of the Future. In: Proceedings of the ACM Conference on Object-Oriented Programming, Systems, Languages and Applications, Vancouver, British Columbia, Canada, (2004)
7. Klamt, S., Haus Utz-Uwe, Theis, F: Hypergraphs and cellular networks. PLoS Comput. Biol. 5(5) (2009)

8. Kok, S., Domingos, P.: Learning markov logic network structure via hypergraph lifting. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, pp. 505–512. ACM, New York (2009)
9. Resource Description Framework, http://www.w3.org/RDF/
10. RDF Semantics, http://www.w3.org/TR/rdf-mt/
11. Carroll, J.J., Bizer, C., Hayes, P., Stickler, P.: Named graphs, provenance and trust. In: Proceedings of the 14th international conference on World Wide Web, WWW 2005, pp. 613–622. ACM, New York (2005)
12. Topic Maps, http://www.isotopicmaps.org/
13. Berge, C.: Hypergraphs: combinatorics of finite sets. North-Holland, Amsterdam (1989)
14. Gallo, G., Long, G., Pallottino, S., Nguyen, S.: Directed hypergraphs and applications. Discrete Applied Mathematics 42(2-3), 177–201 (1993)
15. XML Schema, http://www.w3.org/XML/Schema
16. Boley, H.: Declarative operations on nets. In: Lehmann, F. (ed.) Semantic Networks in Artificial Intelligence, pp. 601–637. Pergamon Press, Oxford (1992)
17. Aczel, P.: Non-well-founded sets. CSLI Lecture Notes, vol. 14. Stanford University, Center for the Study of Language and Information, Stanford, CA (1988)
18. Olson, M.A., Bostic, K., Seltzer, M.: Berkeley DB. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference, p. 43. USENIX Association, Berkeley (1999)
19. IETF UUID draft specification, http://www.opengroup.org/dce/info/draft-leach-uuids-guids-01.txt
20. Girard, J.-Y., Lafont, Y., Taylor, P.: Proofs and Types. Cambridge University Press, Cambridge (1989)
21. SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/
22. FIPA Agent Communication Language Specification, http://www.fipa.org/repository/aclspecs.html
23. Searle, J.: Speech Acts. Cambridge University Press, Cambridge (1969)
24. Miller, G.A.: WordNet: A Lexical Database for English. Communications of the ACM 38(11), 39–41 (1995)
25. Sleator, D., Temperley, D.: Parsing English with a Link Grammar. Carnegie Mellon University Computer Science technical report CMU-CS-91-196 (1991)
26. RelEx Dependency Relationship Extractor, http://opencog.org/wiki/RelEx
27. Morrison, J.P.: Flow-Based Programming: A New Approach to Application Development. Van Nostrand Reinhold, New York (1994)

Survey of Graph Database Performance on the HPC Scalable Graph Analysis Benchmark*

D. Dominguez-Sal, P. Urbón-Bayes, A. Giménez-Vañó, S. Gómez-Villamor, N. Martínez-Bazán, and J.L. Larriba-Pey

DAMA-UPC, Universitat Politècnica de Catalunya,
Jordi Girona 1,3 08034 Barcelona, Spain
{ddomings,purbon,agimenez,sgomez,nmartine,larri}@ac.upc.edu

Abstract. The analysis of the relationship among data entities has lead to model them as graphs. Since the size of the datasets has significantly grown in the recent years, it has become necessary to implement efficient graph databases that can load and manage these huge datasets.

In this paper, we evaluate the performance of four of the most scalable native graph database projects (Neo4j, Jena, HypergraphDB and DEX). We implement the full HPC Scalable Graph Analysis Benchmark, and we test the performance of each database for different typical graph operations and graph sizes, showing that in their current development status, DEX and Neo4j are the most efficient graph databases.

1 Introduction

Relational database models are providing the storage support for many applications. Relational database systems store biological datasets, economic transactions, provide storage for dynamic websites, etc. Although the relational model has proven efficient and scales well for large datasets in table form, it is not adequate for applications that deeply analyze relationships among entities. For example, the computation of a shortest path is not supported by a standard SQL query, and needs a stored procedure with a large number of joins, which are typically expensive to be computed.

On the other hand, the database community has become aware of this need for data storage applications for certain environments that do not require SQL and relational storage. In the recent literature, we find many projects that set alternative constraints to those imposed by the relational model in order to provide more natural query interfaces and improve the system performance. Some examples of this trend are: documental databases such as Lucene [2], which is able to fully index large document collections and support queries that rank the documents according to information retrieval measures; key-value data storages, such as BerkeleyDB [5], which stores the data collections as pairs that map a

* The members of DAMA-UPC thank the Ministry of Science and Innovation of Spain and Generalitat de Catalunya, for grant numbers TIN2009-14560-C03-03 and GRC-1087 respectively.

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 37–48, 2010.

given key to an object; or even tabular-like representations without SQL support, such as Bigtable [8], which stores data as three dimensional tables that are indexed by two strings plus a timestamp that allows storing temporal sequences.

One type of data representation that is growing in popularity is graph databases (GDB) because many datasets are naturally modeled as a network of relationships: web data, authorship relationships, biological food chains, social networks, protein interaction, etc. Besides, the analysis of these datasets is guided by graph operations such as finding neighborhood of a node, graph traversals, finding minimum paths or community detection. Therefore, the storage and query execution in GDBs become the natural choice to store and develop queries on graphs.

Given that there are several implementations of GDBs, in this paper we will measure the performance of the most popular GDB alternatives. We analyze four different GDB libraries: Neo4j, HypergraphDB, Jena and DEX. We omitted InfoGrid, which is also a popular graph analysis tool, from this comparison because InfoGrid does not support attributes on edges (which are required by the HPC-SGAB) and the recommended workaround is to create additional intermediate nodes as associative entities, which store the weight [10]. This requires to implement specialized implementations for InfoGrid, that would differ from the shared algorithm patterns implemented for the rest of GDBs.

For the evaluation, we apply the recently developed HPC Scalable Graph Analysis Benchmark [3] (HPC-SGAB). This benchmark was designed by researchers from academia, as well as members of several industrial companies to capture the most representative graph operations such as graph loading, short navigations or full graph traversals. It is out of the scope of this paper to present novel techniques to solve more efficiently the HPC-SGAB kernels: our main focus is to take advantage of the benchmark to evaluate the performance of the most representative GDBs available.

To our knowledge, the present work is the first full implementation of the HPC-SGAB on several GDB engines, running in the same hardware setup. Furthermore, we discuss several aspects from the HPC-SGAB, that we think that should be considered for future refinement of the benchmark. Our results show that some of the current GDBs can handle millions nodes efficiently, proving that DEX and Neo4j are the most efficient current GDB implementations.

The paper is organized as follows: Section 2 presents the four GDBs under study and summarizes its main features, Section 3 describes the HPC-SGAB, Section 4 reports and discusses the experimental results, Section 5 describes our experience implementing the benchmark, and finally, Section 6 concludes the paper.

2 Graph Database Libraries

2.1 Neo4j

Neo4j is a GDB designed for network oriented data, either in tree or general graph form. Neo4j does not rely on a relational layout of the data, but a network model storage that natively stores nodes, relationships and attributes [14]. Although the wiki provides detailed information about configuring Neo4j [16],

to our knowledge, there is no research publication or technical report with an extensive description of the internals of Neo4j.

Neo4j is one of the most popular alternatives of GDBs due to its dual free software/commercial license model (AGPL license). Neo4j is fully written in Java and can be deployed on multiple systems. It supports transactions and fulfills the ACID consistence properties [14].

2.2 HypergraphDB

Hypergraph is a GDB, which was designed for artificial intelligence and semantic web projects [9]. As a consequence of the the requirements from these environments, HypergraphDB stores not only graphs but also hypergraph structures. A hypergraph is a mathematical generalization of the concept of a graph, in which the edges are substituted by hyperdges. The difference between edges and hyperedges is the number of nodes that they connect: a regular edge connects two nodes of a graph, but a hyperedge connects an arbitrary set of nodes. Given that the tests in the benchmark only compute operations on regular graphs, we will not take advantage of this feature from HypergraphDB.

HypergraphDB stores all the graph information in the form of key-value pairs. Each object of the graph, either nodes or edges, is identified with a single key that is called atom. Each atom is related to a set of atoms, which can contain zero or any number of elements. These relationships create the topological structure of the graph. HypergraphDB also supports node and edge typing. The types are also implemented as atoms that contain all the elements of a particular type [9].

The key-value structures are stored on an external library, BerkeleyDB [5], which does not support relational queries but provides specialized structures for this type of storage. BerkeleyDB offers three access methods to data: hashes, B-trees and Recno [17]. The first is implemented as a linear hash, which in case of multiple collisions performs multiple hashes to redistribute the colliding keys. The B-trees store the data in the leaves of a balanced tree. Recno assigns a logical record identifier to each pair, which is indexed for direct access.

2.3 Jena (RDF)

The Resource Description Framework (RDF) is a standard for describing relationships among entities. A relationship expressed in RDF is expressed as a triplet: a subject, a predicate and an object. The interpretation is that the subject applies a predicate on the object. Therefore, a RDF description can be viewed as a graph where the subject and the object are the vertices, and the predicate corresponds to the edge that relates them. There are several RDF based graph implementations: Jena [11], Sesame [18], AllegroGraph [1], etc. In our tests, we selected Jena because it is one of the most widely used.

Jena can be run on two storage backends, which store the RDF triplets: SDB and TDB [11]. In this paper, we limit our tests to the TDB backend because it provides better performance according to the Jena documentation. TDB stores

the graph as a set of tables and indexes: (a) the *node table* and (b) the *triple indexes*[1].

The node table stores a map of the nodes in the graph to the node identifiers (which are a MD5 hash), and viceversa. The storage is a sequential access file for the identifier-to-node mapping, and a B-tree for the node-to-identifier mapping.

The triple indexes store the RDF tuples, that describe the structure of the graph. Jena does not implement a table with the three attributes of the RDF tuple, but three separate indexes. Each of these three indexes takes as the sorting key, each one of the three attributes of an RDF structure. Nevertheless, the value stored in the index is the full tuple for all the three indexes, in other words an access to any of the three indexes is able to retrieve a RDF relationship.

2.4 DEX

DEX is an efficient GDB implementation based on bitmap representations of the entities. All the nodes and edges are encoded as collections of objects, each of which has a unique oid that is a logical identifier [13].

DEX implements two main types of structures: bitmaps and maps. DEX converts a logical adjacency matrix into multiple small indexes to improve the management of out-of-core workloads, with the use of efficient I/O and cache policies. DEX encodes the adjacency list of each node in a bitmap, which for the adjacent nodes has the corresponding bit set. Given that bitmaps of graphs are typically sparse, the bitmaps are compressed, and hence are more compact than traditional adjacency matrices.

Additionally, DEX is able to support attributes for either vertices and edges, which are stored in a B-tree index. DEX implements two types of maps, which are traversed depending on the query: ones that map from the oid to the attribute value, and others from the attribute value to the list of oids that have this value. This schema fully indexes the whole contents of the GDB.

In a nutshell, this implementation model based on multiple indexes favors the caching of significant parts of the data with a small memory usage, reverting in a better efficient storage and query performance [13].

3 Benchmark Description

In this paper, we will implement the queries of the HPC Scalable Graph Analysis Benchmark v1.0 for the previously described GDBs. This benchmark was designed from a collaboration between researchers from universities and engineers from industrial partners that analyze different aspects of the performance of a GDB. In the rest of this section, we summarize this benchmark.

3.1 Data Generation

In this benchmark, we test the performance of operations on directed and weighted graphs. The dataset is generated using the R-MAT algorithm, which is able to

[1] An additional small table, called prefix table, is also used to export the results to other formats such as XML. This table is not used in a regular query execution [11].

build graphs of any particular size and edge density [7]. Furthermore, the graphs that R-MAT builds follow typical real graph distributions (power law distributions), such as those found in real life situations [7].

The general idea of R-MAT is the following. Suppose that we represent the graph as an adjacency matrix, in which each row and column is a vertex and the coordinates into the matrix indicate if the two vertices are connected. R-MAT divides the matrix in four equal squared sections, which have an associated probability in HPC-SGAB: a, b, c and d. Then, R-MAT picks one of the four sections according to these probabilities. Once the section is selected, R-MAT iterates recursively this process dividing again the selected section in four equal parts. The process finishes when the size of the selection is equal to one position, and then, R-MAT selects this position as the next inserted edge. The process can be iterated arbitrarily to create graphs of the desired density.

In general, the parameters must not be symmetric in order to have power law distributions in the graph (which are the most common distribution in real huge graph datasets [12]). In our benchmark, we take the recommended values by the HPC-SGAB for all the graphs tested: $a = 0.55$, $b = 0.1$, $c = 0.1$ and $d = 0.25$. The number of nodes and edges of the graph is indicated by a parameter called *scale*. The number of nodes is $N = 2^{scale}$, and the corresponding number of edges is $M = 8 \cdot N$. Finally, the weight of each edge is a positive integer value following a uniform distribution with maximum value 2^{scale}. In the experimental section, we fix the probabilities to the stated parameters and study the performance of the different GDBs for different scales.

The R-MAT generation time is not computed for the benchmark, since it is not part of the GDB performance. Therefore, R-MAT generates a file of tuples with the form <StartVertex, EndVertex, Weight>, that will be loaded into the native GDB format.

3.2 Kernel Description

This benchmark is composed of four kernels, where a kernel is an operation whose performance is tested. The first kernel loads the data in the GDB, and the rest of kernels will use this image to compute the queries. The kernels are the following:

- **Kernel 1:** The first kernel measures the edge and node insertion performance of a GDB. This kernel reads the database file, in the format described in Section 3.1, and loads it. The loading includes the creation of all the indexes needed to speedup the computation of the following kernels. Note that the R-MAT file generation process is not included in this timing. In the following queries to the GDB, we use the graph loaded by this kernel, which cannot be modified during the following kernels.
- **Kernel 2:** This kernel measures the time needed to find a set of edges that meet a condition, in this case it finds all the edges with the largest weight. The algorithm output is the list of edges and nodes that connect them. The list of nodes is stored in order to initialize Kernel 3.

- **Kernel 3:** This kernel measures the time spent to build subgraphs in the neighborhood of a node. It computes a k-hops operation, which we implemented using a breadth first search traversal, starting from the edges produced by Kernel 2. We set 2 as the number of additional hops that the algorithm traverses from the tail node of each edge, which adds up to total length of three including the initial edge. Since the graphs follow power law distributions, this operation may access a significant number of nodes.
- **Kernel 4:** This kernel estimates the traversal performance of the GDB over the whole graph. This value is estimated as the Traversed Edges Per Second (TEPS). Kernel 4 estimates the TEPS using a complex graph operation that calculates the Betweness Centrality (BC) of the graph. The BC of a graph gives a score to each node that indicates how far is a node from the center of the graph. It assigns low scores to the nodes that are in the external parts of the graph, and high scores to the nodes that are in the internal parts of the graphs. More formally, the BC of one vertex v accounts for the ratio of shortest paths (σ_{st}) between any pair s and t of nodes that pass through v ($\sigma_{st}(v)$) with respect to those that do not:

$$BC(v) = \sum_{s \neq v \neq t \in G} \frac{\sigma_{st}(v)}{\sigma_{st}}.$$

  Note that when we refer to the BC of a node we are not considering the weight of the edges. However, this kernel does not compute the BC on the whole graph but on the induced graph that removes all edges with a weight multiple of eight and keeps all the nodes, as indicated by the benchmark. We implement this restriction implementing a BC algorithm that skips the edges that are multiples of eight while exploring the graph, instead of computing this subgraph during Kernel 1.
  We implement the BC using the Bader's algorithm [4] for a single core, which provides an approximated solution based on the Brande's exact algorithm [6]. The complexity of Bader's strategy is $O(k \cdot M)$, where k is the number of samples, and is the recommended algorithm by the HPC-SGAB. The Bader's algorithm calculates the centrality using a sample of multiple BFS traversals, starting from different nodes. In our tests, we pick 8 (in other words, scale 3) as the number of samples for computing the BC.

4 Experiments

4.1 Experimental Setup

For each of the databases, we used the latest available versions: Neo4j v.1.0, Hypergraph v.1.0, Jena v.2.6.2 (with TDB 0.8.4), and Dex v.3.0. All the benchmarks were implemented using the Java interface of the GDBs, which is the recommended programming API for all of them. In order to minimize differences in the performance because of the algorithm implementation, we followed

the recommended algorithms described in the HPC-SAGB description, and all the implementations were written following the same generic implementation.[2]

In order to configure each database, we used the default configuration plus the recommendations found in the documentation of the websites. For Neo4j, we implement Kernel 1 with the batch inserter (which disables transactions), configured with the recommended values: $N \cdot 9$ bytes for the edge store, $N \cdot 33$ bytes for the node store [15]. For HyperGraphDB, we disabled the transactional capabilities, which were not required in the benchmark and are an additional overhead.

The kernels are executed in the order indicated by the HPC-SGAB. In Section 4.2, we report the results after a warm up stage, that computes kernels 2 and 3 once. This setup resembles a database system that is computing queries during long timespans. Nevertheless, we also discuss the results without a warm up stage in Section 5. We halt the execution of kernels that take more than 24 hours to compute.

We execute our experiments in a computer equipped with two Quad Core Intel Xeon E5410 at 2,33 GHz, 11 GB of RAM and a LFF 2.25Tb disk. We use the default parameterization of the Java Virtual Machine for all the kernels except for the largest ones: 2GB for scale 20, and 10GB for scale 22 and up.

4.2 Analysis of Results

We executed the HPC-SGAB with four different scales for the already described databases: 10, 15 and 20. These setups correspond to databases with 1k, 32k and 1M nodes, which account for a total number of objects ranging from approximately 10k to more than 9.4M objects.

Tables 1, 2 and 3 summarize the results of the benchmark for the different kernels tested. We were not able to scale the comparison over larger scale factors because most GDBs had problems to load the full graph in a reasonable time with our current hardware, and hence we could not continue the comparison for even larger datasets. We could not load the scale 22 graph in less than 24 hours in the GDBs tested, except for DEX where it took 27.5 minutes. Furthermore, we were not able to load the scale 24 graph in less than three days in any of the databases, except for DEX, for which it took 28 hours.

We first observe that the scalability of all the databases is not equivalent. Although DEX and Neo4j were able to scale up to the 1M nodes dataset, Jena and HypergraphDB could not load the database in a comparable time. We were not able to load the graphs with 1M nodes in HypergraphDB in 24 hours, and although Jena was able to load the graph with 1M nodes faster than Neo4j, Jena did not scale to the largest scale.

Regarding Kernel 1, which measures the load time, we find that the fastest alternative is DEX that loads the dataset at least one order of magnitude faster than the rest of algorithms. The insertion rate for DEX scales well for the different benchmarks, adding roughly 30k objects per second as depicted in Figure 1. Jena inserts approximately 2k objects per second, and Neo4j is up to two orders

[2] The query implementations are publicly available in the following website: `http://trabal.ac.upc.edu/public/dama-upc-iwgd2010-hpca-sgab-source.zip`

Table 1. Scale factor 10

Kernel	DEX	Neo4j	Jena	HypergraphDB
K1 Load (s)	**0.316**	19.30	7.895	376.9
K2 Scan edges (s)	**0.001**	0.131	0.090	0.052
K3 2-hops (s)	**0.003**	0.006	0.245	0.015
K4 BC (s)	0.512	**0.393**	5.064	1.242
Db size (MB)	2.1	**0.6**	6.6	26.0

Table 2. Scale factor 15

Kernel	DEX	Neo4j	Jena	HypergraphDB
K1 Load (s)	**7.44**	697	141	+24h
K2 Scan edges (s)	**0.001**	2.71	0.689	N/A
K3 2-hops (s)	**0.012**	0.026	0.443	N/A
K4 BC (s)	14.8	**8.24**	138	N/A
Db size (MB)	30	**17**	207	N/A

Table 3. Scale factor 20

Kernel	DEX	Neo4j	Jena	HypergraphDB
K1 Load (s)	**317**	32094	4560	+24h
K2 Scan edges (s)	**0.005**	751	18.60	N/A
K3 2-hops (s)	0.033	**0.023**	0.458	N/A
K4 BC (s)	**617**	7027	59512	N/A
Db size (MB)	893	**539**	6656	N/A

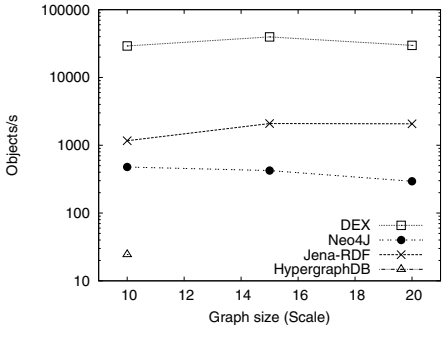


Fig. 1. Objects loaded per sec. (Kernel 1)

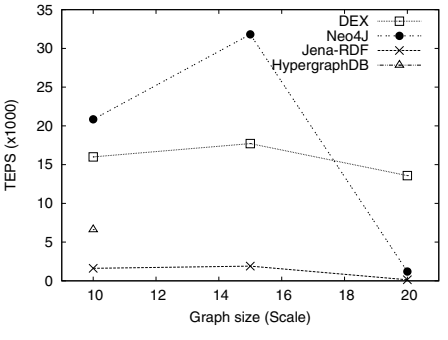


Fig. 2. TEPS (Kernel 4)

of magnitude slower than DEX for adding data. On the other hand, the image generated by Neo4j is the most compact: it is 40% smaller than the following one, which is DEX, and 90% and 98% with respect to Jena and HypergraphDB.

Kernel 2, measures the time to find a subset of edges. For this task, DEX is the fastest algorithm again for all the tested sizes because it can take advantage of its indexes over edges [13]. It is a fast operation for all the GDBs, because

it consists of a single iteration over the weights of all edges. However, for large graphs this operation is expensive for Neo4j because the API does not provide a direct access for iterating over all edges. In order to retrieve the edges, Neo4j iterates over all the nodes in the database, and explores the adjacent edges, which supposes an additional overhead to Kernel 2. This is particularly expensive when the graph is large because since it follows a power law distribution, some nodes have an elevated degree.

All the GDBs are able to compute the third kernel very fast. The operation accesses the local neighborhood of one node and we observe that Neo4j is the best for larger graphs for Kernel 3. On the other hand, DEX is the fastest for the small graph and is very close to Neo4j for the large graph. As we discuss in Section 5, this kernel is heavily influenced by the warm up of the database because of the very short execution time.

Finally, Kernel 4 measures the traversal performance of each algorithm over the whole graph. For the smallest graph size, Neo4j obtains the best performance, but for larger graphs DEX scales better. For small graphs, DEX and Neo4j are significantly faster than the other two GDBs: Neo4j is up to five times faster than HypergraphDB and one order of magnitude faster than Jena. For large graphs, there are big differences among the three databases. DEX obtains a speedup above 60 over Jena and a speedup above 11 over Neo4j.

Figure 2 depicts the discussed results for Kernel 4 as a function of the traversed edges per second, which is more adequate for comparing different database sizes. We observe that DEX and Jena scale with the size of the graph because the slope of the curve is not very pronounced. Nevertheless, DEX is able to traverse 10 times more edges than Jena, which makes DEX a better choice for traversing operations. On the other hand, Neo4J traverses more than 32k TEPS for the small graph but when the graph is large the performance decreases to less than 1.2k TEPS.

In summary, we found that DEX and Neo4j are the only databases that were able to load the largest graphs and compute the queries efficiently. HypergraphDB failed to load the scale 15 and 20 databases in a reasonable time,

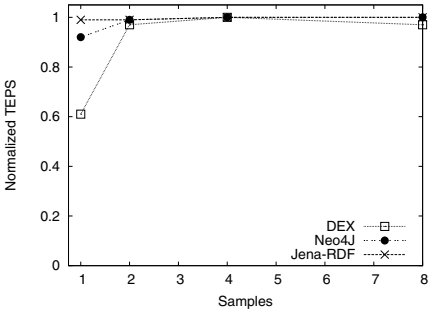


Fig. 3. TEPS Normalized to the fastest value for a given GDB

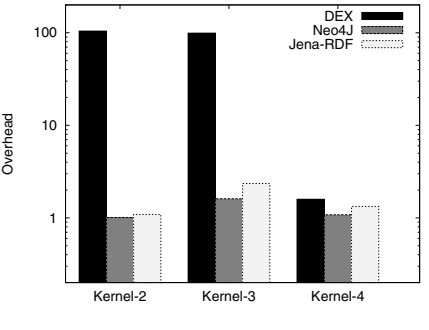


Fig. 4. Warm up overhead (scale 20)

and in any case, it was not faster than DEX and Neo4j for any kernel. Jena scaled up to factor 20 but it was slower than Neo4j and DEX. We observed that DEX is the fastest for most kernels (9 out of 12 kernels), and in the three cases where Neo4j is the fastest, the difference is small. Furthermore, we detected that Neo4j did not scale as well as DEX for Kernels 1, 2 and 4, in which DEX is up to more than ten times faster than Neo4j.

5 Experience from the Benchmark

In this section, we discuss some considerations that we have drawn after implementing the benchmark. We group them into the following points:

- **Imbalance in Kernels 2 and 3 costs:** We found that the computation cost of the different kernel operations is very different, specially for the largest graphs, where the computation of Kernel 4 consumes 10k more time than Kernels 2 and 3 for scale 20. Kernels 2 and 3 measure the performance of operations that are local in nature and whose cost does not grow at the same rate of the graph. For example, the number of objects grows 1000 times from scale 10 to 20, and thus Kernel 1 building costs are multiplied by this magnitude. But, the cost of computing Kernel 3 is only about ten times larger with the change of scale.
- **TEPS estimation:** On the other side of the spectrum, the BC function calculated in Kernel 4 is a very complex operation. The use of the exact BC is not affordable for large graphs because the number of BFS traversals is equal to the number of nodes. Therefore, the application of the Brandes approximated algorithm for Kernel 4 is necessary for large graphs.
  Furthermore, we found that the number of samples to estimate the TEPS for a particular GDB does not need many traversals. In Figure 3, we depict the average TEPS of the databases for an increasing number of samples of Kernel 4. In this plot, the TEPS is normalized to the TEPS of the fastest sample of the BC algorithm for each database. We observed that the first sample exhibits a warm up effect, such as for DEX and Neo4j, because they load for first time some of the data structures of the graphs. However, once the number of samples is above two the TEPS stabilizes for all the databases. Thus, as Figure 3 shows, Kernel 4 does not need an excessive number of iterations to estimate the TEPS obtained by a GDB, i.e. more than eight iterations would not be necessary to measure the traversal performance of the GDB.
- **Warm up:** An important topic that is not discussed in the benchmark description is the warm up process of the GDB. Typically, the implementation of good caching strategies improves the performance of an application because it does not read information from the secondary storage. In Figure 4, we plot the overhead produced by the first execution of one kernel compared to the following computation of the same kernel. The difference between having all data in memory or loading part from disk is more severe for kernels

that measure simple operations that are computed very fast, such as the kernel 2 and 3 of the HPC-SGAB. For example, in DEX the first computation of Kernel 2 takes approximately 100 times more to be computed than the following executions. The reason is that the first execution loads from disk the necessary data pages, which takes half a second, and then computes the query in five milliseconds more. The following executions skip the cost to load from disk, and thus are significantly faster. This effect is clearly visible in DEX and Jena in Figure 4.

As a consequence of the previous issues we believe that some aspects can still be improved for future revisions of the benchmark. From our point of view, the time needed to compute all kernels should be harmonized and scale at a similar rate to the graph growth. For example, we suggest that Kernel 3 should perform the k-hops operation over a larger set of nodes as origin. Instead of only picking the set of edges with the largest weight, we suggest to pick a fixed percentage of edges, such as the set of the 1‰ largest edges.

Furthermore, we think that it is important to include a remark about the warm up of the GDB in the benchmark description. Since GDBs are complex and have their own bufferpool implementation, we think that the fairest option would be to enable a warm up process before the timed kernel executions. This is particularly important for graphs that do not fit in memory and where the bufferpool hit rate plays a significant role in the final system performance.

6 Conclusions

In this paper, we have tested the performance of four of the most popular graph data management applications with the recently developed HPC-SGAB. The results derived from the benchmark show that for small graphs all four databases are capable to achieve a reasonable performance for most operations. However, in our hardware setup, only DEX and Neo4j were able to load the largest benchmark sizes.

DEX is the fastest database loading the graph data and performing the operations that scan all the edges of the graph, exhibiting an improvement over one order of magnitude than the second best graph database, Neo4j. For Kernel 3, we found that Neo4j is faster than DEX for the large dataset, and the reverse happens for the small dataset. Nevertheless, the difference between both databases is small for all the data sizes tested in Kernel 3. For Kernel 4, Neo4j is able traverse more than 32k TEPS, but they do not scale properly for graphs with a million nodes because the performance drops to 1,2k TEPS. On the other hand, DEX is able to scale better, traversing roughly 15k TEPS either for small and large graphs. All in all, Neo4j obtained a good throughput for some operations, but we found that it had scalability problems for some operations on large data volumes, specially in the full graph traversals. DEX achieved the best performance for most operations and close to Neo4j, in those where Neo4j was the fastest.

Finally, we discussed two aspects to take into account for improving future versions of HPC-SGAB kernels based on our experience implementing and testing it on real hardware: (a) harmonize the costs of Kernels 2, 3 and 4 with operations that scale similarly with the graph size in order to obtain a more balanced benchmark, (b) defining a warm up policy that is able to show the performance of the databases with respect to the secondary storage devices for huge graphs.

References

1. AllegroGraph. AllegroGraph website, http://www.franz.com/agraph/ (last retrieved in May 2010)
2. Apache Lucene (September 2008), Lucene website, http://lucene.apache.org/
3. Bader, D., Feo, J., Gilbert, J., Kepner, J., Koetser, D., Loh, E., Madduri, K., Mann, B., Meuse, T., Robinson, E.: HPC Scalable Graph Analysis Benchmark v1.0. HPC Graph Analysis (February 2009)
4. Bader, D., Madduri, K.: Parallel algorithms for evaluating centrality indices in real-world networks. In: ICPP, pp. 539–550 (2006)
5. BerkeleyDB. BerkeleyDB website, http://www.oracle.com/database/berkeley-db/index.html (last retrieved in March 2010)
6. Brandes, U.: A faster algorithm for betweenness centrality. Journal of Mathematical Sociology 25(2), 163–177 (2001)
7. Chakrabarti, D., Zhan, Y., Faloutsos, C.: R-mat: A recursive model for graph mining. In: Jonker, W., Petković, M. (eds.) SDM 2004. LNCS, vol. 3178. Springer, Heidelberg (2004)
8. Chang, F., Dean, J., Ghemawat, S., et al.: Bigtable: A distributed storage system for structured data. ACM Trans. Comput. Syst. 26(2) (2008)
9. HypergraphDB. HypergraphDB website, http://www.kobrix.com/hgdb.jsp (last retrieved in March 2010)
10. Infogrid. Blog, http://infogrid.org/blog/2010/03/operations-on-a-graph-databae-part-4 (last retrieved in March 2010)
11. Jena-RDF. Jena documentation, http://jena.sourceforge.net/documentation.html (last retrieved in March 2010)
12. Leskovec, J., Lang, L., Dasgupta, A., Mahoney, M.: Statistical properties of community structure in large social and information networks. In: WWW, pp. 695–704 (2008)
13. Martínez-Bazan, N., Muntés-Mulero, V., Gómez-Villamor, S., et al.: Dex: high-performance exploration on large graphs for information retrieval. In: CIKM, pp. 573–582 (2007)
14. Neo4j. The neo database (2006), http://dist.neo4j.org/neo-technology-introduction.pdf
15. Neo4j. Batch Insert, http://wiki.neo4j.org/content/Batch_Insert (last retrieved in March 2010)
16. Neo4j. Neo4j wiki documentation, http://wiki.neo4j.org/content/Main_Page (last retrieven in March 2010)
17. Olson, M., Bostic, K., Seltzer, M.: Berkeley db. In: USENIX Annual Technical Conference, FREENIX Track, pp. 183–191. USENIX (1999)
18. Sesame. Open RDF website, http://www.openrdf.org/ (last retrieved in May 2010)

A Model for Automatic Generation of Multi-partite Graphs from Arbitrary Data

Ricardo Baeza-Yates[1], Nieves Brisaboa[2], and Josep Larriba-Pey[3]

[1] Univ. Pompeu Fabra & Yahoo! Research, Barcelona, Spain
rbaeza@acm.org
[2] Univ. of A Coruña, A Coruña, Spain
brisaboa@udc.es
[3] DAMA-UPC, Universitat Politècnica de Catalunya, Barcelona-Tech, Barcelona, Spain
larri@ac.upc.edu

Abstract. In this paper we propose a generic model to generate basic multi-partite graphs obtained by associations found in arbitrary data. The interest of such a model is to be the formal basis behind a tool for automatic graph generation that we are developing. This tool will automatically generate the basic multi-partite graphs that represents the arbitrary data provided as input.

We consider input data as collections of complex objects composed by a set or a list of heterogeneous elements. Our tool will provide an interface for the user to specify the kind of nodes that are relevant for the application domain in each case. Those nodes will be obtained from the complex input objects by simple *extraction* rules.

The objective of this paper is to present the model to represent basic multi-partite graphs and the way to define the nodes of the graph using simple *derivation rules* defined by the user. In order to validate our model we give three examples of radically different data sets. Those examples come from the Web log queries, processing text collections, and bibliographic databases.

1 Introduction

The use of graphs for data analysis becomes essential in many situations. Plain texts, for instance, formed by paragraphs, sentences and words at the simplest, have no structure but giving them a graph structure may lead to better analysis of their characteristics [11]. In other situations, like bibliographic analysis [10], where data are structured, or in web structure analysis [7], the use of graphs gives a significant insight that allows for deeper and faster analysis.

Nowadays the data to be analyzed grow in size and in the number of varied sources, making graphs and the use of graph analysis commonplace in some situations. Thus, at times, the use of graphs may provide and easier and more understandable artifact to provide simple views of the data and their relationships. In other cases, however, the complexity of the graphs obtained from plain

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 49–60, 2010.

or structured data may deter the proper analysis, or even the capability of computers to process them. Thus, it seems obvious that creating graphs in a simple way will be necessary since graphs are becoming important in many situations.

However, creating a graph from a data collection is not a trivial nor a fast operation. Therefore, tools for the automatic generation of basic graphs from a general collection of input data will be very useful and welcome in the future. In order to do so, it is first necessary to create the ground basis for these tools.

The objective and main contribution of this paper is to set the basis for the creation of tools for the automatic generation of graphs from basic data sets in the framework of a broader research project oriented to high performance graph management and analysis [3].

The type of tool we are planning will allow the user to define the kind of nodes and relationships of interest. Thus, a single data set, by means of using different transformations and rules, will lead to different graphs that are, in a sense, views of a more generic graph that suit and simplify further analytic queries on the resulting product graphs.

The rest of the paper is organized as follows. Section 2 covers related work. The model is presented in Section 3, while in Section 4 we present three examples of its use. We end in Section 5 with some final remarks.

2 Related Work

Graphs are a general mathematical model that formalizes each connection between two entities and emerges anywhere when we encode or represent that such two entities are associated. These connections can occur between different entity sets, for example which user visits which web site. Thus, graphs can model bibliographical databases, XML databases, legal databases or the Web itself. In that case we talk about multi-partite graphs which is the focus of this paper.

In fact, link analysis can be considered the birth of graph mining in the Web, with PageRank and HITS being the main simple landmark results. In the context of the Web, graphs are dynamic, changing over time and also with different characteristics in different cultural and geographical scopes. In social networks, graph mining also enables the study of populations' behaviour and finding communities. Successful graph mining not only enables segmentation of users but also enables prediction of behaviour. Therefore, graph mining remains an area of high interest and development. A detailed picture with respect to algorithms and potential for graph mining is presented by Chakrabarti and Faloutsos [8]. The structure of their survey focuses around what are the graph patterns that seem to be frequent in real graphs reflecting the many domains of human behavior model in them. Interestingly, the survey points out that algorithms for computing community effects, cluster coefficients and the like have emerged from many disciplines and this is the aspect of graph mining that reflects the largest impact as well as the potential from cross-discipline fertilization. Another important work is the graph mining book by Cook and Holder [9].

Current research has mainly focused on efficiency aspects to handle the scalability issues that arise in very large graphs, for example, stream processing

models that can mine a graph in linear time [15]. Here, we look at the modeling side, so we can define and prototype applications to test hypotheses or to do exploratory research before handling large volumes of data. This is similar to the spirit of WIM, where Web mining applications can be prototyped very fast using a high-level programming language [14].

3 A Model for Generating Multi-partite Graphs

The generic model we propose in this paper was designed to support the general tool we are developing. This tool will automatically create the basic multi-partite graphs that can be obtained applying *derivation rules* over collections of objects presented as input. Each kind of *input* object can be a set or a list of heterogeneous elements. That is, our goal is to implement a tool to automatically generate an initial and basic multi-partite graph from the collections of complex objects (set or list of elements) provided as inputs. The generated graphs, that the tool we are developing will represent using GraphML [6], could be used later for more complex manipulations and queries using general graph management tools such as Neo4j [13] or DEX [12].

Our model distinguishes three levels of objects that will represent the nodes of the graph. We call those: *Input* objects (I), *Derived* objects (D) and *Atomic* objects (A) depending on the level they occupy. All the objects, except those provided as input, are generated using *derivation or extraction rules* (R). Those rules will be defined by the user according to the characteristics and needs of the application domain. The rules are always applied over objects of some type and produce objects of other types with, usually, a lower granularity level. The tool will apply such rules R_i over the collection(s) of input objects I_i, obtaining derived objects D_j of the different types defined by the user. In the same way, the atomic elements A_k belonging to the different defined atomic types will be obtained applying derivation rules over derived or input objects. Normally, the input data will be at one extreme of the multi-partite graph while the atomic elements will be at the other extreme, representing the highest and lowest possible levels, respectively.

Summarizing, the main elements of our model are:

- **Derivation rules R_i.** Those are user defined procedures that the tool will use to produce derived and atomic objects from the input objects. The tool will provide simple algorithms for the user to define rules such as "extract all the words from a document" or "eliminate stopwords". The interesting derivation rules are domain dependant, therefore, the tool will allow for the definition of more complex derivation rules that will use external procedures provided by the user. Notice that the relationships among an object and those objects derived from it are defined by the derivation rule itself. Therefore, we can use the derivation rules to label the edges of our graphs.
- **Input objects I_i.** Those are the objects provided as input to the tool. All the input objects could be of the same or different type. For example, we may have a collection of documents and a collection of Universities. Each document and each University are in this case *input objects*, but those original objects will

belong to two different object types (or collections). We will denote objects as ID_i (Input Document) and IU_j (Input University), respectively, depending on the collection they belong to. The main characteristic of our model is that we consider that each type of input object can be a set (unordered) or a list (ordered) of *elements*. For example, a document is a list of paragraphs, or a list of words, while a university can be seen as a set of data of different type about the university such as name, departments, etc.

- **Derived Objects** D_j. These objects are simpler (lower level) than the input objects but that are still complex enough to be decomposed again. That is, they can be obtained from an input object or from other derived objects, but they can be decomposed again. Therefore, the model allows for derived objects with different levels of "granularity". That is, from one or more input objects, one could obtain different derived objects with different granularity levels. For example, from an input document ID_i, we can obtain its paragraphs DP_{i_j} and from those paragraphs we can obtain their sentences DS_{i,j_k} or their sentences without stopwords DS'_{i,j_k}. From a university IU_i we could derive its name DN_i or the set of its departments (each department will be a derived object) DD_{i_j}.
- **Atomic Objects** A_k. They are the basic objects of our model and are defined depending on the target application. They are obtained by decomposition of input objects I_i or derived objects D_j using derivation rules R_i. As they are atomic, they can not be used to produce new objects by decomposing them. It is mandatory in our model to define at least one type of atomic object and therefore the derivation rule needed to obtain it.

The tool must provide a user interface to allow defining all the object types for each of the three levels, and the derivation rules that applied over input and derived objects produce new objects (with lower granularity). Notice that it is not mandatory to define derived objects. The tool will provide basic algorithms to create the derivation rules, some of them accepting parameters, but the user can provide more complex algorithms to define other rules such as external procedures.

Regarding our multi-partite graphs, the mappings of the graph are:

- **Nodes.** Objects of the three different levels (input, derived and atomic) will be the graph nodes. Notice that the relationship between two nodes will usually be associated to the derivation rule that obtains one node from the other, and can be weighted.
- **Edges.** The relationships among nodes will be represented by the edges of the graph. Generally graph edges will link each derived or atomic object with the object in the previous level from which it has been derived, and can be weighted. Besides those basic edges, other edges are also possible to support some basic relationships such as "order relationships". For example, if we extract the words $AW_{i,j}$ of each document ID_i, the order among words is relevant to reproduce the text. Our tool will automatically create an edge among each atomic object A_{i_j} and the next one $A_{i_{j+1}}$ when they are derived from input object I_i that is a list of elements, such as documents. These links between atomic objects will not appear for sets that do not preserve an order.

Note that the use of a derivation rule over objects of a type to produce objects of a lower level, produce a structure that could be a graph or a tree depending of the definition of the rule. For example, to decompose documents ID_i into words, the rules can extract the list of words in the text (tree) or the vocabulary from the collection of documents (graph). That is, the rule for word extraction could extract each word from the ordered list of words in the document text AWT_{i_j}, linking each document with its words and the words among them by its position in the document. Another possibility would be for the rule to extract the words to the vocabulary of the document text AWV_j eliminating duplicates and linking each word with **all** the documents ID_i where it appears. In this second case the set of words is the vocabulary of the collection of documents, in the first case the set of words of each document represent the text of each document (in this way the graph will have all the document texts).

4 Use Cases

The objective of this section is to validate the model proposed in Section 3. In order to do so, we explain three currently relevant domains, and the useful rules that, in each case, would decompose and transform the original input objects I_i to obtain some useful derived D_j and atomic A_k objects. We will also present some useful queries that could be directly performed on the basic graph obtained with such decomposition to point out the utility of the multi-partite graph model we propose. We present them in order of difficulty.

4.1 Web Query Logs

Query logs are one of the main data streams that allow search engines to learn from users. Query logs can be used for many different purposes, including query recommendations, spelling correction or even extracting semantic relations [1,4]. Graphs have been proposed as a good way to exploit query logs for different purposes [2].

Query logs gather all the interaction of users with a search engine, including each query and the clicked pages. Assume that each query log has at least information about users (through anonymized ids), sessions (through cookies), queries, top k results (e.g. $k = 20$), and clicked answers. Then, we can define the following objects:

- IQ_i Each query log in the input data.
- DU_j Each different user in a query log.
- DS_k Each session of a user.
- DQ_ℓ Each query in a session.
- DR_m Each result in the answer to a query, including a Web page (URL), the text snippet and the rank of the result.
- $DURL_p$ Each clicked URL.
- AVW_n Each word in a query, snippet or URL text.

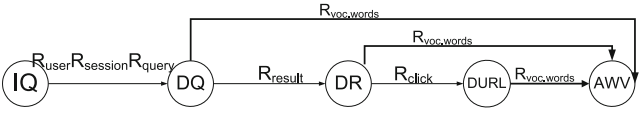


Fig. 1. Multi-partite graph model for Query log collections

To obtain those derived and atomic objects the rules could be:

- R_{user} Collect all the queries of the same user.
- $R_{session}$ Split the user queries in sessions.
- R_{query} Split the sessions in queries.
- R_{result} Collect all the results of a query in a session.
- R_{click} Extract all the results that were clicked in a given query-session pair. In addition, we could also have the non-clicked results.
- $R_{voc.word}$ Extract all the different words used by a given object, eliminating duplicates and creating the whole vocabulary.

Our derived and atomic objects can be generated using the rules described above, with a simple composition notation, as follows:

- $DU_j \leftarrow IQ_i \ R_{user}$
- $DS_k \leftarrow DU_j \ R_{session}$
- $DQ_\ell \leftarrow DS_k \ R_{query}$
- $DR_m \leftarrow DQ_\ell \ R_{result}$
- $DURL_p \leftarrow DR_m \ R_{click}$
- $AVW_n \leftarrow DQ_\ell \ R_{voc.word} \cup DR_m \ R_{voc.word} \cup DURL_p \ R_{voc.word}$

Notice that if we do not need so many derived objects, we could define a sequence of rules, like $DQ_\ell \leftarrow IQ_i \ R_{user} \ R_{session} \ R_{query}$. Figure 1 represent the abstract multi-partite graph model for this last possibility. Notice how only three different derived objects types are defined and that, from all of them the atomic object type AVW is derived with the rule $R_{voc.word}$. Clearly the atomic object typeAVW will be instanced with each one of the vocabulary words.

Figure 2 represents an instance of the previous graph model. The edges representing the derivation rule $R_{voc.word}$ are wider only for clarity. Using our planned tool, it would be possible to know which URL is the most visited counting the number of input edges it has. It is possible to know which queries produce some common results linking them through the result nodes DR and so on.

4.2 Bibliographic Databases

Bibliographic databases are in the center of research and technology progress. They are used to know the relationships among authors, recommend paper reviewers or assess authorities in a field [5]. Graphs have been proposed as a way to obtain information that goes beyond the simple list oriented keyword search answers [10].

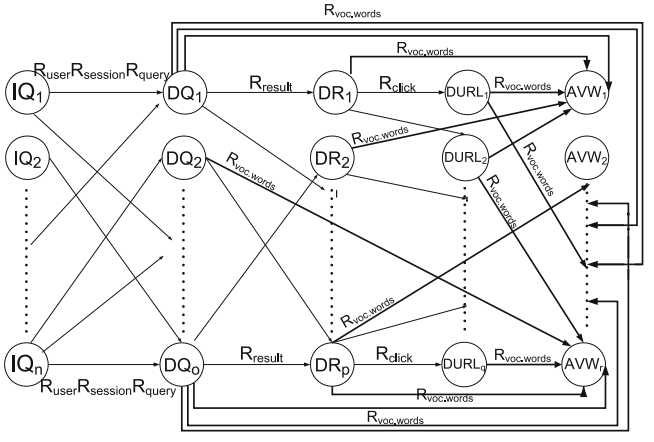


Fig. 2. Graph instance for Query log collections

A bibliographic database gathers all the information that relates authors and papers by means of the relationships among them, i.e. coauthoring relations and citations. Thus, a paper reference is a bibliographic item including the names of the authors and their affiliations and the citations to other papers. A schema of this information is shown in Figure 3 (left).

We consider the following objects:

- IB is the set of bibliographic items.
- DP_i is the derived element paper.
- AA_j are the atomic elements author.
- AC_k are the atomic elements citation.

As it can be deduced from the previous set of objects, authors and citations are atomic objects, and their relationships through papers allow for the creation of a

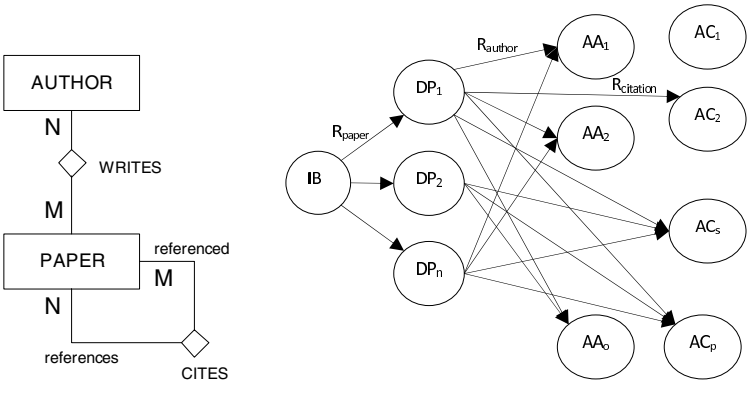


Fig. 3. Schema of a bibliographic database (left) and graph instance (right)

multi-partite graph that relates papers with their authors, and the papers they cite. To obtain those derived and atomic objects, the rules are:

- R_{paper} Collects all the papers from the set of bibliographic items, with a parameter "range years".
- R_{author} Collects the authors of each paper, creating a link with all the papers they coauthored.
- $R_{citation}$ Collects all the citations of a paper, creating a link with the papers that created it.

Our derived and atomic objects can be generated by using the rules in the following way:

- $DP_i \leftarrow IB\ R_{paper}[rangeyears]$.
- $AA_j \leftarrow DP_i\ R_{author}$.
- $AC_k \leftarrow DP_i\ R_{citation}$.

Figure 3 (right) represents an instance of the previous graph model.

However, the complexity of the multi-partite graph that could be generated from a set of references makes any query oriented to a specific set of authors also complex and time consuming.

Thus, for example, a social study of all the authors cited by papers in this century, say between 2000 and 2010, would be difficult to perform if the whole graph was obtained, not to mention the difficulty to generate the graph itself with such an amount of information and its relationships. So in this case the model would help us to reduce the scope of the mining.

Thus, given this example, we could take object IB and apply different rules R_j and the derivative objects to create the multi-partite graph. The rules are: R_{paper} that extract papers written between 2000 and 2010, $R_{citation}$ extracts citations and R_{author} extracts authors from a paper. In the resulting bi-partite graph we can perform queries to create a graph with just relationships among authors. Now the authors social network can be analyzed by just querying the multi-partite graph.

4.3 Structured Text

Text collections are useful in many application domains where they play different roles from information repository to tool for linguistic analysis. In each case the document collection is used with different algorithms. Our model allows different user rules to define different derived D_j and atomic A_k elements. This way, our model allows for the generation of basic multi-partite graphs from any application domain.

In the following example we assume that we have only one collection of XML documents where authors and title of each document are identified by specific labels. We also assume that each document has only one title but some authors. Lets suppose that those documents have geographic references and pictures with captions that are relevant in the application domain where the collection is going to be used. In this example some of the derived and atomic useful objects could be:

- *Input objects*
  - ID_i Each document in the collection.
- *Derived objects*
  - DT_i Document *Title*. That is, the sentence between the tags <title> < /title>
  - DC_k Each figure *Caption*, that is, sentences between tags <caption> and < /caption>.
  - DS_l Each *Sentence* between two consecutive "periods" of the document (final stops, dots, question mark, etc.).
  - DS'_m Each *Sentence* between two consecutive "periods" without stopwords. It means to check for each word in the sentence if it is in a stopword list.
- *Atomic objects*
  - ATW_{i_k} Each word of each document ID_i, that is, words will appear as many times as they do in the document text, and its order will be preserved to reproduce the text of each document ID_i.
  - AG_n Each toponym in the Gazetteer provided as parameter. That is, words starting with a capital letter that are in a gazetteer
  - AN_v Each author name, that is, sentence between the tags <author> and < /author>
  - AW'_4 Titles without stopwords.
  - AVW_i. Each word that appears in any document. They will just appear once, i.e. duplicates are eliminated. This is the vocabulary of the collection.

To obtain those derived and atomic objects the rules could be:

- $R_{extract}[tag]$ Extract sentences between pairs of tags provided as a parameter.
- R_{split} Cut the document into sentences. That is, it extract each portion of the document between two consecutive period marks.
- $R_{words-in-list}[list]$ Extract the successive words if they are in a specific list provided as parameter (such as stopwords or gazetteers in this example)
- $R_{text.words}$ Obtain the successive words of the given text.
- $R_{voc.word}$ Extract all the different words used by a given object, eliminating duplicates and creating the whole vocabulary.

Next we show how the derived and atomic objects can be generated using the described rules as follows:

- $DT_i \leftarrow ID_i\ R_{extract}[title]$.
- $DC_j \leftarrow ID_i\ R_{extract}[caption]$.
- $DS_j \leftarrow ID_i\ R_{split}$
- $DS'_j \leftarrow DS_i$ - $\{DS_i\ R_{words-in-list}[stopwords]\}$
- $ATW_{i_k} \leftarrow ID_i\ R_{text.words}$
- $AG_k \leftarrow ID_j\ R_{words-in-list}[gazetteer]$
- $AN_k \leftarrow ID_j R_{extract}[author]$.

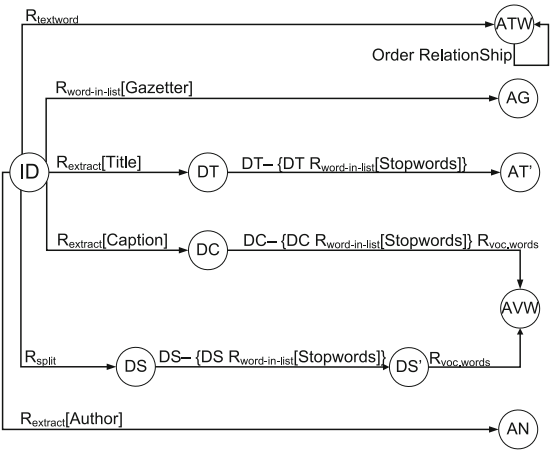


Fig. 4. Multi-partite graph model for XML documents

- $AT'_{j,i} \leftarrow DT_j$ - $\{DT_j R_{words-in-list}[stopwords]\}$
- $AVW_k \leftarrow (DC_i - \{DC_i\ R_{words-in-list}[stopwords]\})R_{voc.words}$
- $AVW_k \leftarrow DS_i\ R_{voc.words}$

As result we will obtain a graph where edges will be between each object and its derived objects of the different types. Notice that authors or toponyms will be linked with all the objects they are related to, but each word AW_{i_j} is only associated to one document ID_i and they also have *order edges* between each word AW_{i_j} and the next $AW_{i_{j+1}}$ in the text they came from. Figure 4 represent the abstract model of the multi-partite graph defined.

In a simpler application it could be possible to have different collections of plain text and the interest could be just to know the vocabulary they have. In this domain maybe only the rule $R_{voc.words}$ for vocabulary extraction would be necessary and the nodes would be only the documents (input objects ID_i) and the words included in the vocabulary of the collection (atomic objects AVW_k), where the set of AVW_k would represent the vocabulary of the collection.

Using any tool for graph management, any instance of the graph model presented in this example (Fig. 4), can be exploited and useful queries can be efficiently answered. For instance, it is possible to know which authors were coauthors in any document using the edges generated by the rule $IDR_{extract}[author]$, that is, we can link any author with any other author through those edges and the nodes for the input object document. In the same way, we can know which toponym appears the most, it requires to see which node AG (remember each one represents a atomic object toponym) has more edges.

This graph model can also be used in more sophisticated applications, for example it would be possible to detect plagiarism looking for sentences (nodes DS), with more than one input edge, that is, sentences that are equal in different input documents. In this way we could obtain documents that have some

common sentences. In the same way, using titles without stopwords (nodes AT) we can detect documents speaking about a similar topic.

In all those cases the graph representation of the collections of documents allow for a straightforward analysis of the data that would not be so efficiently performed over the XML document representation.

5 Final Remarks

We have presented a model to generate multi-partite graphs and show their applicability in different data sets. From those graphs, many other graphs can be obtained by applying standard queries over graphs. To use our generated graphs a tool for graph management such us [12] can be used. The tool we are developing will provide the generated graph in GML to make our graphs readable by any such tool, such that useful queries could be efficiently answer.

So, the next challenge is to design a powerful tool that is simple to use and it is based on our model. A first stage will be through a simple interface, while a second stage will include a graphical interface where nodes and edges will be easily represented.

Acknowledgements

The authors want to thank MICINN, the Ministry of Science and Innovation of Spain, for grants with numbers TIN2009-14560-C03-01, TIN2009-14560-C03-02 and TIN2009-14560-C03-03. The members of DAMA-UPC thank and Generalitat de Catalunya, for grant number GRC-1087.

References

1. Baeza-Yates, R.: Query usage mining in search engines. In: Scime, A. (ed.) Web Mining: Applications and Techniques. Idea Group, USA (2004)
2. Baeza-Yates, R.: Graphs from search engine queries. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 1–8. Springer, Heidelberg (2007)
3. Baeza-Yates, R., Brisaboa, N., Larriba-Pey, J.L.: Tin2009-14560-c03: High performance processing of large datasets represented as graphs
4. Baeza-Yates, R., Tiberi, A.: Extracting semantic relations from query logs. In: Berkhin, P., Caruana, R., Wu, X. (eds.) KDD, pp. 76–85. ACM Press, San Jose (2007)
5. Bibex: http://www.dama.upc.edu/bibex
6. Brandes, U., Pich, C.: GraphML transformation. In: Pach, J. (ed.) GD 2004. LNCS, vol. 3383, pp. 89–99. Springer, Heidelberg (2005)
7. Broder, A.Z., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J.L.: Graph structure in the web. Computer Networks 33(1-6), 309–320 (2000)
8. Chakrabarti, D., Faloutsos, C.: Graph mining: Laws, generators, and algorithms. ACM Comput. Surv. 38(1) (2006)

9. Cook, D.J., Holder, L.B.: Mining Graph Data. John Wiley & Sons, Chichester (2007)
10. Gómez-Villamor, S., Soldevila-Miranda, G., Giménez-Vañó, A., Martínez-Bazan, N., Muntés-Mulero, V., Larriba-Pey, J.L.: Bibex: a bibliographic exploration tool based on the dex graph query engine. In: EDBT, pp. 735–739 (2008)
11. Knowledge, S.: http://www.semantic-knowledge.com/tropes.htm
12. Martínez-Bazan, N., Muntés-Mulero, V., Gómez-Villamor, S., Nin, J., Sánchez-Martínez, M.A., Larriba-Pey, J.L.: Dex: high-performance exploration on large graphs for information retrieval. In: CIKM, pp. 573–582 (2007)
13. Neo4j, http://neo4j.org/
14. Pereira, Á., Baeza-Yates, R.A., Ziviani, N., Bisbal, J.: A model for fast web mining prototyping. In: WSDM, pp. 114–123 (2009)
15. Tong, H., Papadimitriou, S., Sun, J., Yu, P.S., Faloutsos, C.: Colibri: fast mining of large static and dynamic graphs. In: KDD, pp. 686–694 (2008)

A Fast Two-Stage Algorithm for Computing SimRank and Its Extensions

Xu Jia[1,2], Hongyan Liu[3], Li Zou[1,2], Jun He[1,2], and Xiaoyong Du[1,2]

[1] Key Labs of Data Engineering and Knowledge Engineering, Ministry of Education, China
[2] Department of Computer Science, Renmin University of China, 100872, China
{jiaxu,zouli_happy,hejun,duyong}@ruc.edu.cn
[3] Department of Management Science and Engineering, Tsinghua University, 100084, China
liuhy@sem.tsinghua.edu.cn

Abstract. Similarity estimation can be used in many applications such as recommender system, cluster analysis, information retrieval and link prediction. *SimRank* is a famous algorithm to measure objects' similarities based on link structure. We observe that if one node has no *in-link*, similarity score between this node and any of the others is always zero. Based on this observation, we propose a new algorithm, fast two-stage *SimRank* (*F2S-SimRank*), which can avoid storing unnecessary zeros and can accelerate the computation without accuracy loss. Under the circumstance of no accuracy loss, this algorithm uses less computation time and occupies less main memory. Experiments conducted on real and synthetic datasets demonstrate the effectiveness and efficiency of our *F2S-SimRank*.

Keywords: *SimRank*, similarity, random walk.

1 Introduction

Similarity measure can be used in many applications such as recommender system, cluster analysis, information retrieval, link prediction, and so on.

For example, the recommender system calculates a list of similar recommended products for the user and cluster analysis is the assignment of a set of items into subsets so that similar items are more likely in the same cluster.

A wide variety of similarity measures have been proposed in the past. Content-based analysis and link-based analysis are two major approaches for measure similarity. *Vector Space Model* [16] is one well-known model in content analysis, which can only detect explicit information. *Co-citation* [2] is a famous algorithm in link analysis, which aims to find related objects using citation structure among the objects.

SimRank [1] provides a better mechanism to discover more implicitly similar objects based on the intuition "two objects are similar if they link the similar objects". But it suffers from high computational cost.

A significant observation is that if there is a node without any *in-link*， the similarity score between this node and any of the others is always zero. Our idea is partly motivated by [12]. We propose a new algorithm named *F2S-SimRank* to accelerate the *SimRank* computation. First we begin with a simple example in Fig.1 to show how *F2S-SimRank* can save the computation time as well as the main memory.

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 61–73, 2010.

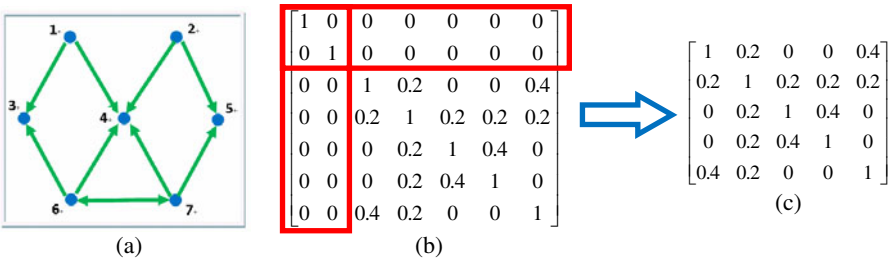


Fig. 1. A simple example for *SimRank* computation

The graph in Fig.1(a) contains seven nodes and ten directed edges. The matrix in Fig.1(b) stores the similarity scores between any two nodes calculated by *SimRank*. We observe that node 1 and 2 have no *in-link*, and after the *SimRank* computation, the similarity scores between node 1 and any of the other six nodes are all zeros as shown in the first row of the similarity matrix. In this situation, the size of similarity matrix is 7×7. In fact we do not need to store the first two rows and columns due to the characteristic of node 1 and 2. So we reserve all link information during the similarity computing process and reorder the nodes according to their *in-links*. But do not store the nodes with no *in-link* in the similarity matrix. At this time the matrix's size is 5×5 and we save approximately 50% of the memory in this case. All similarity scores in Fig.1(c) remain the same as shown in Fig.1(b), which means there is no accuracy loss using this method.

The nodes without any *in-link* are usually in two situations: 1) the nodes are newly added. There is no time for other resources to recognize them. So no resources refer to them; 2) the nodes are added maliciously. These nodes are added into the resource in order to promote some other nodes' similarity or importance maliciously.

In summary, our contributions in this paper are concluded as follows:

- Based on our observation, to avoid unnecessary computing, we propose a novel algorithm, *F2S-SimRank*, which can quickly calculate each *nodepair*'s similarity score and reduce main memory consumption.
- Our algorithm can be combined with other acceleration methods of *SimRank*.
- We conduct experiments on both real and synthetic datasets to evaluate the *F2S-SimRank's* performance and demonstrate its advantages.

The rest of this paper is organized as follows. Preliminaries are presented in Section 2. The details of the *F2S-SimRank* algorithm and the experimental evaluation are discussed in Section 3 and 4 respectively. Section 5 provides a brief survey of the related work. Finally we conclude the paper.

2 Preliminaries

In this section, we first give some definitions for key terms that will be used in the rest of this paper. Then we will review the algorithm *SimRank*.

2.1 Notations

Given a directed graph $G(V, E)$, where V is a set of n nodes and E is a set of edges linking nodes, we have the following terms.

Transfer probability matrix: Theoretically, let M be the link matrix, i.e., $M_{ij} = 1$ if there exists an edge pointing from node i to node j. Then, the transfer probability T_{ij} is equal to $(1/|O(i)|)M_{ij}$ and the value $T_{ij} \in [0, 1]$.
Adjacency table: This table contains n rows. The i^{th} *row* describes the link relationship between node i and any of other nodes.
In-link (Out-link): In the graph G, when there is a directed edge going from node A and pointing to node B, we say node B has an *in-link* and node A has an *out-link*. We also regard node B as an in-neighbor of A. For example, in Fig.1(a) node 6 has one *in-link* and two *out-links*.
CitedNode (No-CitedNode): we regard the node in the directed graph that has at least one *in-link* as a *citednode*. Nodes with no *in-link* are called *no-citednode*. Both node 1 and 2 are *no-citednodes* in Fig.1(a).

2.2 Review of *SimRank*

SimRank [1] is a classical method of measuring link-based similarity between objects in a graph that models the object-to-object relationships. The intuition behind *SimRank* is that "two objects are similar if they link to the similar objects".

We first present the formula to compute *SimRank* score. Given a graph $G(V, E)$ consisting of a set V of nodes and a set E of links, the *SimRank* score between objects a and b, denoted as $S(a, b)$, is computed recursively as follows:

$$S(a,b)=\begin{cases}1 & (a=b)\\ \dfrac{c}{|I(a)||I(b)|}\displaystyle\sum_{i=1}^{|I(a)|}\sum_{j=1}^{|I(b)|}S(I_i(a),I_j(b)) & (a\neq b)\end{cases} \tag{1}$$

where c is a decay factor, $c \in [0, 1]$. $I(a)$ is the set of in-neighbor of a and $I_i(a)$ is the i^{th} in-neighbor of a. $|I(a)|$ is the number of in-neighbors of a. In case when $I(a)$ or $I(b)$ is an empty set, $S(a, b)$ is specially defined as zero.

A solution to the *SimRank* equation (1) can be reached by iterations. For each iteration k, let $S_k(a, b)$ be the similarity score of *nodepair (a, b)* after k iterations. The iteration process is started with S_0 as follows:

$$S_0(a,b)=\begin{cases}0 & (\text{if } a\neq b)\\ 1 & (\text{if } a=b)\end{cases} \tag{2}$$

To calculate $S_{k+1}(a, b)$ from $S_k(a, b)$, we have the following equation:

$$S_{k+1}(a,b)=\frac{c}{|I(a)||I(b)|}\sum_{i=1}^{|I(a)|}\sum_{j=1}^{|I(b)|}S_k(I_i(a),I_j(b)) \tag{3}$$

In the equation (3), $1/|I(a)|$ is the single step probability of random walk from *node a* to a node in $I(a)$. Therefore we can use the transfer probability matrix T [14] to capture the single step probability in a Markov Chain. Thus, *SimRank* algorithm can be described by the matrix calculation.

$$S_0 = E \tag{4}$$

where E is an identity matrix. Equation (3) can be rewritten as follows:

$$S_k(a,b) = c \cdot \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} T_{aI_i(a)} \cdot T_{bI_j(b)} \cdot S_{k-1}(I_i(a), I_j(b)) \tag{5}$$

Although the convergence of iterative *SimRank* algorithm can be guaranteed in theory, practical computation uses a tolerance factor ε to control the number of iterations such that a finite number of iterations will be performed. It is recommended to set $\varepsilon = 10^{-4}$, the same as in *PageRank* [14]. Specifically, the ending condition of the iteration is as follows:

$$Max_{(a,b)} [\ |S_k(a, b) - S_{k-1}(a, b)| / S_k(a, b)\ ] \leqslant \varepsilon \tag{6}$$

It says that the iteration stops if the maximal change rate of similarity scores between two consecutive iterations for any *nodepair* is smaller than the threshold ε.

3 Fast Two-Stage *SimRank* Algorithm

In this section, motivation of *F2S-SimRank* is introduced first. Then, we will describe the details of algorithm *F2S-SimRank*. We will discuss implementation issue of *F2S-SimRank* in the end.

3.1 Motivation

Let us consider the results in Fig.1 again. There are some *no-citednodes* in the graph. After whole iterations by *SimRank* the similarity scores between these nodes and other nodes remain zeros. In this situation, the computation for these *nodepairs* is unnecessary. In the same time, some main memory is unnecessary consumed.

We preserve all link information among nodes in order to keep no accuracy loss. All similarity scores in Fig.1(c) compared with that in Fig.1(b) show that there is no accuracy loss. But the similarity matrix is smaller (in Fig.1(b) the similarity matrix is 7×7 whereas in Fig.1(c) the similarity matrix is 5×5) because we do not store all *no-citednodes*. The computation time is less because some unnecessary computation is avoided.

Next we will describe the details of our algorithm *F2S-SimRank*.

3.2 Algorithm

We propose an algorithm, which has two stages outlined as follows:

The first stage: Search for all *no-citednodes* and mark these nodes. Construct the adjacency table and reorder all the nodes in the adjacency table. Link information among all nodes is preserved.

The second stage: Store similarity matrix only for *citednodes*. Then the remaining computational process is the same as *SimRank*.

The major steps of the *F2S-SimRank* algorithm are shown below.

Algorithm. *Fast Two-Stage SimRank*

Input: Graph G(V, E), Decay Factor c, Tolerance Factor ε
Output: Similarity Matrix

```
      // The first stage
1  NodeList = ∅, k=0;
2  for i ← 1 to n do
3    if ( indegree[i] != 0 )
4      NodeList ←NodeList ∪{i} ;
5  NoCited-NodeList ← V− NodeList;

      // The second stage
6  build similarity matrix [NodeList.length][NodeList.length];
7  initialize similarity matrix [NodeList.length][NodeList.length];
8  Do
9    k ←k+1;
10   for i ← 1 to NodeList.length
11     for j ← 1 to NodeList.length
12       S_k(v_a, v_b) = c · 1/|I(v_a)| · 1/|I(v_b)| Σ_{i=1}^{|I(v_a)|} Σ_{j=1}^{|I(v_b)|} S_{k-1}(v_i, v_j)
13 While Max [ |S_k(a, b) - S_{k-1}(a, b)| / S_k(a, b) ] > ε
14 return S_k
```

In this algorithm, first we search for all *no-citednodes (line 1-3)*. The *citednodes* are added into the *NodeList* whereas other nodes are added into the *NoCited-NodeList (line 4-5)*. The nodes are reordered (we will explain this in section 3.4) and s*imilarity matrix* is initialized in (*line 6-7)*. The algorithm then uses Equation *(line 12)* to compute the similarity score for each *nodepair* (*a, b*). When the convergence condition is satisfied *(line 13)*, the iteration stops and the algorithm outputs the results *(line 14)*.

3.3 Analysis

We first analyze the time and space complexity of the algorithm *SimRank*. Suppose there are n nodes in the entire graph and only m of them are *citednodes*. We can easily observe that the space requirement for *SimRank* is $O(n^2)$ to store the similarity matrix. According to [1], let d_2 be the average of $|I(a)||I(b)|$ over all *nodepairs*. So the computational complexity of *SimRank* is $O(Kd_2n^2)$ using data structure of adjacency table. Here K represents the number of iterations. The time and space complexity of *F2S-SimRank* are $O(Kd_2m^2)$ and $O(m^2)$ respectively. For example, if $m \leq 0.7n$, we will save 50% of computation time and 50% of main memory approximately. *F2S-SimRank* preserves all link information in the computation, so there is no accuracy loss. In real applications there are usually some *no-citednodes*, so we can save both computation time and main memory by using *F2S-SimRank*.

Because *F2S-SimRank* only changes the order of nodes in the similarity matrix and ignores some unnecessary rows and columns, this method can be combined with other *SimRank* acceleration methods such as *Power-SimRank* [20], *SmallWorld-SimRank* [21] and *Adaptive-SimRank* [22].

The convergence of *F2S-SimRank* is the same as original *SimRank* because zero value doesn't affect the convergent results. *F2S-SimRank* contains all link information and ignores those unnecessary zero value. Therefore, in this situation the results are the same as that computed by *SimRank*.

The decay factor c controls the rate of the convergence because c affects the propagation of the similarity scores. When c is set to a small value, the propagation power of *SimRank* will be weakened and only nodes nearby can contribute in the structural similarity calculation. But when c is set high, more nodes can participate in the process of the recursive propagation. In this situation *SimRank* scores can be accumulated more easily and the convergence will need more computation time. Similarity scores only represent relative meaning instead of absolute meaning. In paper [1], the author pointed out that the *SimRank* has a rapid convergence (about five iterations). However, in our experiments some *nodepairs' SimRank* scores still have huge changes after five iterations. So we set $\varepsilon = 10^{-4}$.

F2S-SimRank works well when the graph contains some *no-citednodes*. The worst case is that all nodes in the graph have *in-links*. In this situation, the *F2S-SimRank* can save neither computational cost nor the main memory. Fortunately, many real datasets usually contain some *no-citednodes*.

3.4 Implementations

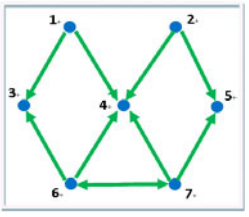
Because the real data is usually very sparse, we use the adjacency table instead of the traditional transfer probability matrix.

Transfer probability matrix was proposed for computation of *PageRank* [14] and many optimal techniques can be used in the transfer probability matrix. But the transfer probability matrix has some drawbacks when the data is sparse. For example, there is a graph containing 10K nodes and 50K edges. When we use the transfer probability matrix, 400M main memory was occupied. But when we use the adjacency table, only 200K main memory was consumed. Generally, the larger proportion of *no-citednodes* in the graph is, the more main memory can be saved. As far as we know, there is no author who uses the adjacency table earlier than us. The data structure of the adjacency table can dramatically save main memory and the computation time as well.

Adopting the data structure of adjacency table, the computation time drops dramatically compared with using transfer probability matrix (usually more than 10 times faster). Due to space limitation, we do not give the computation time by using these two data structures.

Let L be the average degree of all nodes, L_c be the average number of the nodes which have in-links, and L_{nc} be the average number of the nodes which have no *in-link*. In Fig.2 (a) and (b) it is easy to see that node 1 and 2 have no *in-link*. After the first stage of *F2S-SimRank*, the algorithm recognizes all *no-citednodes* and reorders the nodes as in Fig.2(c). The nodes which have *in-links* are presented first and other nodes are followed. By this way we can assure that all link information is preserved. At this time *no-citednodes* are not stored in the similarity matrix during the computation. The average computational cost for *nodepair's* similarity scores is shown in Table 1.

In *SimRank* the average computational cost for certain *nodepair* is L^2 and $L=L_c+L_{nc}$. In the second stage of *F2S-SimRank*, the *citednodes* are computed in the same way as used by *SimRank*. The computational cost for this part is L_c^2. Other nodes can be set



(a)

N1 →(null)
N2 →(null)
N3 →N1 →N6
N4 →N1 →N2 →N6 →N7
N5 →N2 →N7
N6 →N7
N7 →N6

(b)

N3 →N1 →N6
N4 →N1 →N2 →N6 →N7
N5 →N2 →N7
N6 →N7
N7 →N6
~~**N1** →(null)~~
~~**N2** →(null)~~

(c)

Fig. 2. A simple example for adjacency table

directly. If the similarity of a node with itself is one, otherwise is zero. The computational cost in this part is L_{nc}^2. If we optimize this process, the computational cost can be reduced to $L_{nc} \times L_{nc}$. So the cost of *F2S-SimRank* is $L_c^2 + L_{nc}^2$ (or $L_c^2 + L_{nc} \times L_{nc}$). For example, if $L_c : L_{nc} = 4:1$, the acceleration rate of *F2S-SimRank* is 32%.

Table 1. The average computational cost for a certain *nodepair*

SimRank	*F2S-SimRank*
$(L_c + L_{nc})^2$	$L_c^2 + L_{nc}^2$

In real applications, either sparse matrix or hash table can be applied as the core data structure to optimize the process. Because the graph can be so large that it cannot be stored in the main memory, any advanced data structure that can optimize external memory accesses can be used.

4 Experimental Evaluation

To measure the performance of *F2S-SimRank*, we conducted a series of experiments. The system used for experiments have 3GB RAM, a 3.0GHz Intel Core 2 Duo Processor, and ran Microsoft Windows Vista Ultimate Edition. We implemented all algorithms in java.

4.1 Datasets

Our experiments focus on three different datasets: ACM dataset, RUC dataset and Synthetic datasets.

ACM dataset: the dataset contains 14008 papers crawled from ACM CSS [18], which is a credible subject classification system for computer science. Each paper is considered as a node and citations are considered as directed edges.

RUC dataset: this dataset consists of 8107 web pages crawled from RUC [19], which contains three parts (*info.ruc.edu.cn, deke.ruc.edu.cn* and *news.ruc.edu.cn*) of web pages. Each web page becomes a node and each hyperlink becomes a directed edge.

Synthetic datasets: synthetic datasets contain two parts: ***SYN*** and ***SYN-X***.

SYN: this generated dataset follows the power-law [17] distribution because real data usually follow the power-law distribution. We use this dataset to test the performance of *F2S-SimRank*.

SYN-X: all of these datasets contain 10K nodes and 20K edges. ***X*** means there are ***X*** *citednodes* in the dataset (***X*** = 5K, 5.5K, 6K…10K). There are totally eleven datasets. We use these datasets to compare the performance between *F2S-SimRank* and *SimRank* as the proportion of *citednodes* changed.

Statistical details of these datasets are shown in Table 2.

Table 2. Statistical details of these datasets

Datasets	ACM	RUC	SYN	SYN-X
Nodes(n)	14008	8107	10000	10000
Edges(e)	13181	159818	19990	20000
CitedNodes	4705	4632	5460	5000~10000
Proportion	*33.58%*	*57.14%*	*29.81%*	*50%~100%*

From Table 2 we can see that the percentages of *citednodes* in datasets ACM, RUC and SYN are different. For example, there are 4632 *citednodes* in RUC dataset, accounting for 57.14% of the total 8107 nodes. The proportion of *citednodes* in the three datasets is 40.18% on average.

In our experiments we regard each dataset as a directed graph, consisting of a set of nodes with directed edges between nodes.

For simplicity and fairness of comparison, we set the decay factor $c = 0.8$ and $\varepsilon = 10^{-4}$ for both *SimRank* and *F2S-SimRank*. All the default value of parameters are set in accordance with [1].

4.2 Performance

First we compare the computation time and the main memory occupied between *SimRank* and *F2S-SimRank*.

Table 3. Runtime of the two algorithms

Datasets	*SimRank*	*F2S - SimRank*
ACM	3066.1s	*1637.8s*
RUC	94437.2s	*75036.7s*
SYN	822.9s	*318.5s*

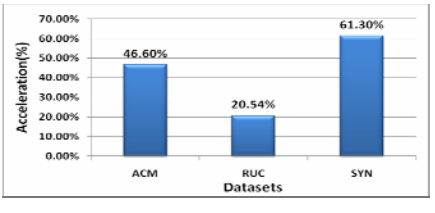


Fig. 3. Acceleration of the Computation

Table 3 shows the runtime of the two algorithms, and Fig. 3 illustrates the acceleration of the runtime by *F2S-SimRank* comparing with *SimRank*. We can see that *F2S-SimRank* saves 42.81% of computation time on average. For example, the computation time of *SimRank* on ACM is 3066.1s. In ACM dataset only 33.58% of nodes are *citednodes*. *F2S-SimRank* marks these nodes and only measures the similarity scores among these nodes. In this situation the computation time is reduced to 1637.8s. From Fig.3 we can see that 46.60% (≈(3066.1-1637.8)/3066.1) of the computation time can be saved in ACM dataset, nearly half of the original computation time.

Table 4. RAM Cost of the two algorithms

Datasets	*SimRank*	*F2S SimRank*
ACM	748.5M	*84.4M*
RUC	250.7M	*81.8M*
SYN	381.5M	*113.7M*

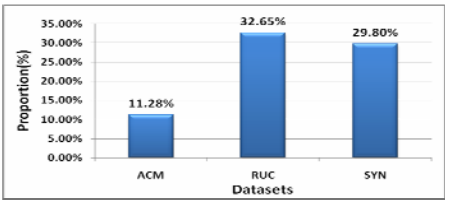


Fig. 4. Proportion of RAM Cost

The main memory consumed by the two algorithms is shown in Table 4 and the proportion of the memory cost by *F2S-SimRank* to that by *SimRank* is shown in Fig.4. We can observe that *F2S-SimRank* outperforms *SimRank* on all datasets. For example, there are 8107 nodes in the RUC dataset, the similarity matrix for *SimRank* is 250.7M (8107×8107) in memory and the adjacency table is only 623.9K (8107×19.7). Compared with the similarity matrix, the size of the adjacency table can be ignored. There are only 4632 *citednodes* in the RUC dataset. The similarity matrix for *F2S-SimRank* is 81.8M (4632×4632), 32.65% (≈81.8/250.7) of the original matrix size.

From Table 3 and Table 4 we can see that *F2S-SimRank* can reduce the computation time and save the main memory dramatically. When the graph cannot be held into the main memory for *SimRank*, *F2S-SimRank* will be more effective and efficient.

The main reason for the speedup is that *F2S-SimRank* ignores the *no-citednodes*. Unnecessary computations are avoided. The less the proportion of *citednodes* is, the less main memory will be consumed.

4.3 Extension

F2S-SimRank can be easily combined with other acceleration approaches such as *Power-SimRank* [20] and *SmallWorld-SimRank* [21](*SW-Simrank* for short). These two algorithms make use of the power-law distribution and the small world theory to optimize the original *SimRank* computation.

Table 5. Computation time of SW-SimRank VS. SW&F2S-SimRank

Datasets	*SW SimRank*	*SW&F2S SimRank*
ACM	743.6s	*378.3s*
RUC	43196.6s	*36892.5s*
SYN	396.7s	*159.5s*

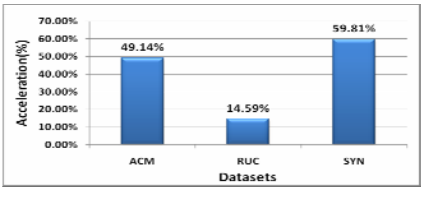


Fig. 5. Acceleration of *SW&F2S-SimRank*

Both *SW-SimRank* and *Power-SimRank* can accelerate the *SimRank* computation. *F2S-SimRank* does not change the characteristic of the *SimRank*. So we can easily combine it with *SW-SimRank* and *Power-SimRank* to speed up the computation further. Table 5 and Table 6 show the computation time of *SW-SimRank, SW&F2S-SimRank, Power-SimRank* and *Power&F2S-SimRank*. From Fig.5 and Fig.6 we can

Table 6. Computation time of *Power-SimRank* VS. *Power&F2S-SimRank*

Datasets	*Power SimRank*	*Power&F2S SimRank*
ACM	681.4s	*333.2s*
RUC	52798.9s	*30386.6s*
SYN	429.3s	*162.8s*

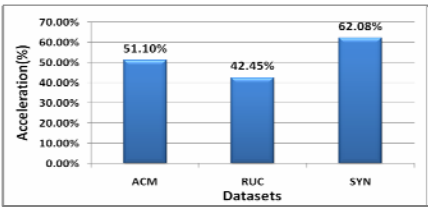


Fig. 6. Acceleration of *Power&F2S-SimRank*

observe that the process of computation is even faster when *F2S-SimRank* is pluged into the *SW-SimRank* and *Power-SimRank*. For example, the computation time of *Power-SimRank* in ACM is 681.4s. When we combine *F2S-SimRank* with *Power-SimRank*, more than half of the computation time can be saved.

4.4 The Worst Case

In the end, we want to discuss the worst case of algorithm *F2S-SimRank*.

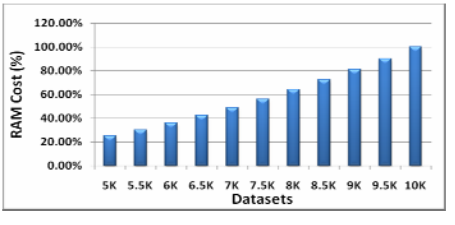


Fig. 7. RAM Cost of *F2S-SimRank*

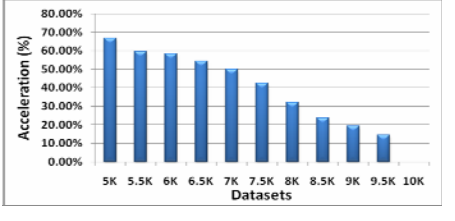


Fig. 8. Acceleration of *F2S-SimRank*

From Fig.7 and Fig.8 we can see that *F2S-SimRank* works well when the graph contains some *no-citednodes*. The eleven datasets all have 10K nodes and 20K edges. The number of *citednodes* in these datasets is varied from 5K to 10K. For example, there are 8K *citednodes* in dataset ***SYN-8K***, for which the main memory cost is only 64%, and the computation time of *SimRank* and *F2S-SimRank* are 1475.4s and 1003.3s respectively. Hence, 32.0% of the computation time can be saved.

We can also know from these figures that the more the proportion of *citednodes* is in the graph, the less the performance speed up. In our experiments, we find that the performance of *F2S-SimRank* does not keep very well when the graph becomes to contain only few *no-citednodes*. Thus, *F2S-SimRank* doesn't have a great advantage in this kind of graphs. When all nodes in the dataset are *citednodes*, the computation time and the main memory consumed by *F2S-SimRank* are nearly the same as *Sim-Rank* because the computation time of *F2S-SimRank* in the first stage can be ignored compared with the second stage.

5 Related Work

We categorize the related existing research work into three aspects below.

Link-based similarity evolution: In bibliometrics the most noteworthy work includes *Co-citation* [2], *Co-coupling* [3] and *Amsler* [4]. *Co-citation* means if two documents are often cited together by other documents, they may focus on the same topic. *Co-coupling* means if two papers cite many papers in common, they may focus on the same topic. *Amsler* combines *Co-citation* and *Co-coupling* methods to determine similarity. However, all these methods only considered their immediate neighbors. *SimRank* [1] is proposed to consider the entire graph to determine the similarity between nodes iteratively. Many iterative algorithms over web link graph, such as *PageRank* [13] and *HITS* [15], have been applied and studied to compute the importance of the web pages.

Extension and acceleration of SimRank Computation: *SimRank++* [11] extends the *SimRank* by adding weights on the edges and considering the number of neighbors. *P-Rank* [24] enriches the *SimRank* by jointly encoding both *in-link* and *out-link* relationships into structural similarity computation. According to the strategy of *PageRank* score propagation, *PageSim* [8] is capable of measuring similarity between any two web pages. Lizorkin et al [10] presented some optimization techniques which reduce the computational complexity from $O(n^4)$ to $O(n^3)$. Our research group proposed *Power-SimRank* [20], *SW-SimRank* [21], *Adaptive-SimRank* [22], and *Block-SimRank* [23] to improve *SimRank* computational performance based on some characteristics of link graph such as the power-law distribution and the hidden block structure.

Other similarity computation algorithms: Yin [7] proposed *SimTree* to represent similarity between objects and developed an effective similarity calculation algorithm *LinkClus*. *SimFusion* [6] aimed at "combining relationships from multiple heterogeneous data sources". Random walk with restart provides a good relevance score between two nodes in a weighted graph and [9] proposed a fast solution to solve this problem. Blondel et al [25] introduced a new similarity matrix and explained how to associate a score with the similarity of vertices of two graphs A and B. The similarity matrix can be obtained as $S_{k+1}=BS_kA^T+B^TS_kA$. Fogoras's method [5] was presented in a general framework of Monte Carlo similarity search algorithms.

6 Conclusion and Future Work

How to calculate similarity between objects is a classical problem in computer science and has a wide range of applications. *SimRank* can measure objects' similarities based on link structure but suffers from high computational cost. We present a fast two-stage algorithm for calculating *SimRank*. The algorithm exploits the following observation: if one node has no *in-link*， the similarity scores between this node and any other node is always zero. In the first stage, the focus is on searching for the *no-citednodes*; in the second stage, the algorithm only compute similarity score for the *citednodes*. Extensive experiments demonstrate that *F2S-SimRank* is efficient in practice. Possible directions for future studies include analyzing the *no-citednodes* in details and setting different weights to these nodes.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grant No. 70871068, 70890083 and 60873017.

References

1. Jeh, G., Widom, J.: SimRank: A Measure of Structural-Context Similarity. In: SIGKDD, pp. 538–543 (2002)
2. Small, H.: Co-citation in the scientific literature: A new measure of the relationship between two documents. Journal of the American Society for Information Science 24(4), 265–269 (1973)
3. Kessler, M.M.: Bibliographic coupling between scientific papers. American Documentation 14(1), 10–25 (1963)
4. Amsler, R.: Applications of citation-based automatic classification. Linguistic Research Center (1972)
5. Fogaras, D., Racz, B.: Scaling link-based similarity search. In: WWW, Chiba, Japan, pp. 641–650 (2005)
6. Xi, W., Fox, E.A., Zhang, B., Cheng, Z.: SimFusion: Measuring Similarity Using Unified Relationship Matrix. In: SIGIR, Salvador, Brazil, pp. 130–137 (2005)
7. Yin, X.X., Han, J.W., Yu, P.S.: LinkClus: Efficient Clustering via Heterogeneous Semantic Links. In: VLDB, Seoul, Korea, pp. 427–438 (2006)
8. Lin, Z.J., King, I., Lyu, M.R.: PageSim: A Novel Link-Based Measure of Web Page Similarity. In: Edinburgh, W.W.W. (ed.) WWW, Edinburgh, Scotland, pp. 1019–1020 (2006)
9. Tong, H.H., Faloutsos, C., Pan, J.Y.: Random walk with restart: fast solutions and applications. In: ICDM, Hong Kong, China, pp. 613–622 2006)
10. Lizorkin, D., Velikhov, P., Grinev, M., Turdakov, D.: Accuracy Estimate and Optimization Techniques for SimRank Computation. In: VLDB, Auckland, New Zealand, pp. 422–433 (2008)
11. Antonellis, I., Garcia-Molina, H., Chang, C.C.: SimRank++: Query rewrite through link analysis of the click graph. In: VLDB, Auckland, New Zealand, pp. 408–421 (2008)
12. Chris, P.L., Gene, H.G., Stefanos, A.Z.: A Fast Two-Stage Algorithm for Computing PageRank and Its Extension. Technical Report SCCM 2003-15, Stanford University (2003)
13. Page, L., Brin, S.: The anatomy of a large-scale hypertextual web search engine. Comput. Netw. ISDN Syst. 30(1-7), 107–117 (1998)
14. Langville, A.N., Meyer, C.D.: Deeper Inside PageRank. Internet Mathematics, 335–400 (2004)
15. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM 46(5), 604–632 (1999)
16. Wong, S.K.M., Ziarko, W., Wong, P.C.N.: Generalized vector spaces model in information retrieval. In: SIGIR, Montreal, Canada, pp. 18–25 (1985)
17. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On Power-Law Relationships of the Internet Topology. In: SIGCOMM, Cambridge, MA, USA, pp. 251–262 (1999)
18. ACM dataset, `http://www.acm.org/`
19. RUC dataset, `http://www.ruc.edu.cn/`
20. Cai, Y.Z., Cong, G., Jia, X., Liu, H.Y., He, J.: Efficient Algorithms for Computing Link-based Similarity in Real World Networks. In: Perner, P. (ed.) Advances in Data Mining. Applications and Theoretical Aspects. LNCS, vol. 5633. Springer, Heidelberg (2009)
21. Jia, X., Cai, Y.Z., Liu, H.Y., He, J., Du, X.Y.: Calculating Similarity Efficiently in a Small World. In: Huang, R., Yang, Q., Pei, J., Gama, J., Meng, X., Li, X. (eds.) Advanced Data Mining and Applications. LNCS, vol. 5678, pp. 175–187. Springer, Heidelberg (2009)
22. Cai, Y.Z., Liu, H.Y., He, J., Du, X.Y., Jia, X.: An Adaptive Method for Efficient Similarity Calculation. In: Chen, L., Liu, C., Liu, Q., Deng, K. (eds.) Database Systems for Advanced Applications. LNCS, vol. 5667, pp. 339–353. Springer, Heidelberg (2009)

23. Li, P., Cai, Y.Z., Liu, H.Y., He, J., Du, X.Y.: Exploiting the Block Structure of Link Graph for Efficient Similarity Computation. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 389–400. Springer, Heidelberg (2009)
24. Zhao, P.X., Han, J.W., Sun, Y.Z.: P-Rank: a comprehensive structural similarity measure over information networks. In: CIKM, Hong Kong, China, pp. 553–562 (2009)
25. Blondel, V.D., Gajardo, A., Heymans, M., Senellart, P., Dooren, P.V.: A Measure of Similarity between Graph Vertices: Applications to Synonym Extraction and Web Searching. SIAM Review 46(4), 647–666 (2004)

Hotspot District Trajectory Prediction*

Hongjun Li[1,2], Changjie Tang[1], Shaojie Qiao[3], Yue Wang[1], Ning Yang[1], and Chuan Li[1]

[1] Institute of Database and Knowledge Engineering, School of Computer Science, Sichuan University, Chengdu, 610065, China
[2] School of Computer Science, South West University of science and technology, Sichuan Mianyang, 621010, China
[3] School of Information Science and Technology, Southwest Jiaotong University, Chengdu, 610031, China
{achilee.lihj,chjtang.scu}@gmail.com

Abstract. Trajectory prediction (TP) of moving objects has grown rapidly to be a new exciting paradigm. However, existing prediction algorithms mainly employ kinematical models to approximate real world routes and always ignore spatial and temporal distance. In order to overcome the drawbacks of existing TP approaches, this study proposes a new trajectory prediction algorithm, called **HDTP** (**H**otspot **D**istinct **T**rajectory **P**rediction). It works as: (1) mining the hotspot districts from trajectory data sets; (2) extracting the trajectory patterns from trajectory data; and (3) predicting the location of moving objects by using the common movement patterns. By comparing this proposed approach to E^3TP, the experiments show HDTP is an efficient and effective algorithm for trajectory prediction, and its prediction accuracy is about 14.7% higher than E^3TP.

Keywords: Trajectory patterns, Trajectory prediction, moving objects database, hot districts.

1 Introduction

With the rapid developments of the wireless and mobile technologies, People can collect a large volume of trajectory data from moving objects such as vehicles or human. These data contain the instant information of moving objects, such as the current location and historical location. This paved the way for the deployment of location-based services and applications including: location aware information retrieval, emergency services, location-based billing, or moving objects tracking [1].

Usually, the position information of moving objects is periodically delivered to the server and stored in database. However, the location of a mobile object is often

* Supported by the National Science Foundation of China under Grant No.60773169,the 11th Five Years Key Programs for Sci. and Tech. Development of China under grant No.2006BAI05A01, the National Science Foundation for Post-doctoral Scientists of China under Grant No.20090461346.

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 74–84, 2010.

unknown for a long period of time due to some unavoidable factors, (signal congestions, signal losses due to natural phenomena, the power supply shortage of the mobile device, etc)[1].The location-based services and applications need to know the approximate position of a moving object in order to work. Examples of such services include navigational services, traffic management and location-based advertising. Thus, it is important to predict the position of a moving object at a given time or in a given area instantly with acceptable accuracy.

Accurate trajectory prediction of moving objects is challenging for the following reasons [10]: (1) the location prediction mechanism must guarantee to return accurate location of moving objects while not requiring extensive computation; (2) the performance of prediction should not fall drastically as the number of objects explodes; (3) prediction efficiency is as important as well, since the success of a location-based service depends on whether the service is delivered to an object at a particular location and on some particular time [2]. Specifically, if the objects often change their speed or directions, the approach of prediction should give a quick response while the objects still reside at a certain location.

Many algorithms have been proposed to predict the location of moving object [2, 3, 4]. But current approaches have some drawbacks, e.g. they do not utilize historical data, and their calculation cost is quite high that cannot scale up with the number of objects.

In order to predict the location of moving objects, we propose a **H**otspot **D**istrict **T**rajectory **P**rediction algorithm (**HDTP**). The original contributions of this paper include:

- Propose a method to mining hotspot districts which can be the common movement patters hide in real-world scenes.
- Extract the common movement patterns by the trajectory pattern mining algorithm developed in [5].
- Propose a novel trajectory prediction algorithm using the common movement patterns.
- Conduct sufficient experiments to evaluate the efficiency and effectiveness of our algorithms and compare them with the E^3TP method [6].

The rest of this paper is organized as follows. Section 2 surveys the related work. Section 3 briefly presents hotspot districts mining algorithm. Section 4 introduces the extracting of the common movement patterns. Our prediction algorithm is presented in section 5. Section 6 describes performance studies and experimental results. Finally, Section 7 concludes the paper and outlines the directions for future work.

2 Relate Work

The basic concepts and the related works of the location predicting of moving objects include frequent sequential pattern (FSP), Temporally Annotated Sequences (TAS) and trajectory prediction.

(1) Frequent Sequential Pattern mining [5]. The frequent sequential pattern (FSP) problem is defined over a sequential database *D*, where each element of each sequence is a time-stamped set of items. Time-stamps determine the order of elements in the sequence. E.g., a database may contain the sequences of visits of customers to a

supermarket, each visit being time-stamped and represented by the set of items bought together. Then, the basic idea of the FSP problem is to find all the sequences that are frequent in D, i.e., appear as subsequence of a large percentage of sequences of D. A sequence $\alpha=\alpha_1\rightarrow\alpha_2\rightarrow\ldots\ldots\rightarrow\alpha_k$ is a subsequence of $\beta=\beta_1\rightarrow\beta_2\rightarrow\ldots\ldots\rightarrow\beta_m$ ($\alpha\prec\beta$) if there exist integers $1\leq i_1<\ldots\ldots<i_k\leq m$ such that $\forall 1\leq\cdot n\leq\cdot k$, $\alpha_n\subseteq\beta_{in}$. Then we can define the support $supp_D(S)$ of a sequence S as the percentage of transactions $T\in D$ such that $S\prec T$, and say that S is frequent w.r.t. threshold s_{min} if $supp_D(S)>s_{min}$.

Several algorithms were proposed to efficiently mine sequential patterns. In [7] PrefixSpan employees the method of database projection to make the database for next pass much smaller and consequently make the algorithm more speedy. SPADE [8],that is a method employing efficient lattice search techniques and simple joins that needs to perform only three passes over the database.

(2) Temporally Annotated Sequences. Temporally annotated sequences (TAS), introduced in [5], are an extension of sequential patterns that extend sequences with information about the typical transition times between their elements. TAS's have the following form: $s_0 \xrightarrow{a1} s_1 \xrightarrow{a2} \ldots \xrightarrow{an} s_n$(, it represented as a couple $T=(S,A)$ of a sequence $S=<s_0,\ldots,s_n>$ with temporal annotations $A=<a_1,\ldots,a_n>$. In [5], an efficient prefix projection-based algorithm for extracting frequent TAS's was provided, that interleaves prefix extension steps and detection of frequent annotations for each prefix, exploiting the monotonicity properties of density over the annotation space and the relations between frequent sequences and frequent TAS's.

(3) Trajectory Prediction. There are several studies that address the problem of predicting future locations of moving objects. Most of them use a model based on frequent patterns and association rules and define a trajectory as an ordered sequence of locations, where the time is used as a time-stamp. In [2, 9] Morzy introduces a new method for predicting the location of a moving object. In particular, the method extracts association rules from the moving object database. Then, gives a trajectory of a moving object, by matching functions it selects the optimal association rule to match the trajectory for the prediction.

In [10] the authors propose a prediction algorithm, called PutMode (Prediction of uncertain trajectories in Moving objects databases) for predicating uncertain trajectories. It works by modeling uncertain trajectories by constructing TCTBNs and combining three important variables (i.e., street identifier, moving speed, and moving direction) to predict possible motion trajectories. In [6] the authors propose a general trajectory prediction algorithm called **E^{3}TP** (an **E**ffective, **E**fficient, and **E**asy **T**rajectory **P**rediction algorithm). The algorithm is with four key steps: mining "hotspot" regions from moving objects databases; discovering frequent sequential routes in hotspot areas; computing the speed of a variety of moving objects; and finally predicting the dynamic motion behaviors of objects. But these methods are based on the notions of support and confidence and do not consider any notion of spatial and temporal distance.

3 Mining Hotspot Districts

This section first presents some terms, then introduces the algorithm to mine hotspot districts.

3.1 Preliminary

Definition 1(Trajectory). [10] A trajectory of a moving object is a sequence of triples:

$$S = \{(x_1, y_1, t_1) \ldots (x_i, y_i, t_i) \ldots (x_n, y_n, t_n)\} \quad (1)$$

where t_i is a time stamp, $\forall i \in [1, n-1]$, $t_i < t_{i+1}$, and (x_i, y_i) represents the 2D coordinates.

Definition 2 (Stay point). [11] A stay point s represents a geographic district where a moving object stayed over a certain time interval. The extraction of a stay point depends on two scale parameters, a time threshold (T_{threh}) and a distance threshold (D_{threh}).Below is the formal definition of stay point.

Given a group of consecutive trajectory points $P=\{(x_m, y_m, t_m), (x_{m+1}, y_{m+1}, t_{m+1}) \ldots (x_n, y_n, t_n)\}$, D_{threh} and T_{threh}, if $\forall m \leq i < n$, $Distance((x_i, y_i), (x_{i+1}, y_{i+1})) \leq D_{threh}$ and $| t_n - t_m | \geq T_{threh}$. Then we say $(x, y, t_{start}, t_{end})$ is a stay point w.r.t. P, where

$$x = \sum_{i=m}^{n} x_i / | P | \quad (2)$$

$$y = \sum_{i=m}^{n} y_i / | P | \quad (3)$$

(x, y) represent the coordinates of stay point s, and $t_{start} = t_m$ and $t_{end} = t_n$ represent a user's start and end times in s respectively.

Figure 1 gives an example about stay point. Typically, these stay points occur in: (a) an individual remains stationary exceeding a time threshold, (b) a user wanders around within a certain geospatial range for a period. Each stay point stands for the location that a user is interesting in.

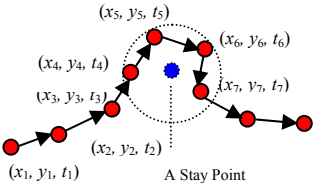


Fig. 1. A trajectory and a stay point

Definition 3 (Hot Districts). Given a region $R=\{s_i, \ldots\ldots s_j\}$ that consist of stay points and a density threshold (N_{threh}), R is a hot district when $Density_R \geq N_{threh}$,where $Density_R$ is the density of region R.

Hot Districts represents the interesting regions. Thus, a hot district should include more stay points as possible. This study uses DBSCAN to cluster the stay points and get hot districts.

3.2 Mining Hot Districts

Algorithm 1 gives a formal description of mining the hot districts.

In Algorithm 1, as each trajectory *traj* in *D*, the function StayPointDetection is called to extract stay points. It seeks the spatial regions where the user spent a period exceeding a certain threshold and puts these stay points to a set *SP*. Then DBSCAN is applied to cluster the stay points and finally output hot districts. In this paper, the value of N_{threh} is $|SP|/|HD|$, $|SP|$ and $|HD|$ are the length of *SP* and *HD* respectively.

Algorithm 1. MiningHotDis(D, T_{threh}, D_{threh}, N_{threh})

Input: a trajectory data set D, a time threshold T_{threh}, a distance threshold D_{threh}, a density threshold N_{threh}.
Output: a set of hot districts $R=\{R_1,\ldots\ldots,R_{num}\}$

1. $SP \leftarrow \varnothing$; // the set of stay points
2. $R \leftarrow \varnothing$;
3. for each $traj \in D$ do
4. $SP \leftarrow SP \cup$ StayPointDetection($traj$, T_{threh}, D_{threh});
5. $HD \leftarrow$ DBSCAN(SP);
6. for each cluster $c \in HD$ do
7. if $Density_c >= N_{threh}$
8. $R \leftarrow R \cup c$;
9. output R;

4 Trajectory Pattern Mining

Trajectory pattern represents a set of individual trajectories sharing the property of visiting the same sequence of places with similar time intervals [12].Each trajectory pattern extracted from trajectory data sets is a description of frequent motion patterns, in terms of both space (i.e. the regions of space visited during movements) and time (i.e. the duration of movements). For example, the pattern *Air Port* $\xrightarrow{[30:40]\min}$ *Museum* $\xrightarrow{[120:150]\min}$ *Hotel* may be a typical behavior of tourists that spend 30 to 40 minutes to reach museum from air port and spend about two hours to two and a half hour before arriving hotel. The definition of trajectory pattern is as follow:

Definition 4 (Trajectory Pattern). [12] A trajectory pattern is a pair *(S,A)*, where $S=<R_0,\ldots,R_n>$ is a sequence of regions, and $A=<a_1,\ldots a_n>$ is the temporal annotation of the sequence, a_i is a time interval. Trajectory pattern *(S,A)* can also represent as $R_0 \xrightarrow{a1} R_1 \xrightarrow{a2} \ldots \xrightarrow{an} R_n$.

The key step of Trajectory Pattern mining is to mine frequent TAS (temporally-annotated sequences). In [5] the authors introduce an efficient algorithm to extract frequent TAS from trajectory with respect to two thresholds δ and τ: the former represents the minimum support and the latter a temporal tolerance. In order to extract trajectory patterns, we propose a trajectory pattern mining algorithm that consists of two steps. The two steps are as follows: (1) trajectory preprocessing, each

trajectory is transformed location history that is represented as a sequence of hot districts through this process. (2) Mining frequent TAS using the algorithm introduced in [5]. Algorithm 2 describes the detail of mining trajectory patterns.

Algorithm 2. MiningTPatterns(D, R, δ, τ)

Input: a trajectory data set D, a set of hot districts R, a minimum support threshold δ, a temporal threshold τ
Output: a set of trajectory patterns TP.

1. $T \leftarrow$ Translate(D,R); // the input trajectories are preprocessed to sequences of hot districts.
2. $TP \leftarrow$ TAS_mining(T, δ, τ); ([5])
3. output TP;

Algorithm 3. Translate(D, R)

Input: a trajectory data set D, a set of hot districts R.
Output: a set of sequences of hot districts T.

1. $T \leftarrow \varnothing$;
2. for each $traj \in D$ do
3. $t \leftarrow \varnothing$; // t is the corresponding sequence of hot districts of $traj$.
4. for each $(x_i, y_i, t_i) \in traj$ do
5. $Region \leftarrow \varnothing$;
6. for each $R_i \in R$ do
7. if (x_i, y_i) is inside R_i
8. $Region \leftarrow R_i$;
9. break;
10. $t \leftarrow t \cup (Region, t_i)$;
11. $t' \leftarrow$ merge(t); // merge the consecutive trajectory points.
12. if the num of null region in t' is not beyond threshold r_{null}.
13. $T \leftarrow T \cup$ DelNullRegion(t'); // delete the null region points in t' and add it to T.
14. output T;

Algorithm 3 is applied to translate trajectory to sequence of hot districts. Lines 4 to 10 are employed to check the hot district which each trajectory point belongs to. If there is no corresponding hot district R_i in R for the trajectory point, the region is *null*. Line 11 merges the consecutive trajectory points if the regions of two points are the same. After that, the trajectory t is translated to t'. t' is the sequence of the couple (R_i,t_i), (R_i, t_i) implies that the moving object is inside the region R_i at time t_i. Because R may be *null*, so here, we only consider the trajectories with num of null regions greater than the threshold r_{null}. And the threshold r_{null} is equal to 1/3 of the length of t'. Line 13 deletes the null region points from t' and add it to T.

The outputs of Algorithm 2 are trajectory patterns. An example of a trajectory pattern is shown in Figure 2, where *A*, *B* and *C* represent the different hot districts respectively. (*C,17*) is the support of the pattern $A \rightarrow B \rightarrow C$ is 17. [105,137] represents the time interval spent from Region *B* to Region *C*.

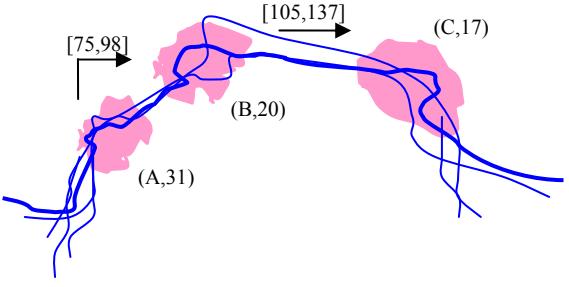


Fig. 2. An example of Trajectory pattern

5 Location Prediction

The goal of location prediction is as follows: given a trajectory data set *D* and a new trajectory *T*, it predicts the next location of *T*. This study extracts the trajectory patterns from the trajectory data set *D*, and finds the best trajectory pattern that match the given new trajectory *T*. and applies to predict the next location of *T*. Our prediction algorithm **HDTP** is described in Algorithm 4.

Algorithm 4. HDTP(D, T_{threh}, D_{threh}, N_{threh} ,δ, τ, T)

Input: a trajectory data set *D,* a time threshold T_{threh}, a distance threshold D_{threh}, a density threshold N_{threh}. a minimum support threshold δ, a temporal threshold τ , a given new trajectory *T* that need to predict the next location.

Output: a set of trajectory patterns *TP*.

1. $R \leftarrow$ MiningHotDis(D, T_{threh}, D_{threh}, N_{threh});
2. $TP \leftarrow$ MiningTPatterns(D, δ, τ);
3. $BestTP \leftarrow$ FindBestTrajPatt(T, TP);
4. output *BestTP*;

In Algorithm 4, the key step is used to find the best trajectory pattern that matches the new trajectory *T*. To find out the best trajectory pattern, Algorithm 4 checks whether the end point of *T* is in a hotspot district R_i. If so, it calculates the scores of all the trajectory patterns that contain R_i relative to *T*. Before computing the score of trajectory pattern, it computes the score for each hotspot district of this pattern from R_i to its first hotspot district. The score of each hotspot district is called punctual score [13].

5.1 Punctual Score

Let $RScore_B$ is the punctual score of hotspot district B w.r.t trajectory T. $RScore_B$ indicates the goodness of hotspot district B w.r.t. T. It measures the possibility that a moving object has already reached C had passed by B by trajectory T.(C and B are the consecutive hotspot districts of a trajectory pattern).Note that the moving object that follows the trajectory T. We introduce the notation of $Window_B$ to identify a location window where the moving object *was located* before the time interval specified in the edge towards C. For example, in Figure 2 $Window_B$ is location window where the moving object that current location is C was located before the time interval [105,137] that specified in the edge towards C.

Then we define the punctual score of hotspot district B according to the spatio-temporal distance between $Window_B$ and the spatial region of hotspot district B. There are three different possible cases in this study:

Case 1: the $Window_B$ intersects the region of hotspot district B. The punctual score $RScore_B$ is equal to the support value of hotspot district B.

Case 2: the $Window_B$ enlarged by temporal tolerance *tht* intersects the region of hotspot district B. The punctual score $RScore_B$ is the support value of the node divided by 1+*tht*.In this paper, the value of *tht* is equal to the time interval.

Case 3: the $Window_B$ enlarged by temporal tolerance *tht* does not intersect the region of hotspot district B. the punctual score $RScore_B$ is 0 since the trajectory is too far away from the region to be considered acceptable.

5.2 Trajectory Pattern Score

In this study, the score of a trajectory pattern is based on the value of the punctual score for each region of the pattern. This study applies the sum score as score function. Given a trajectory tr, a trajectory pattern TP, $R=\{R_1,R_2,\ldots R_n\}$ is the regions of TP. Let $sumScore(tr,TP)$ be the score of TP. then $sumScore(tr,TP) = \sum_{k=1}^{n} RScorek$. Then, we treat the pattern that has the maximum trajectory pattern score to be the most possible trajectory.

6 Experiments

6.1 Experimental Setting

This section reports the experiments by comparing HDTP with E^3TP prediction algorithm introduced in [6]. Basically, the E^3TP approach mines the frequent trajectory based on FP-tree. It ignores the time interval where moving objects stayed in a region. All algorithms are implemented in Java and the experiments are conducted on a PC equipped Intel(R) Dual E2160 1.80GHz CPU, 2.0G RAM, and running under Windows XP professional system.

All experiments are run on the following data sets generated by Brinkhoff's famous network-based generator [14].The data were generated based on real-world maps by the network-based spatial-temporal data generating approach.

– The Oldenburg data set contains more than 10,000 trajectories with 6105 nodes and 7035 edges of one day movement over the road-network of the city of Oldenburg.

Table 1. Properties of the Oldenburg data set

Parameter	Value
Map width	23,572
Map height	26,915
Number of trajectories	1k~10k

6.2 Performance Analysis of HDTP

This section analyzes the performance in terms of time and memory cost of HDTP as the number of trajectories grows in the Oldenburg data set. The results are shown in Fig.3 and Fig.4, where the *x*-axis is the number of trajectories and the *y*-axes are the time and the memory cost of HDTP, respectively.

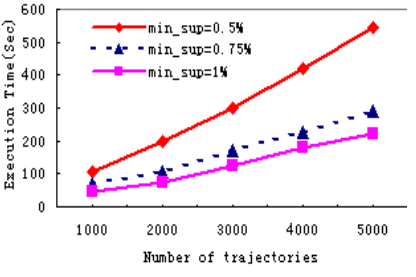


Fig. 3. Execution time of prediction

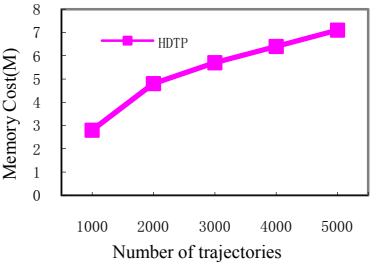


Fig. 4. Memory Cost of prediction

By Fig. 3, we can see that the execution time of the proposed TP algorithm change in an almost linear fashion w.r.t. the trajectories dataset size. Its performances will decrease quickly in the case of smaller minimum support thresholds. By Fig. 4, the memory cost of prediction increases with the number of trajectories, but it does not change drastically. The reason is that the number of new hotspots and extracted trajectory patterns increases fewer as the number of trajectories increase.

6.3 Prediction Accuracy Estimation

This section applies the **accuracy** to evaluate proposed prediction algorithm with the E^3TP over Oldenburg data set. The results are shown in Fig. 5, where the *x*-axis is the number of trajectories and the *y*-axis is the accuracy of prediction.

Fig. 5 shows that the prediction accuracy increases with the number of trajectories. This is because the extracted trajectory patterns are approximate to the real-world case when the number of trajectories grows larger. It helps us to find the best trajectory pattern matching given trajectory. It also shows that the new prediction algorithm outperforms E^3TP with an average gap of 14.7% in accuracy. The reason is that E^3TP obtains the possible speed for moving objects by computing their average speed, but ignores the motion patterns of other objects and spatio-temporal distance, whereas, HDTP does not only considers the frequent motion patterns in the data set, but also takes into account spatial and temporal distances which help improve the accuracy of computing the speed of moving objects.

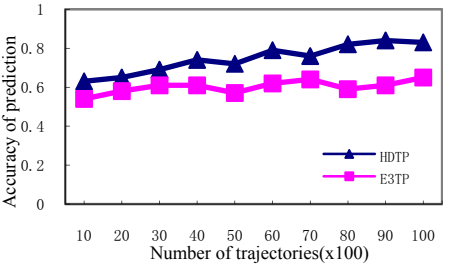


Fig. 5. Accuracy of prediction

7 Conclusions

Existing trajectory prediction algorithms mainly focus on discovering movement behavior of moving objects without constraints and constructing kinematical models to approximate real-world routes, which is far from the real world situations. To overcome these limitations, this study proposes an efficient and effective trajectory prediction algorithm, namely HDTP. It works by: (1) mining the hotspot districts based on the trajectory data; (2) extracting the movement patterns from trajectory data; and (3) predicting the location of moving object using the common movement patterns. By experiments, we compare our trajectory prediction algorithm with the E^3TP method, which show the advantages of our solution.

The further research directions will include that: (1) improve the prediction algorithm for uncertain trajectories [10], and (2) depict the motion behavior based on the random walk as well as predict the location of moving objects.

References

1. Saltenis, S., Jensen, C.S.: Indexing of Moving Objects for Location-Based Service. In: ICDE, pp. 463–472 (2002)
2. Morzy, M.: Mining frequent trajectories of moving objects for location prediction. In: Perner, P. (ed.) MLDM 2007. LNCS (LNAI), vol. 4571, pp. 667–680. Springer, Heidelberg (2007)

3. Trajcevski, G., Wolfson, O., Hinrichs, K., Chamberlain, S.: Managing uncertainty in moving objects databases. ACM Trans. Database Syst. 29(3), 463–507 (2004)
4. Tao, Y., Faloutsos, C., Papadias, D., Liu, B.: Prediction and indexing of moving objects with unknown motion patterns. In: SIGMOD, pp. 611–622 (2004)
5. Giannotti, F., Nanni, M., Pedreschi, D.: Efficient mining of temporally annotated sequences. In: SIAM, pp. 346–357 (2006)
6. Long, T., Qiao, S., Tang, C., Liu, L., Li, T., Wu, J.: E^3TP: A novel trajectory prediction algorithm in moving objects databases. In: Chen, H., Yang, C.C., Chau, M., Li, S.-H. (eds.) PAISI 2009. LNCS, vol. 5477, pp. 76–88. Springer, Heidelberg (2009)
7. Pei, et al.: Prefixspan: Mining sequential patterns by prefix-projected growth. In: ICDE, pp. 215–224 (2001)
8. Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. J. Machine Learning 42(1/2), 31–60 (2001)
9. Morzy, M.: Prediction of moving object location based on frequent trajectories. In: Levi, A., Savaş, E., Yenigün, H., Balcısoy, S., Saygın, Y. (eds.) ISCIS 2006. LNCS, vol. 4263, pp. 583–592. Springer, Heidelberg (2006)
10. Shaojie, Q., Changjie, T., Huidong, J., Teng, L., Shucheng, D., Yungchang, K., Chiu-Lung, C.M.: PutMode: Prediction of Uncertain Trajectories in Moving Objects Databases. J. Applied Intelligence (2009), doi: 10.1007/s10489-009-0173-z
11. Zheng, Y., Zhang, L., Xie, X., Ma, W.: Mining interesting locations and travel sequences from GPS Trajectories. In: WWW, pp. 791–800 (2009)
12. Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D.: Trajectory pattern mining. In: KDD, pp. 330–339 (2007)
13. Monreale, A., Pinelli, F., Trasarti, R., Giannotti, F.: WhereNext: a Location Predictor on Trajectory Pattern Mining. In: KDD, pp. 637–645 (2009)
14. Brinkhoff, T.: A framework for generating network based moving objects. J. Geoinformatica 2(6), 153–180 (2002)

Effective XML Keyword Search through Valid Lowest Information Unit

Ying Lou, Peng Wang, Zhanhuai Li, Qun Chen, and Xia Li

School of Computer, Northwestern Polytechnical University, 710072 Xi'an, China
{louying,wangpeng,lizhh,chenbenben,lixia}@mail.nwpu.edu.cn

Abstract. Keyword search for XML documents has attracted much attention recently. Existing approaches mainly retrieve search results through identifying the Lowest Common Ancestor (LCA) of keyword match nodes. There are also techniques for removing redundant and irrelevant search results. However, the problem of whether a LCA fragment contains complete information has not been adequately investigated. To address this challenge, we first introduce the notion of Information Unit (IU). Its structural property ensures that it can provide complete information on topic to user. We then reason whether a search result is valid by analyzing the context of keyword match nodes. We propose that a meaningful returned result should be a Valid Lowest Information unit (VLIU), which is the lowest IU that contains all the keywords and is semantically valid. Finally, we conduct extensive experiments to demonstrate the effectiveness and efficiency of our approach.

Keywords: XML, keyword search, IU, VLIU.

1 Introduction

As the amount of available XML data grows, the problem of how to retrieve useful information from them emerges. Keyword search has been widely accepted in text document system and World Wide Web. Users do not need to master the complex query language such as XPath or XQuery, or prior know the structure of underlying XML data when they use keyword search.

Compared with web and text retrieval, keyword search in XML data has its distinct characteristics. The structure of XML data contains rich semantic information. These semantics can be exploited to effectively improve keyword search quality. The main idea of existing approaches about XML keyword search is to identify the LCA (Lowest Common Ancestor) of inputted keywords [3][4][5][6]. We use *T(n)* of keywords represent the subtree which is rooted with node *n*. For example, Issuing a keyword query "*XML*, *Jack*" in Figure 1 which intends to find the papers about *XML* written by *Jack*. The indexed node 9 is the LCA of *XML* and *Jack*. The LCA result *T(9)* is relevant to the keyword query. However, the results retrieved by the LCA approach may not always be valid. Consider the following examples.

Example 1.1: *Processing* a keyword query *"XML, search" whose intention is to find the papers about search techniques on XML data. T(10) should be returned as a*

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 85–94, 2010.

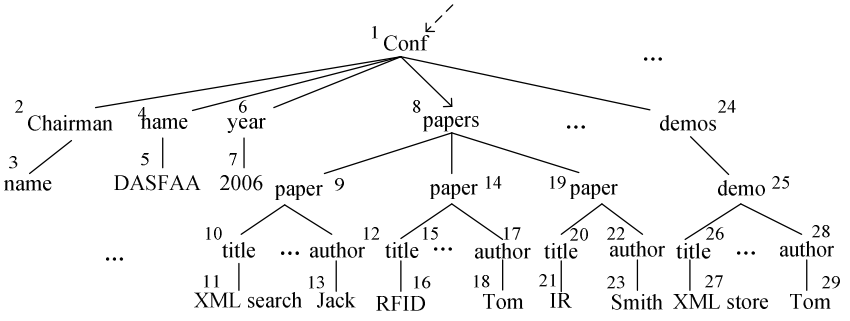


Fig. 1. Sample XML Document

result by LCA approach. It can be observed that T(10) is actually not a good answer. T(10) is only a paper's title, which provides incomplete information to users.

Example 1.2: *Consider "XML, Tom". There are three matching LCA results, which are rooted at conf, papers and demo respectively. Intuitively, T(25) is the most meaningful result among them. T(1) is a result with redundant information. T(8) is an irrelevant result because the keywords XML and Tom appear in different paper entities.*

Example 1.3: *Consider "Jack, Smith". The LCA rooted at papers would be returned. However, the most appropriate answer returned to user should be the T(1) because user may be interested in the conference both Jack and Smith attends.*

From the examples, the existing LCA-based approaches for XML keyword search have the following challenge: 1) *Incompleteness.* A LCA result may be an incomplete information fragment for a keyword query. It only provides partial information of user's topic of interest. 2) *Semantic Irrelevance.* A LCA result may be semantically irrelevant to a keyword query even it contains all the query keywords 3) *Redundancy*. A LCA result may contain redundant information to a keyword query.

In this paper, we propose a novel XML keyword search approach based on the concept of Information Unit to address these issues. Our contributions include:

1. We introduce the concept of Information Unit to represent an XML fragment which provides more reasonable information fragments on the topic of user search intention.
2. We propose that an answer returned to user should be a Valid Lowest Information Unit(VLIU). VLIU not only contains complete information but also is semantically relevant to user's information need.
3. We have conducted an extensive experimental study on real and synthetic datasets to verify the effectiveness and efficient of the VLIU approach. The results demonstrate that VLIU performs better than existing LCA-based approaches on search quality and achieve high efficiency as well.

This paper is organized as follows. Preliminaries and related works are presented in Section 2. Section 3 presents the concept of Information Unit. Section 4 introduces the notion of VLIU and presents the XML keyword search approach based on VLIU. We present the experimental study in Section 5 and conclude this paper in Section 6.

2 Background

In this section, we present preliminaries and review related work on XML keyword search.

2.1 Preliminaries

We model an XML document as a rooted and labeled tree. *Root(T)* denotes the root node of an XML tree *T*. For a node *n* in *T*, *d(n)* represents the label of node *n*. As mentioned in Section 1, we use *T(n)* to denote the subtree rooted at node *n*. *descendant(n)* denotes the set of nodes that are descendants of node *n*. *children(n)* denotes the set of nodes that are children of node *n*. $u \prec v$ ($u \succ v$) denotes that node *u* is an ancestor(descendant) of node *v*. $u \approx v$ denotes that node *u* and *v* do not satisfy ancestor-descendant relationship.

For a keyword *k* and an XML tree *T*, if a keyword *k* is the same as the name of a node *u*, or is contained in the value of the node *u*, we say that *u* is a *match* to *k*. For example, *T(1)* in Example 1.2 is a result for (*XML*, *Tom*). Node 11 and 27 are matches to *XML*. Given an element *e*, $\lambda(e)$ is the set of sub-elements that *e* directly includes.

2.2 Related Work

Most current strategies of XML keyword search are based on improving the approach of finding LCA of keywords. They have proposed their solutions to the problems of LCA that summarized in Section 1. The notion of MLCA (Meaningful LCA) is proposed in [4]. Given a keyword query, if a LCA tree does not contain any subtree that also includes all keyword matches, it is considered to be a meaningful result. SLCA (Smallest LCA) [5] is an extension of MLCA. Given a keyword query *K* and a LCA *l* of *K*, if *descendant(l)* contains no other *LCA* of the query *K*, *l* is a SLCA. The concepts of MLCA and SLCA can remove redundant results from *LCAs*.

The XSEarch[7] and VLCA (Valuable LCA) approaches[6] proposed the criteria for a Valuable LCA. Their definitions are the same: Given any two keyword matches in a LCA, if any pair nodes on the distinct path from the LCA root to each *match* are *homogenous* or *interconnected*, the LCA is valuable.

Different from the above approaches, XSeek [8][10] focus on identifying the information that has to be returned to user. Its search method is the same as the LCA approach. XReal [9] aims to resolves keyword ambiguity problems and proposes an IR-style approach to rank query results. EASE [11] combines IR ranking and structural compactness based on graph structure and rank to fulfill keyword search on heterogeneous data. Both of them rank the search results with the TF*IDF model. The workshop INEX [12] has also organized, which aims to evaluate information retrieval on XML document.

3 Information Unit

The user's information need to XML keyword search is generally fragments of an XML document. What kind of information fragments are fit for keyword query answers?

Firstly, the fragments should cohere with the semantics of keyword query. Secondly, the fragments should have enough and little irrelevant information for information needs. How to extract the structural property of the fragments that are able to answer keyword query more reasonably is our concern.

Definition 3.1 (Information Branch). Given a subtree t in XML data, if $\exists c \in children(root(t))$, $T(c)$ is a *information branch* of t. If $\exists c_1, c_2 \in children(root(t))$ and $d(c_1)=d(c_2)$, c_1 and c_2 are considered to be the *same-type* of information branch.

Semantically, the root label of a subtree represents its information topic and its information branches describe specific information aspects. For example, *T(paper)* in Figure 1 specifies that all information it contains is about a paper, *T(Author)* and *T(title)* are its information branches.

Definition 3.2 (Linear Structure). Given a subtree t in XML data, if $\forall n \in descendant(t)$ (n is not a text node) and n has only one information branch, we say that t has a *Linear Structure*.

We observe that there are usually keywords serving as search conditions in a search result. The information of search conditions is known to user. Suppose a search result has a linear structure and contains all inputted keywords, it means that the result includes only one information aspect, which users have known. The result does not contain unknown information of user's intension. Therefore it should be considered incomplete. For instance, in Example 1.1, both *XML* and *search* are the keywords in the search condition of node *title*. *T(title)* has a linear structure and is an incomplete information fragment for users' information need.

Definition 3.3 (Partial Linear Structure). Given a subtree t in XML tree, if all *information branches* of *root(t)* are of the *same-type*, we say that t has a *Partial Linear Structure.*

A search result that has a partial linear structure contains a group of same-type information branches. If it is returned as a search result, there are two possibilities: The first case is that the keywords belong to different instances of same information topic. The result is irrelevant. For instance, consider the result *T(8)* in Example 1.2, *T(8)* satisfies partial linear structure and contains many *paper* instances. *XML* and *Tom* belong to two different paper entities. It is therefore an irrelevant result. The other possibility is that: Although the keywords locate at different instances of same information topic, the user wants to find relationship of different instance. Unfortunately, the result that satisfies partial linear structure does not contain any common information of different instance. As shown in Example 1.3, user may intend to find the common information about *Jack* and *Mike*(the conference information that *Jack* and *Mike* both attend), but *T(8)* does not contain information for the information need.

According to above discussion, we formally define the notion of information unit which exhibits the structure of a desirable search result.

Definition 3.4 (Information Unit). Given a subtree t in an XML data T, if t does not has a linear or partial linear structure, t is called an *Information Unit*. Give two information unit I_1 and I_2, if $d(root(I_1))=d(root(I_2))$, I_1 and I_2 are *same-type*. $f \in descendant(root(t))$. f is an IU *feature* of t if f is not an IU root.

An IU is the XML information fragment with more reasonable structure on a search topic. Its feature attributes represent the specific aspects of the information it contains. Its feature values describe the contents of these aspects. For example, *T(1)*, *T(2) and T(9)* are all information units in Figure 1. *Name*, *year* and *2006* are the IU features of *T(1)*. *Name* and *year* are feature attributes and *2006* is a feature value.

In this paper, we use information units as the basic fragments for search results. Only information units include all inputted keywords would be returned to user. The Lowest Information Unit(LIU) is defined as follows:

Definition 3.5 (Lowest Information unit). Given a XML tree T and a keyword query $K=\{k_1,k_2,k_3,\ldots k_n\}$, an information unit *t* is a *Lowest Information Unit* for query *K* if :

I. *t* contains all keywords in *K* at least once;
II. There doesn't exist another *information unit t'* containing all the keywords in *K* and satisfying $root(t) \prec root(t')$.

The notion of LIU helps us find compact IU that includes all keywords. However, not all LIUs are desirable results for XML keyword search. The question of what kind of LIU could be returned as final results is discussed in next section.

4 Valid LIU

In this section, we propose some rules to infer whether a LIU is valid. Valid LIUs are supposed to be returned to users.

4.1 Analysis of Existing Algorithms

In an XML data, if the roots of subtrees have the same label, they are supposed to correspond to two distinct instances of an object class. According to the XSearch and VLCA approaches, if the keywords matches in a subtree are the descendants of two or more elements with the same label, this result is considered invalid. Unfortunately, this inference is not always true.

Example 4.1: *Consider a keyword query "Tom, Smith, Conf, DASFAA" that searches for the Conf about DASFAA, which both Tom and Mike haven taken part in. Obviously, T(1) is a desirable result for the query.*

However, according to the XSearch and VLCA approaches, this result is not valid because there are two elements with the same label (paper) in the paths of $Conf \rightarrow Tom$ and $Conf \rightarrow Smith$. Therefore, XSearch and VLCA may return false negative results.

Existing proposals on the XML keyword search do not provide with a universal inference rule to determine whether a search result is valid or not. They mainly focus on the disparity of distinct subtrees with the same label of their roots, which ignore the common information about the similar entities. Information fragments in an XML document are not isolated. They are semantically related. For example, *T(9)* and *T(14)* in Figure 1 also have their association that the two papers are published on the same conference named *DASFAA*. Two bidders may be interested in the same goods of an auction in another scenario.

4.2 VLIU

We infer the validity of a LIU by analyzing the semantic relationship between the keyword matches. It is usually difficult to judge the relevance between keywords matches only the matches themselves. We instead analyze the relationship between keyword matches from their contexts. As IU is a relative integrated information fragment in XML data, the IU that contains the keyword matches can be deemed as the keywords' context.

Consider keyword query "*XML Tom*" again. *T(25)* is a more desirable result. Note that *XML(node 27)* and *Tom(node 29)* locate in the single IU *T(demo)*, it means that they are in the same context and describe a same object. These matches are semantic relevant each other. Therefore, we have:

Rule 4.1. Given a keyword query K and a LIU t of K, if two keyword matches in t locate at a same IU, they are *semantic relevant*.

However, in many cases, the keyword matches of K may locate at different IUs. With considering the relationship of IUs, we present other rules to determine whether two keyword matches are semantic relevant.

The relationships of IUs are determined by the structure between them. If two IUs C_1 and C_2 satisfy $\text{root}(C_1) \prec \text{root}(C_2)$, we say that they satisfy inclusive relationship. Inclusive relationship indicates one fragment is a part of the other. The IUs satisfying inclusive relationship are usually in closely relations. It means the context of keywords in closely relation when keyword matches locate at inclusive IUs. So the keyword matches are *semantic relevant*. For example, consider a keyword query "*DASFAA, 2006, XML*" which searches for the papers about *XML* that were published in *DASFAA 2006*. *T(1)* is a valid result. *DASFAA* and *XML* locate at IU *T(1)* and *T(9)* respectively, which *T(9)* is included by *T(1)*. They are semantic relevant by the inclusive context, we have:

Rule 4.2. Given a keyword query K and a LIU t of K, if two keyword matches in t locate at inclusive IUs, they are *semantic relevant*.

If two IUs don't satisfy inclusive relationship, it means that there is no overlapped information fragment between IUs. In this case, we say that they satisfy separate relationship.

It can be not simple concluded whether two IUs that satisfy separate relationship have semantic relevance. According to the XML semantics, we use connection distance to measure the semantic closeness between two IUs satisfying separate relationship.

Definition 4.1 (Connection Distance). Supposed that two IUs C_1 and C_2 satisfy separate relationship and C is the LIU which includes both C_1 and C_2. The *connection distance* between C_1 *and* C_2 is defined to be the number of IUs (except C, C_1 and C_2,) in the paths of $root(C) \rightarrow root(C_1)$ and $root(C) \rightarrow root(C_2)$.

When the connection distance between two IUs is 0, it indicates that there is another IU that includes both of them directly. These two IUs are thus direct descriptions of another IU. Consider *T(9)* and *T(14)* in Figure 1. There is no other IU in the path from node 1 to node 9 and node 14. We infer that two IUs are probably relevant if their connection distance is 0; otherwise, they are irrelevant. Note that zero is a threshold value of connection distance which can vary according to the semantics of XML documents. In this paper, we set the threshold to be 0.

If two keyword matches belong to two IUs that are irrelevant, the search result that includes them is not relevant to information need. When keyword matches belong to two separate IUs that connection distance is 0, it is more complicated to determine whether they are *semantic relevant*.

Definition 4.2 (Coordinate and Combined). Supposed that two IUs C_1 and C_2 are probably relevant. If C_1 and C_2 are same-type, we say that they are *coordinate*, otherwise, they are *combined*.

We have the observation that two combined IUs describe different aspects of the IU that directly include them. They are similar to the distinct information branches of an IU. For example, *T(2)* and *T(9)* in Figure 1 are combined. They describe different aspects of IU(*T(1)*). Therefore, we have:

Rule 4.3. Given a keyword query K and a LIU t of K, if two keyword *matches* in t locate at two combined IUs in t, they are *semantic relevant*.

In the case that two keyword *matches* are from two coordinate IUs, they describe different entities as shown in the following example.

Example 4.2: *Consider a keyword query "paper, Jack, IR" which searches for the papers about IR written by Jack. T(1) is a LIU. The keyword matches of "Jack" and "IR" are from T(9) and T(19) respectively. T(1) is an invalid result. Note that the IUs T(9) and T(19) are coordinate.*

However, let's review Example 4.1, *T(1)* is a LIU of *"Tom, Smith, Conf, DASFAA"*. *Tom* and *Smith* are also from *T(9)* and *T(14)* respectively. *T(1)* is a meaningful search result to K_2, which user want to the conference that *Tom* and *Smith* have both attended.

Jack and *IR* are the *feature values* of different *feature attributes*, which represent different information aspects. The keyword *Jack* describes the attribute *author* and *IR* describes the attribute *title*. Different from *Jack* and *IR* in Example 4.2, *Tom* and *Smith* in Example 4.1 are the *feature values* of the same *attribute* (author). We have the observation that if users provide with the *feature values* of the same *feature attribute*, they may want to retrieve the common information of the same kind of IUs. Therefore, we have:

Rule 4.4. Given a keyword query K and a LIU t of K, if keyword *matches* in t locate at from two *coordinate* IUs and they are the *feature values* of a same *feature attribute*, they are *semantic relevant*.

According to the above rules, we give the definition of Valid LIU:

Definition 4.3 (Valid LIU). iven an XML tree T and a keyword query $K=\{k_1,k_2,\ldots k_n\}$, t is a LIU of K. For any If there exist *matches* of any two keyword (k_m, $k_n \in K$) in t are *semantic relevant*, we say that t is a *Valid LIU* to K.

Valid LIUs not only guarantees reasonableness of search results from their structure but also excludes those fragments not semantically relevant to user's information need. We design and implement a novel and effective algorithm to computer VLIU of imputed keywords, which is not shown due to space limit.

5 Experimental Study

We compare VLIU with SLCA[5] and CVLCA[6]. All experiments are implemented by Java and run on a PC machine with 2GB RAM and a 2.5GHz CPU. We have tested both synthetic and real datasets. The synthetic dataset is an XMark data [1] with a size of 52MB. Real datasets include WSU(2MB) and DBLP(127MB) downloaded from Washington XML Data Repository [2].

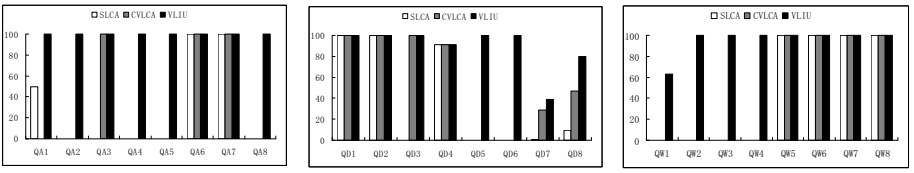
The tested query set consists of three parts with 36 queries in total. We pick eight distinct queries for each data set, part of which are shown in Figure 2.

To measure the search quality, we use the metrics of *precision* and *recall.*

Query		Target	VLIU	CVLCA	SLCA
Auction(50M)					
QA_1	Person20 Person4	open_auction	open_auction	null	open_auction/people
QA_2	Saul Schaap	Person	Person	Name	Name
QA_3	Martti Halgason open_auction4	Person	null	null	people
QA_4	creditcard payment	item	item	creditcard	creditcard
QA_5	Open_auction9 ope_ auction1w atch	Person	Person	null	w atches
Dblp(127M)					
QD_1	Steve Daniel inproceedings	inproceedings	inproceedings	inproceedings	inproceedings
QD_2	Raymond Boyce Author	Article/ inproceedings	Article/ inproceedings	author	author
QD_3	XML search Frank	Article/ inproceedings	null	null	Dblp
QD_4	java book	book	book/inproceedings	book/title	book/tilte
QD_5	World Congress on Formal Methods booktitle	inproceedings	inproceedings	Booktitle	Booktitle
WSU(12M)					
QW_1	320	Course	place/course	Room/crs	Room/crs
QW_2	MICROTECH	Course	Course	Title	Title
QW_3	days th tu	Course	Course	Days	Days
QW_4	Credit 3.0	Course	Course	Credit	Credit
QW_5	AG MACH SYST prefix	Course	Course	Course	Course

Fig. 2. Comparison between Target and Returned Results

Recall & Precision. The comparisons between relevant results and returned results of three approaches are shown in Figure 2. As we can see, VLIU achieves perfect performance in most cases. The only exception is QW_1 *320*, which consists a single value. "*320"* can be a value of course No. or room No. The elements of course and place are both IUs. So the subtrees rooted at *course* and *place* are returned as search results. In the case that the keyword LCA is an IU, the results of three approaches are identical, such as QD_1 and QW_5. In both QA_1 and QA_5, CVLCA returns null. QA_1 intends to find the *open_auctions* that both *Person 20* and *Person 4* are involved in. Unfortunately, CVLCA considers such open_auctions to be not valuable because *Person 20* and *Person 4* are *heterogenous*[6]. Both VLIU and CVLCA return null in QA3 because "*Martti Halgason*" and "*open_auction4*" are from different persons. The intention of QD3 is to search for an article or inproceeding about "*XML search*" written by "*Frank*". VLIU returns null but other approaches return results which are not valid. In these search results, "*Frank*" is an author of an article and "*XML search*" is instead included in the title of another inproceeding. Note that the elements of article and inproceeding are *homogenous*. These results are therefore

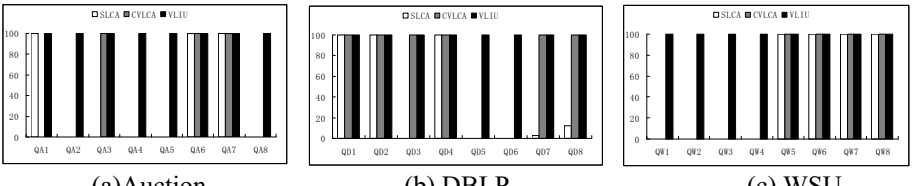


(a)Auction (b) DBLP (c) WSU

Fig. 3. Precision Measurement

invalid. Although CVLCA also returns null in QD_3, the reason is different: the keyword LCA is the root of XML document. The improved performance of VLIU in other queries comes from the fact that VLIU correctly returns all target results but other approaches only return part of them.

As shown in Figure 3 and Figure 4, the evaluation results of recall and precision are consistent with our analysis. When the keywords are text values included in a single element, SLCA and CVLCA often return incomplete results. In these cases, users do not provide with the label of elements they are interested in. For instance, users want to search for a paper but they do not know whether it is labeled with article or inproceeding or something else in XML documents. VLIU instead achieves perfect performance, such as QA_2, QA_4, QD_5, QW_3 and QW_4. In general, SLCA has low recalls in some cases. The structures and data features of DBLP are more complicated than other datasets. So the experimental results on DBLP are various. It is worthy to point out that the precisions of three approaches are all low in the case of QD_7. It intends to search for the papers published in VLDB 2000. There are a lot of irrelevant results that cite the papers published in VLDB 2000. In summary, VLIU achieves better performance than other approaches measured by precision and recall on all tested datasets.

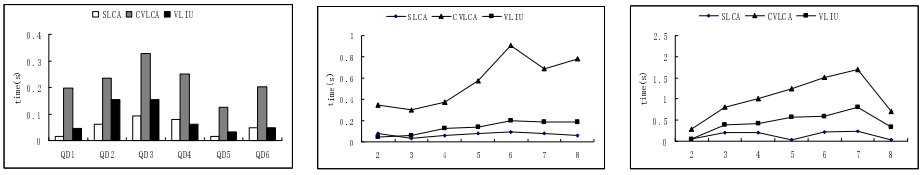


(a)Auction (b) DBLP (c) WSU

Fig. 4. Recall Measurement

We have compared the response time of VLIU with that of other approaches on six DBLP queries. The results are shown in Figure 5 (a). It is observed that SLCA and VLIU are obviously more efficient than CVLCA. The efficiency difference between SLCA and VLIU is instead much smaller.

We have also tested the scalability of VLIU with the query size. The range of keyword number is set to be between 2 and 8. The experiments were conducted on the datasets of Auction and DBLP. The results are shown in Figure 5 (b) and (c). In general, SLCA is more efficient than both VLIU and CVLCA. Since CVLCA and VLIU need to



(a) Response Time on DBLP (b) Processing Time on Auction (c) Processing Time on DBLP

Fig. 5. Response Time and Scalability on Different Number of Keywords

judge semantic relevance of returned LCAs, they consume more processing time. VLIU is considerably more efficient than CVLCA.

6 Conclusion

In this paper, we proposed the notion of IU to ensure the completeness of XML keyword search results. We further presented the rules to determine whether a search result is valid by analyzing the context relationship between keyword matches. Extensive experiments showed that our proposed approach VLIU is more effective than existing techniques and is efficient on both synthetic and real datasets. In the future, we will investigate effective ranking schemes. Valid search results should be output in the order of their relevance.

Acknowledgement. This work is supported by the National Natural Science Foundation of China under Grant No. 60803043 and No. 60720106001, the National High Technology Development 863 Program of China under Grant No. 2009AA1Z134.

References

1. http://www.xml-benchmark.org/
2. http://www.cs.washington.edu/research/xmldatasets/
3. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: Xrank: Ranked keyword search over xml documents. In: SIGMOD, pp. 16–27 (2003)
4. Li, Y., Yu, C., Jagadish, H.V.: Schema-free xquery. In: VLDB, pp. 72–84 (2004)
5. Xu, Y., Papakonstantinou, Y.: Efficient keyword search for smallest lcas in xml databases. In: SIGMOD, pp. 527–538 (2005)
6. Li, G., Feng, J., Wang, J., Zhou, L.: Effective keyword search for valuable LCAs over XML document. In: CIKM, pp. 30–41 (2007)
7. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: Xsearch: A semantic search engine for XML. In: VLDB, pp. 45–56 (2003)
8. Liu, Z., Chen, Y.: Identifying meaningful return information for xml keyword search. In: SIGMOD, pp. 329–340 (2007)
9. Bao, Z., Ling, T., Chen, B., Lu, J.: Effective XML Keyword Search with Relevance Oriented Ranking. In: ICDE, pp. 517–528 (2009)
10. Liu, Z., Chen, Y.: Reasoning and Identifying Relevant Matches for XML Keyword Searc. In: VLDB, pp. 921–932 (2008)
11. Li, G., Ooi, B.C., Feng, J., Wang, J., Zhou, L.: Ease: Efficient and adaptive keyword search on unstructured, semi-structured and structured data. In: SIGMOD, pp. 903–914 (2008)
12. http://inex.is.informatik.uni-duisburg.de/

Reducing Redundancy of XPath Query over Networks by Transmitting XML Views

Yanjun Xu[1], Shicheng Xu[1], Chengbao Peng[1], and Zhang Xia[2]

[1] Neusoft Corporation, Shenyang, China, 110179
[2] Northeastern University Research, Shenyang, China, 110179

Abstract. XPath is a language to retrieve information in the XML documents, which is used to traverse their elements and attributes. At the present time there are a large number of optimization techniques on XML applications at home and abroad, including axis join queries, query rewriting, the semantic cache using in the client, redundancy elimination based on marked suffix tree (MST) and so on. However, these techniques are lack of research to eliminate redundancy in the network environments. In this paper, we reduce communication costs and computing costs over networks by rewriting the queries submitted by the client. Given a set of queries, we compute a minimal XML view that can answer all the original queries and eliminate possible redundancy.

Keywords: XPath; XML; Redundancy Elimination; Query Rewrite.

1 Introduction

XPath[1] is a language to retrieve information in the XML documents, which is used to traverse elements and attributes of theirs. As main elements of the W3C XSLT[2] standard, XQuery[3] and XPointer[4] were both built on the XPath expression. XPath plays an irreplaceable role during querying XML documents. At present there are a large number of optimization techniques on XML applications at home and abroad, including axis join queries[5], query rewriting[6], answering queries by materialized XML views in the middle tier, reducing the frequency of access to the database, the semantic caching using in the client[7] and so on. On the other hand, redundancy elimination has also been researched, including the path shorten and path complementing strategy[8], redundancy elimination based on marked suffix tree(MST)[9]. However, these techniques are concerned about the optimization of the efficiency, and lack of research to eliminate redundancy in the network environments. To resolve this problem, we propose a method of minimizing communication costs and computing costs. This method is rewriting the queries submitted by the client to eliminate redundancy. The existing algorithms are more effective to reduce network traffic, but higher computing costs. Our algorithm is based on the above query rewriting, semantic caching and other optimization techniques, combining the literature on the issue of redundancy elimination. This method is using XML Schema or DTD of the XML document to filter the query set and estimate results, then to decide which query group is chosen according to assessment results.

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 95–105, 2010.

XPath queries can be classified into two categories: simple queries without predicates and complex queries with predicate branches, the former query nodes locate one query path, and there are no predicate branches. XPath query can be described as a query tree, known as the XPath tree pattern. This tree has many nodes, and generally only one of them is the query results, the remaining edges between nodes are constraint of the node. This node is called return node of the query. The main path is the path from the root to the target node. In this path, the number of '/' or '//' means hierarchy number. The nodes with more than one child nodes are so called branch nodes. In addition, if the node defines predicate conditions of element value or element attribute value besides element name, it is called the value predicate node, where the sub-tree which predicate conditions locate called the predicate sub-tree.

In the network environments, the less data transfer makes for the better effect. Therefore, taking XML data size and network traffic into account, it will be better choice that only the necessary information transmits from the server to the client. But the query submitted by users cannot guarantee the results with fewest data-flow. Considering the following scenario: a company intends to send some young employees to be trained abroad for business expansion in the future; it has qualifications requirement in educational background for these trainees. The company may lower requirement for ineligible trainees, but requires the age being smaller. HR specialists can search for the potential trainees in the company's human resource base, and submit the following two queries to get the information needed:

- Querying less than 26-year-old employees to return the basic information of their resumes;
- Querying less than 30-year-old employees met the requirement of "graduate" to return their entire information.

For those potential trainees less than 26 years of age and being graduates, their resumes will be transmitted to the client twice. If the company's human resource base is very large, it will greatly increase the network communication costs. Sometimes, even in one XPath query, there will be redundancy in the results. For example, retrieving the books' sections or sub-sections containing keyword "database", if a section title contains the word "database" and its sub-sections title also contain the word, the sub-section will be returned twice in the query results. That is to say, one of which is returned alone, another is returned included in the parent node, which results in transmission redundancy over networks.

Eliminating these redundancies, we can not only reduce the web traffic, but also increase query efficiency on account of decreasing data transmitted on the premise of that retains certain transmission rate over networks. On how to eliminate these redundancies, in this paper we discuss about the simple XPath query sets and the complex query sets with predicates and propose a algorithm based on query rewriting and redundancy elimination algorithm in the literature[10]. In brief, this algorithm is to calculate the intersection (so-called redundancy) of a set of queries by rewriting XPath operations to remove redundancy and keep only one. The query sets eliminated redundancy in literature[10] generally are more complex than the original sets, so that the server would cost more time to query, and the client would cost time to extract the

results. Although this method reduces communication costs over networks, the client may not endure time-consuming operation. In this article we keep balance between the network traffic and the complexity of query by analyzing the redundancy, if necessary, allow users to decide to give priority to satisfy time or traffic.

2 Redundancy Elimination of XPath Query Sets

The algorithm proposed in this paper rewrites optimization for relevant query submitted by the client, and constructs the minimum XML view set of queries. For some query sets there are two kinds of optimizing selection, one is using the original query, and another is query rewriting sets. In this paper we use the XPath tree pattern to complement the basic algorithm to determine and optimize. When the query needs the context information, we can use additional coding to store context information instead of complex query calculations.

Redundancy elimination processing of Simple Path XPath query sets (only considering the tree pattern to the return node from the root path does not contain a predicate sub-tree of the query tree pattern, and only check the main path) can divided into two types: the redundancy elimination of non-recursive XPath query sets and redundancy elimination of the recursive XPath query sets. A complex XPath query involves a query with a predicate sub-tree.

2.1 Redundancy Elimination of Non-recursive XPath Query Sets

In general, Xpath queries can be expressed as a five-tuple units, $Q:\langle G_t, T_d, SE, PE, R\rangle$, where G_t is the document pattern, T_d is document data, PE is the query path expression, SE is the initial element sets of the query, and R is the result sets. Here to simplify we assume that the document and the pattern is fixed, then the query results can be expressed as:

$$R = Q(PE, T_d) \tag{2-1}$$

Obviously, for $R_1 = Q("/r/a", D)$ and $R_2 = Q("/r/*", D)$, we can see $R_1 \subseteq R_2$ and $R_1 = Q("/a", R_2)$, and R_2 can meet the query requirements. For $R_3 = Q("/r/c/a", D)$ and $R_4 = Q("/r/*/a", D)$, although there is $R_3 \subseteq R_4$, we can not get R_3 for lacking node filter information of R_4. In order to obtain results we can use $R_3' = R_4 - R_3 = Q("/r/\bar{c}/a", D)$,

then we can get $\begin{cases} R_3 = R_3 \\ R_4 = R_3 \bigcup R_3' \end{cases}$.

To summarize the rules mentioned above, for two XPath simple path queries, if one of which is sub-query of the other, that is to say, one path contains the other and the number of hierarchy is the same, which we can only submit the larger query and obtain the sub-query results through the operation in the client. It has been proven that this method can also be suitable to the queries that do not contain intersection among them. This kind of redundancy elimination process can be summed up as follows:

For the query result sets with the same n hierarchies $R_1, R_2, \ldots R_n$, the minimum set of query view after rewriting is:

$$\{V(Q) \mid Q \neq \varnothing, Q \in \{1,2,\ldots,n\}\}$$

where

$$V(Q) = \bigcap_{i \in Q} R_i - \bigcup_{j \in \{1,\ldots,n\}-Q} R_j \tag{2-2}$$

The logic meaning of the formula is: to the result set R for each query Q in the collection, we can draw the smallest query result sets by R intersecting with the rest of several query result sets and adding difference between R and other query sets. So that we can ensure that intersection occur once in the results. From this view set, we can extract answers by executing:

$$R = Q("/*", V(Q)) \tag{2-3}$$

For example, if n = 2, the minimal view is :

$\{R_1 - R_2, R_1 \bigcap R_2, R_2 - R_1\}$, and we can extract answers to R_1 and R_2 by $\begin{cases} R_1 = Q("/*", R_1 - R_2) \bigcup Q("/*", R_1 \bigcap R_2) \\ R_2 = Q("/*", R_2 - R_1) \bigcup Q("/*", R_1 \bigcap R_2) \end{cases}$. As for two different path queries, their intersection is empty, the formula above does not eliminate the redundancy. The query according to the formula occupies the same network traffic as the original query, but increases the complexity of operation in the server, that is why it is important that we filter the query set before submission.

2.2 Redundancy Elimination of Recursive XPath Query Sets

For a single XPath query expression "//a", if there is recursive structure in the querying XML documents, the results will be redundant. Both ancestor node and descendant node of "a" will be returned, and the former includes the latter which will be sent multiple times. An improvement method is to submit a query expression "//a//a//*", it retains the top ancestor of "a" only and removes descendant branches of "a", the result set is $R_5 = Q("//a//a//*", D)$. After receiving the results the client can get all branches: $R = Q("//a", R_5)$. When the query expression "//" is followed by multiple "/", the client should consider how to obtain "/" query node to extract the results. For example, for query expression "//a/b/a/b", according to the previously mentioned methods, eliminating the sub-branch redundancy gets result set as follows:

$R_6 = Q("//a/b/a/b//a/b/a/b//*", D)$, but you can not just query results obtained are as follows in the client: $R = Q("//a/b/a/b", R_6)$, because a sub-tree begins with "//a/b/a/b" in R_6, and its branches include "/a/b" such as "//a/b/a/b/a/b", its second level node "b" also satisfies query requirements, so is "b/a/b" branch, therefore the following two query results are also integral part of the original query: $Q("/b", D)$ and $Q("/b/a/b", D)$.

Redundancy elimination of recursive XPath queries can be summarized as follows:

For the recursive query expressions: $P_e = /PE_1 //PE_2 //\ldots //PE_n$ (starting with "/") or $P_e = //PE_1 //PE_2 //\ldots //PE_n$ (starting with "//"), where $PE_1, \ldots PE_n$ are relative

location paths that do not include "//". Because whether the query starts with "/" or "//" does not matter in the following discussion, here we assume "/". The redundancy answering this query occurs in two ways:

- Elements matching $P_e // PE_n$
- Elements matching $P_e // P$, where P is some suffix of PE_n, referring to the example of $// a / b / a / b$.

For the former kind of redundancy we can submit a view query:

$$Q_e = Q(P_e, D) - Q(P_e / *, D) \tag{2-4}$$

Returned results are as follows:

$$R = Q("/*", Q_e) \tag{2-5}$$

To remove the latter kind of redundancy, we consider a set of relative location paths:

$$S = \{ * / P_n^{(1,k-1)}, * / * / P_n^{(1,k-2)}, ... * / .../ * / P_n^{(1,2)} \}$$

where k is the length of P_n, and $P_n^{(i,j)}$ is the subsequence of P_n from the position i to the position j. Then the algorithm computes the following views:

$$V(T) : (/ PE_1 // .. // (PE_n \bigcap_{p \in T} p - \bigcup_{p \in S-T} p)) - / PE_1 // .. // PE_n // * \tag{2-6}$$

For each $T \subseteq S$, here we use ordinary $\bigcap$ and $-$ because PE_n and each $p \in S$ have the same length k, that is , their answers can not be subelements of other answers. If the result of $PE_n \bigcap (\bigcap p) - \bigcup p$ is empty for some T, $V(T)$ is discarded. Then, we can get the result:

$$\begin{aligned} &Q(/*, V(T)) \\ &Q(// PE_n, V(T)) \\ &Q(/* / PE_n^{(i+1,k)}, V(T)), \text{for each} * / .../ * / PE_n^{(1,i)} \in T \end{aligned} \tag{2-7}$$

3 Redundancy Elimination of Complex Query

If the query sets include predicate branches, we can not use the method mentioned in previous section to eliminate redundancy alone, and we must deal with it by means of other methods, here we use XML schema.

3.1 Recursive Judgement

Because the improved query operations are often more complex than the original and increase the pressure on the server, we use XML schema or DTD to analyze XML structure in the server, and decide whether to rewrite optimization based on different structural features, which would be more effective in improving query efficiency. If

there are no XML schema or DTD in the XML documents, we can extract the XML schema or DTD by the methods mentioned in literatures[11,12,13]. The recursive judgments can be completed during the analysis.

XML Schema is a XML document pattern definition, it includes unrepeated document structure and has less nodes compared with XML document tree. It is more efficient for a group of XML documents with the same pattern using the same schema to scan the whole document trees, comparing to complex queries over the document. Figure 1 is an abstract XML documents Schema tree, to judge whether there is a recursive in the query expressions, we can preorder traversal the Schema tree according to the expression.

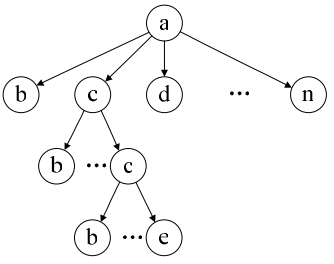


Fig. 1. XML Schema tree structure

3.2 Redundancy Elimination of XPath Query Sets with Predicate Sub-tree

For the query sets with predicate sub-path, we can eliminate redundancy based on XPath tree pattern. The key of to rewrite optimization is to deal with the main path firstly then the predicate sub-tree. If the main path is the same as description in the previous section, we can treat it by the above algorithm. If the main path is completely different, there will be no redundancy. In the following we take different circumstances of predicate sub-tree into account. We shall judge similarity of the main path before rewriting query. For example, the following queries are:

$$\begin{cases} R_5 = Q("a[b/d]/c/e", D) \\ R_6 = Q("a[f/g]/c/h", D) \end{cases}$$

The pattern is shown in figure 2. Dark colored nodes are the nodes returned. Corresponding main paths are $PE_1 = /a/c/e$ and $PE_2 = /a/c/h$, we merge their same parts in accordance with the operation of the tree, and get a new tree pattern structure Q. When we query, only the public branch part of the Q is considered. For example, we only consider "/a/c" path in figure 2.

For the above query, when the primary path is same, that is $PE_1 = PE_2$, then optimization method is to merge its predicate sub-tree, which submit queries $Q("//a/[b/d \mid f/g]/c/e", D)$. The original two predicate paths of "a" will be merged into union $[b/d \mid f/g]$, which avoids the node to meet both $[b/d]$ and

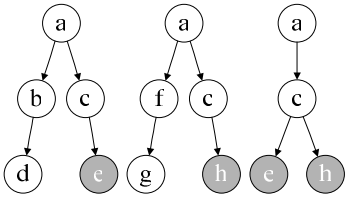


Fig. 2. Predicate sub-tree pattern and the combined pattern

$[f/g]$ returned twice in the results, and decreases the sub-tree of the corresponding node "c" and eliminates redundancy. When query occurs in the server, query results will be returned by scanning the document tree once , and the client only needs the minimal operation, which greatly reduces the query time and traffic. The pattern is shown in figure 3 after the merger.

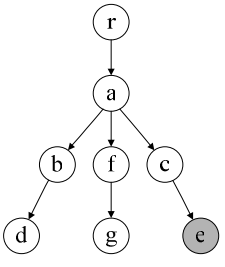


Fig. 3. The merge tree pattern with the predicate sub-tree

These results reduce redundancy, but they are lack of the information of the node "a" being the ancestors of node "e", so we should add additional code on the node "e" as the results returned. Such as code "b" expressing "a" that satisfies $[b/d]$, code "f " expressing "a" that satisfies $[f/g]$, code can be used as an attribute P of the root node. We can do the followings to obtain query results in the client: According to the query root node attribute P encoded as "b" or "f " to distinguish which branch should be chosen, the two predicate branches are all under the node "a". Different documents have the different redundancy, our optimization program is: firstly the client submits the original query sets and the rewriting query sets, then the server determines whether sub-tree "b/d" and "f/g" is under the node "a" according to Schema or DTD before queries, which is shown in figure 4.

In figure 3, two predicate sub-trees are under the same node "a", then the two query nodes "e" satisfying the condition is the same node, and the result is redundancy, this time we use new query sets. In figure 4, the node "e" returned is in two different sub-trees, when the results of two queries will not be duplicated, we can carry out the original query sets, and the computation is reduced both in the server and in the client. Document Schema tree use the prefix code, for node "b" and "f ", if

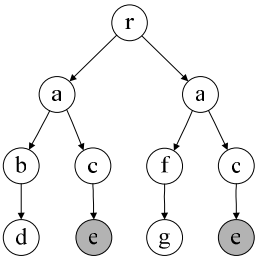


Fig. 4. The tree pattern with the information of the root

the code prefix is the same as "a" and the two nodes code position are the same, we can determine whether the nodes are brother node and sub-tree "b/d" and "f/g" are under the same node "a".

4 Balance of Computing and Size

As we know, the algorithm in literature[10] is very complex and high computing costs. We should construct automation by this algorithm if the query sets are non-recursive; otherwise, we use formulation. Sometimes there are no redundancies in the query sets, but we still have to waste resource to compute using this algorithm. We need not eliminate redundancy when the query sets are few and the size of the documents queried is not enough large(for example, not larger than 1 MB) or the size of redundancy is very small even though there are some redundancies in the query sets.

We can easily measure the size of documents, here we discuss how to judge whether there are redundancies in the query sets or not. Generally speaking, there are redundancies only if the results of the query sets exist intersection. Certainly, it is not worth a straw that we got it after we have executed the queries, we should get to know it before querying. We start from the query to determine the existence of redundancy, find a method both eliminating redundancy and needing less computation time. Although it is unrealistic that we completely eliminate redundancies without consuming computing time, a balanced approach of judging most redundancies and costing less time need to be adopted. That is to proceed from the query path. Currently, we are studying a statistical method by self-learning model to balance, but see any effect at initial step. So it is not the discussible part here.

We use some simplified methods to judge redundancy and do some efforts to reduce computing time greatly. Those methods will process small parts of query with no redundancy as the query containing redundancy. Redundancy is judged by the following steps：

- If the query paths are included, the results must be redundant. It is obvious that we do not discuss here.
- Short path is first. Comparing the paths we start from the shortest one, and if this path is the beginning of the other arbitrary query, we can say that there is

redundancy. For example, for $PE_1 = "/r/a"$ and $PE_2 = "/r/a/b"$, having $R_1 \supset R_2$, where $R_1 = Q(PE_1, D)$, $R_2 = Q(PE_2, D)$.

- The query containing "//" and "*" is processed as redundancy. This treatment is not perfect, in other word, small parts of query with no redundancy will be treated as the query containing redundancy. Such queries are difficult to judge redundancy by a simple method, which may or may not exist. In order not to spend time verifing the redundancy, here we directly deal with it by previous algorithm. One problem here is that there is no redundancy in query sets but we still have to spend much time on calculation by algorithm, maybe we can solve it by the statistical methods that we are studying.

In other circumstances, we inquire directly without processing redundancy.

5 Testing and Analysis

In this chapter we test and analyze the algorithm of eliminating XPath redundancy.

5.1 Testing Environment

The testing environment shown in table 1:

Table 1. Testing environment

	Server	Client
CPU	Intel XEON 2.0G*4	Intel Core2 6300
RAM	4GB	2GB
HDD	RAID0（72G*4 + 36G*2）	160G SATA
OS	Windows Server 2003 Red Hat Enterprise Linux 4	Ubuntu 9.10

5.2 Test Results and Analysis

5.2.1 Test of Traffic and Time

Using xmark[14] benchmark data generator produces about 112M data sets. The test was to demonstrate that search results of query set after redundancy elimination occupy less network traffic than the original query sets. Listed in the following is the original query sets:

$$\begin{cases} PE_1 = /site/regions/namerica/item \\ PE_2 = /site/regions/europe/item \\ PE_3 = /site/regions/*/item/name \\ PE_4 = /site/regions/*/item/description \end{cases}$$

The improved query sets using this algorithm is as follows:

$$\begin{cases} PE_1 = /site/regions/namerica/item \\ PE_2 = /site/regions/europe/item \\ PE_3' = /site/regions/\overline{A}/item/name \\ PE_4' = /site/regions/\overline{A}/item/description \end{cases}$$

where $A = \{namerica, europe\}$, $\overline{A}$ is difference of $\{namerica, europe\}$, queries are:

$$\begin{cases} Q_1 = Q(PE_1, D) \\ Q_2 = Q(PE_2, D) \\ Q_3 = Q(PE_3', D) \\ Q_4 = Q(PE_4', D) \end{cases}$$, the original results are : $$\begin{cases} R_1 = Q_1 \\ R_2 = Q_2 \\ R_3 = Q_1/name \cup Q_2/name \cup Q_3 \\ R_4 = Q_1/description \cup Q_2/description \cup Q_4 \end{cases}$$

We monitor network traffic and query time while querying, and the results are in table 2. In this test, the original query group query data is about 70.3MB, after optimized is just 48.3M, which saves 22.0MB and is 31.3% of the original data; the original query time is 1,988.6 ms, after optimized is 1,633.7 ms, which saves 359.4 ms, is 17.8% faster than the original query; the time of getting and printing in the client is 219,653.3 ms and 144,192.7 ms, which saves 75,460.6 ms, is 34.4% faster than the original query.

Table 2. Comparison of the query optimization

	Network Traffic (KB)	Query Time (msec)	Print Time (msec)
The Original Query Sets	72,072	1,988.6	219,653.3
The Query Sets Optimized	49,544	1,633.7	144,192.7

These results suggest that redundancy elimination algorithm reduces network traffic and improve the efficiency of the query.

5.2.2 Comparison of Computing Time with Original Algorithm

We chose a group of queries that contain no redundancy to compare computing time with the algorithm in literature[10], and the results are in table 3

Table 3. Comparison of Computing Time

	The Algorithm in Literature[10]	The Algorithm in this Paper
Computing Time	340 msec	18 msec

6 Conclusion

In this paper, we propose a new XPath query set redundancy elimination algorithm based on query rewriting. In this algorithm we evaluate redundancy in different XML document structures using XPath tree pattern. The experiments prove that this algorithm has increased the efficiency of the query both in a simple XPath query sets and in complex queries with predicates.

References

1. XML Path Language (XPath) 2.0 [DB/OL], http://www.w3.org/TR/xpath20/
2. XSL Transformations (XSLT) Version 2.0 [DB/OL], http://www.w3.org/TR/xslt20/
3. W3C XML Query (XQuery) [DB/OL], http://www.w3.org/XML/Query/
4. W3C XML Pointer, XML Base and XML Linking [DB/OL], http://www.w3.org/XML/Linking/
5. Wang, Z., Geng, R., Wang, G.-r.: Study on Processing Techniques for XPath Axis Join Queries. Mini-micro Systems 11, 72–77 (2005)
6. Gao, J., Yang, D., Wang, T.: Efficient XML query rewriting over the multiple XML views. In: Proceedings of the 17th Data Engineering Workshop (DEWS 2006) (2006)
7. Ta, N., Feng, J.-h., Li, G.-l.: Survey on semantic cache in native XML databases. Journal of Computer Applications 26(12), 2977–2981 (2006)
8. Lü, J.H., Wang, G.R., Yu, G.: Optimizing path expression queries of XML data. Journal of Software 14(9), 1615–1620 (2003)
9. Han, D.-h., Wang, G.-r., Qiao, B.-y.: Optimizing Common Sub-Queries in XML Data for Regular Path Expressions. Journal of Northeastern University (Natural Science) 26(6), 535–537 (2005)
10. Tajima, K., Fukui, Y.: Answering XPath queries over networks by sending minimal views. In: Proceedings of the 30th International Conference on Very Large Databases, Toronto, Canada, pp. 48–59 (2004)
11. Maarouf, M.Y., Chung, S.M.: XML Integrated Environment for Service- Oriented Data Management. In: Proceedings of the 2008 20th IEEE International Conference on Tools with Artificial Intelligence, vol. 2, pp. 361–368 (2008)
12. Ramanan, P.: Worst-case optimal algorithm for XPath evaluation over XML streams. Journal of Computer and System Sciences 75, 465–485 (2009)
13. Nasser, M., Ibrahim, H., Mamat, A., Sulaiman, M.N.: Generating Free Redundancy XML Documents from Non Normalized Relational Views Using A Statistically Approach. In: Proceedings of 3rd International Conference on Information and Communication Technologies: From Theory to Applications, ICTTA 2008, pp. 1–7 (2008)
14. XMark — An XML Benchmark Project [DB/OL], http://www.xml-benchmark.org/

Building Web Application with XQuery

Zhiming Gui, Husheng Liao, and linlin Fu

College of Computer Science, Beijing University of Technology, Beijing, China
{zmgui,liaohs}@bjut.edu.cn

Abstract. This paper proposes to design and implement a new XQuery-based framework for generating web application. It supports the development of both client and server side program in a uniform way using XQuery. Further more, through translating the fully XQuery based web application into corresponding client and server side code in appropriate target language, it enables web application developed using our framework to run as normal J2EE application. An online book-shop application is given as an example to illustrate how to build a web application using our framework. It shows how our framework can simplify web application development and improves flexibility.

Keywords: XQuery, web programming, HTML.

1 Introduction

The state of the art of web programming is generally to use mixed languages such as Perl, PHP or JavaScript. These web languages aim at supporting primarily the browser or the server. Further more, because the web page is described using HTML/XHTML, which is declarative, mixing the HTML/XHTML and these imperative code makes the web development tedious and hard to manage. The paper proposes a new framework for generating web application based on XQuery. The core of our framework includes our extended XQuery language which enables build web application in fully XQuery, and a compiler which is used to translate our extended XQuery language into appropriate target language for the client and the server. The remainder of this paper is organized as follows: Section 2 describes our framework. Section 3 sketches our implementation. Section 4 gives an on-line book shop application as the example to illustrate how to build a web application with our framework. Section 5 introduces the related works. Section 6 contains the discussion and conclusions.

2 Architecture

As shown in Figure 1, our framework includes a general purpose web language that enables coding both client and sever side program in extended XQuery. We also developed a prototype compiler to translate it into appropriate target languages that can run on normal J2EE environment. The advantages of this framework include: It simplifies the web development by applying the same programming language to both the server and client side; It enables programming both client and server side declaratively,

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 106–109, 2010.

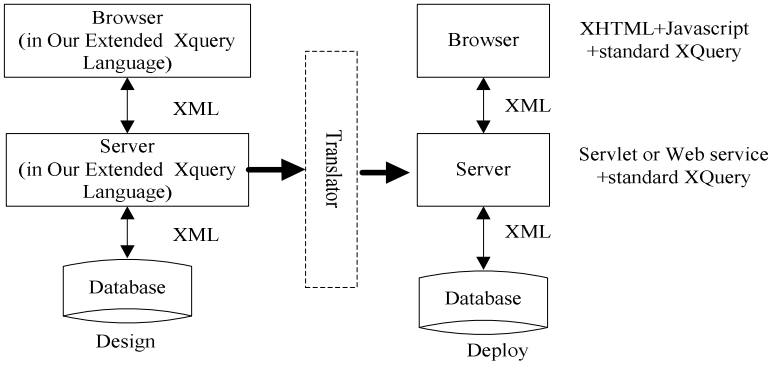


Fig. 1. the Framework

which makes generation and manipulation of web pages easier; It enables the web application developed with our framework to run as normal Java web application.

3 Design and Implementation

We extend XQuery with a set of powerful functionalities for both browser programming and server side programming. For the client side, we add extensions for DOM navigation, invoking the method of DOM objects, DOM updating, browser event model, loading remote XML data source and web service call.

For the server side, our extensions include: the function of designating an XQuery function as the Servlet or web service operator provider; the "forward" mechanism like Servlet does; the corresponding object for Application and Session that is used as internal object in JSP.

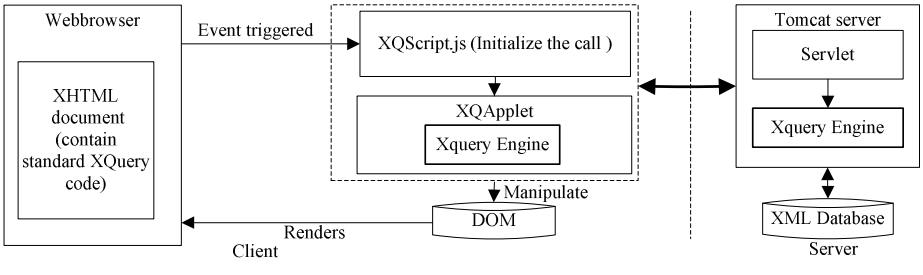


Fig. 2. Running process of our web application

Both the server side and the client side web page in our language have the same structure as an XQuery program that consists of prolog and query body. Prolog includes the declarations of variables and functions. There are two kinds of functions--updating functions and non-updating functions. The updating functions are used to respond the webpage events and re-render the web page, while the non-updating functions are focus on webpage construction or be invoked by other functions. The

Query Body is response for constructing the main body of XHTML document. A real example can be found in section 4.

We provide a compiler to manually translate web application in our language into appropriate normal client side and server side code. For the browser, it compiles it into HTML page with javascript and standard XQuery code. The translated client side is browser independent and can run in all browsers that support Javascript without any pre-installed plug-in. However, to make the translated XHTML page runs in the browser, a series of JavaScript functions and an Applet program will be automatically included in the translated XHTML page during the compile phase. The applet embeds a standard XQuery engine. The Java scripts are used to invoke the applet to evaluate the XQuery expression. For the server side, it compiles it into a Servlet. A standard XQuery engine is also needed at the server side. It should be preinstalled in the web server library. After the compiling, the whole translated web application can be deployed as a normal Java web application. The running process can be described by Figure 2.

4 Example

For the development of the on-line book shop application, we choose typical three-tier application architecture. The server side program only receives queries and returns the search result (not the result web page) to the client. The client interact with the user and make update to the web page upon the return of the result. Here we only list part of the source code.

The Code-1 below shows an example of adding books to the Cart, which is stored as a XML document named item in the session. It exemplifies the main extension of XQuery for the server side.

```
declare  function xcgi:addbooks($item as xs:string*)
{     for $i in $item
          return
            insert nodes <item>{$i} </item> into
            doc("xcgisession:items")//items,
            xcgi:forward("/showchoice")};
```

Code 1: Server Side Module of Adding Books to the Cart

The Code 2 below shows an example of sorting the book list by its attribute. The sort computing and the generating of new web page functions are both described declaratively and run at the browser side.

```
declare function local:Sort($key){
   let $table:=xqp:dom()//table[@id='t']
      return <table id='t' cellspacing='2px'>
        {  for $tr in $table//tr[td//table] order by
           $tr//table//tr[td][$key]  return   lo-
cal:showbook($books//book[@id=$tr/hidden[1]/@value])
}
      </table>};
```

Code 2: Part of Client Side Module for Sorting

5 Related Work

The XQIB[1] proposed an extension of XQuery for the browser programming and built a plug-in for running XQuery in IE. It provides a more comprehensive extension for the browser programming than us. However, it put more emphasis on the browser side. The DEWAX[3] is more concentrate on using XQuery to building a thin-client, jsp-like web application. Our work is more similar with Google Web Toolkit and Kawa. GWT [4] allows cross-compilation from JAVA to Javascript so as to enable programming the web application using purely JAVA. However, compared to a JAVA solution, XQuery is declarative and naturally supports manipulate the XHTML. Qexo [5] can compile an XQuery program to a Servlet using the Kawa framework. Both Qexo and DEWAX are put more emphasis on the server side. Compare to these works, our work provides a more general purpose web programming language, both for the browser and the server. Another difference as compared to these works is that our compiler can translate our web application into normal web application which can run under normal J2EE environment with a standard XQuery engine as well as need not to preinstall any plug-in or rely on specialized XQuery application server.

6 Conclusions

The paper explores a new XQuery-based framework for developing web application. It provides a general purpose web programming language which can cover the whole range of browser and server functionalities. Through avoiding the technology jungle of mixing different technologies and coding both the client and server side declaratively, our framework can simplify the web development and provides a more flexible and scalable architecture.

References

1. Fourny, G., Pilman, M., Florescu, D.: XQuery in Browser. In: Proceedings of WWW 2009, Madrid, Spain, April 20-24 (2009)
2. Chamberlin, D., Carey, M., Florescu, D., Kossmann, D., Robie, J.: XQueryP: Programming with XQuery. In: XIME-P 2006, Chicago, IL, USA (June 2006)
3. Kaufmann, M., Kossmann, D.: Developing an Enterprise Web Application in XQuery. In: ICWE 2009, pp. 465–468 (2009)
4. Google Web Toolkit, `http://code.google.com/intl/en/webtoolkit/`
5. Qexo - The GNU Kawa implementation of XQuery, `http://www.gnu.org/software/qexo/`

Functional Dependencies for XML*

Haitao Chen, Husheng Liao, and Zengqi Gao

College of Computer Sciences, Beijing University of Technology, Beijing, China
chenheyuzhi@yahoo.com.cn, liaohs@bjut.edu.cn, zengqigao@gmail.com

Abstract. In this paper, we present a new approach for defining functional dependencies for XML (XFDs) on XML Schema. While showing how to extend XML Schema, we analyze the expressive power of our XFDs. We focus on supporting complex value (e.g. list, set) in our proposal. A novel concept *match tree* is introduced for judging the value equality of complex type elements. The satisfaction of XFDs in an XML document is defined in terms of the value equality. We also discuss the advantages of our XFDs over other previous work.

1 Introduction

How to define functional dependencies for XML (XFDs) is an important topic. Although XFDs have been investigated more intensively than other data semantic constraints for XML [1], there seems to be no consensus on how to define such dependencies. XML Schema, as a standard language for defining the schema of XML data, provides the mechanism for specifying the identity constraints. However, XFDs can not be defined in XML Schema. With the development of XML and related technologies, it is indispensable to unify the previous definitions and extend XML Schema with the ability to express XFDs.

It is not a trivial task to extend functional dependency from relational database to XML because of the structural difference. Another important problem is how to define *value equality* for complex elements more reasonably.

In this paper, we present a new approach for defining XFDs on XML Schema. The users can use our XFDs to provide more semantic information and maintain the integrity of XML data. In summary, the main contributions are:

1. We extend XML Schema with the ability to specify XFDs, which has been implemented on XSV (the official reference implementation for the XML Schema language).
2. We extend the *field* component in XML Schema to support complex value (e.g. set, list).
3. A novel concept *match tree* is introduced for judging the value equality of complex type elements.

* This research is supported by Beijing Municipal Natural Science Foundation (4082003).

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 110–115, 2010.

This paper is organized as follows. Section 2 presents an example that illustrates our approach. Section 3 formally defines XFDs on XML Schema and the satisfaction in an XML document. Section 4 gives an overview of related work and discusses the advantages of our proposal. Section 5 concludes the paper and states some future work.

2 An Example

We consider the following example. The university includes a set of courses and a set of colleges. Teachers who teach the course are recorded in the corresponding course. The teachers under course have class information that contains the students of the class. The teachers under college do not record the class information. The XML Schema is shown in Fig. 1. We do not give an XML document instance for saving space.

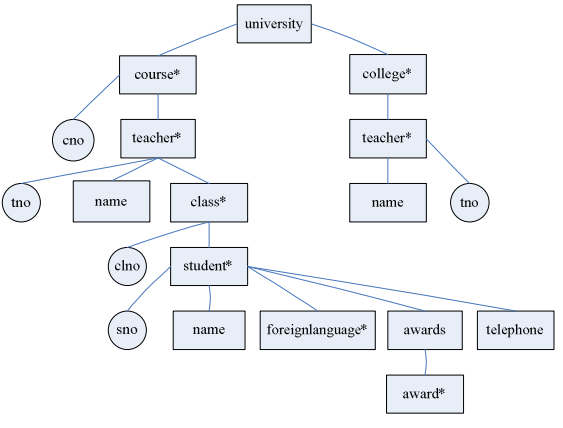


Fig. 1. the definition of XML Schema

We consider the following constraints: (A) Course number and teacher number determine the set of classes, (B) Teacher number determines teacher name in the entire tree, (C) Student number determines the list of foreign languages, awards and telephone.

In general, a functional dependency has the form $X \rightarrow Y$ where $X = \{x_1 \dots x_n\}, Y = \{y_1 \dots y_n\}$. We call X the **determinant** and Y the **dependency**. All involved information items participating in a functional dependency are called **participant**. For defining XFDs on XML Schema, we add a new component *functional*, which can be declared in an element. We use a new component *participant* to express **participant**, which occurs in a *functional* component. In a *participant* component we use existing *selector* component to select the elements to which the constraint applies. D*eterminant* and *dependency* components are introduced to represent the above corresponding concepts, which

include a set of *field* components respectively and appear in the *participant* component. Notably, we employ the existing *field* syntax, but with an extension to its semantics, which will be discussed in the following section. The *participant* component can contain another *participant* component to show the rest of participants that should have a new context element specified by a new *selector* component, which caters the hierarchical structure of XML.

We give three fragments of XML Schema to illustrate how to specify the above three constraints in our approach, which is shown in Fig. 2.

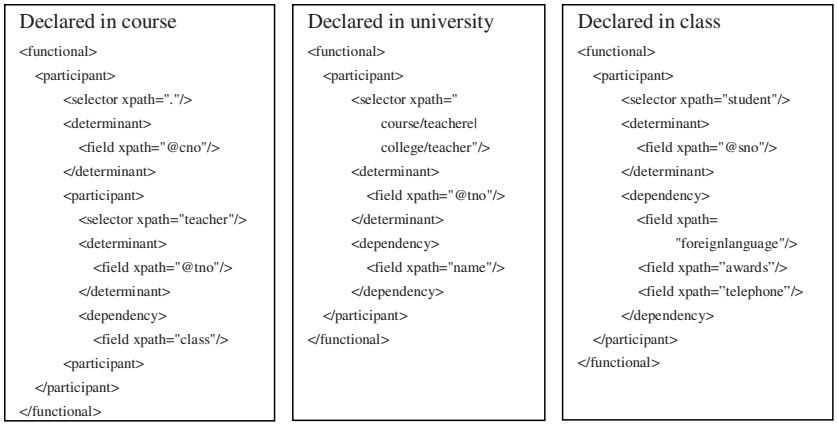


Fig. 2. XFDs on XML Schema: (a) XFDs for example constraint (A), (b) XFDs for example constraint (B), (c) XFDs for example constraint (C)

3 XFDs on XML Schema

In this section, we introduce our XFDs and related concepts. For saving space, some basic concepts, such as XML schema and paths, are omitted, which can be found from the early literature [1,2,3].

3.1 XFDs

We add *functional* component to XML Schema to define XFDs, which is shown in Fig. 3. The above definition can be abstracted as $p_1[q_{11}q_{1i} \rightarrow r_{11}r_{1j}](p_2[q_{21}q_{2k} \rightarrow r_{21}r_{2l}](p_3[q_{31}q_{3m} \rightarrow r_{31}r_{3t}](\ldots p_n[q_{n1}q_{nu} \rightarrow r_{n1}r_{nv}])))$ where (1) i, j, k, l, m, t, u, v are nonnegative integer; (2) if $i \in [2, n]$, p_i is the schema path that is relative to p_{i-1}, p_1 is the *schema path* that is relative to the element in which the XFDs are defined (p_i can be regarded as an abstraction of *selector* component); (3) q_{ij} and r_{ik} are two *schema paths* that is relative to p_i. q_{ij} represents one of *determinant* information items, and r_{ik} represents one of *dependency* information items where i is the level number; (4) the nodes identified by the upper level *schema path* are the context of the nodes identified by the current level *schema path*.

<functional Id=ID Name=NCName /> Content: participant </functional>
<participant> Content: selector, determinant? , dependency? , participant? </participant>
<determinant> Content: field+ </determinant> <dependency> Content: field+ <dependency>
<field Id=ID xpath= a subset of XPath expression type=set or list/>

Fig. 3. the definition of functional component on XML Schema

Notably, we extend the existing component *field* since it is not reasonable to limit the node identified by *field* is a single node and whose value is of simple type. In XML document, a information item maybe a set of nodes or a list of nodes if order is important. And the value of the information item can be a complex value if the corresponding node is a complex type element node. By this extension, we can efficiently support the functional dependencies (A), (C) in Sec. 3.

3.2 Value Equality

Match Tree. It is crucial to compare two complex type elements when we define and check the satisfaction of XFDs. Crux of the matter is Occurs components make elements repeat, but it is unordered at semantic level. We propose a novel concept *match tree* to deal with it.

We express a complex type as a *particle*, which has three properties: *min occurs*, *max occurs* and *term*. The term is either an element declaration, a wildcard, a *Sequence* or a *Choice*. A *Sequence* has a sequence of *particles* and a *Choice* has a set of *particles*. Given a complex type T and an element e, which has type T, a *match tree* of e is a tree that can be constructed by following steps: (A) If the *term* of the current *particle* is an *element declaration* or a *wildcard*, and can be matched by one element, we return the element as the leaf node of the *match tree* below the current position. (B) If the *term* of the current *particle* is a *Choice*, we construct an *unordered node* below current position and set this *unordered node* as the current position, then we match the *particles* in the set of *Choice* in terms of the *occurs* of the current *particle*. (C) If the *term* of the current *particle* is a *Sequence*, we match the sequence of particles in the *Sequence* and organize them under an ordered node. If the *Sequence particle* occurs more than once, a new unordered node is inserted at the current position and the ordered nodes are organized under this unordered node.

Informally, for a valid complex type element, a *match tree* records how an element instance matches its type definition. It is easy to find that for a complex type definition, if two valid element instances, v_1 and v_2, are value equal, the match tree of v_1 can be matched by v_2 using a top-down match algorithm. Since the match tree can capture the ordered and unordered information between elements, it is more reasonable to compare element with complex type by this concept.

Definition of Value Equality. Let v_1 and v_2 be nodes in D. v_1 and v_2 are called value equal, denoted $v_1 =_v v_2$, if $lab(v_1) = lab(v_2)$, and (A) if v_1 and v_2 are both simple type elements or attributes, $val(v_1) = val(v_2)$; (B) if v_1 and v_2 are complex type elements, the match tree of v_1 can be matched by v_2 .

Let l_1 and l_2 be list of nodes. l_1 and l_2 are called value equal, denoted $l_1 =_v l_2$, if the members of two lists are pairwise value equal.

Let s_1 and s_2 be set of nodes. s_1 and s_2 are called value equal, denoted $s_1 =_v s_2$, if for every member m_1 of s_1, there is a member m_2 of s_2 such that $m_1 =_v m_2$ and vice versa.

3.3 The Satisfaction of XFDs in an XML Document

For an XML document and the XFDs in our approach, $p_1[q_{11}q_{1i} \rightarrow r_{11}r_{1j}](p_2[q_{21}q_{2k} \rightarrow r_{21}r_{2l}](p_3[q_{31}q_{3m} \rightarrow r_{31}r_{3t}](\ldots p_n[q_{n1}q_{nu} \rightarrow r_{n1}r_{nv}])))$ where i, j, k, l, m, t, u, v are nonnegative integer, the values of outer level are the context of the values of the inner level. By unnesting, we can transform the result into a table that is similar to relational databases. Thus, we define the satisfaction of XFDs in an XML document by using tuples in the table, which is easy to describe and understand. The similar idea for generating tuples can be found from [2]. We employ the concept of *value equality* discussed before.

For an XML document D and a specified XFDs fd, we assume we have got the table $T(Q_1, \ldots, Q_n, R_1, \ldots, R_m)$. We say D satisfies fd if $\forall t_i, t_j \in T$ where $t_i = \{q_{i1}, \ldots, q_{in}, r_{i1}, \ldots, r_{im}\}$, $\{q_{j1}, \ldots, q_{jn}, r_{j1}, \ldots, r_{jm}\}$, (1) $\exists k \in [1, n]$, q_{ik} or q_{jk} is null value or $q_{ik} \neq_v q_{jk}$; or (2) $\forall k \in [1, n], r_{ik} =_v r_{jk}$.

4 Related Work

XFDs have been studied relatively intensively. Most of them are path-based [3,4,5,6,7,8,9]. Reference [3] proposes the concept of tree tuple, by which the concept of functional dependency in relational database is extended to the XML setting. XFDs are defined on DTD and normalization of XML is studied. Reference [4] extends the definition of functional dependencies in incomplete relations to XML, which has the similar expressive power to [3]. The major difference between them is the treatment of null values. Reference [5] proposes a schema language-independent representation for XFDs. A *header* is used to specify XFDs, which is a path and defines the scope. Local XFDs discussed in [6] to capture the constraints that hold in part of an XML, which is similar to [5]. XFDs in [7] are defined by paths, which differentiate between global XFDs and local XFDs, and support not only string values but also elements. XFDs in [8] and [9] have the similar expressive power to XFDs in [7]. And they also focus on key constraints. Unlike the path-based methods, reference [10] and [11] are tree-based, which can capture further kinds of XFDs.

Comparing with previous methods, our approach has the following advantages: (A) Our approach defines XFDs on XML Schema which is a standard schema language. (B) The information items participating in an XFDs can be set or list, and the single node can be of complex type. (C) By the concept of match tree, we can define the value equality more reasonably.

5 Conclusions and Future Work

XFDs is fundamental to other related areas of XML theories and applications. It is very important to have consensus on how to specify it. This paper proposes a new approach on defining XFDs. Our proposal, which has been implemented on XSV, is path-based. We extend the *field* component to support complex value (e.g. set, list) and complex type element. A novel concept *match tree* is introduced for judging the value equality of complex type elements more reasonably. Our proposal can make data semantic richer and assist to maintain data integrity.

Currently, we are extending XML Schema with mechanism for specifying other constraints. Future work includes reducing restrictions on XPath and more efficient algorithms for checking the constraints. Normalization theory for XML and the reasoning with our XFDs are also worthy of being studied further.

References

1. Wang, J.: A comparative study of functional dependencies for XML. In: Zhang, Y., Tanaka, K., Yu, J.X., Wang, S., Li, M. (eds.) APWEB 2005. LNCS, vol. 3399, pp. 308–319. Springer, Heidelberg (2005)
2. Vincent, M., Liu, J.: Checking functional dependency satisfaction in XML. In: Bressan, S., Ceri, S., Hunt, E., Ives, Z.G., Bellahsène, Z., Rys, M., Unland, R. (eds.) XSym 2005. LNCS, vol. 3671, pp. 4–17. Springer, Heidelberg (2005)
3. Arenas, M., Libkin, L.: A normal form for XML documents. ACM Transactions on Database Systems 29(1), 195–232 (2004)
4. Vincent, M., Liu, J., Liu, C.: Strong functional dependencies and their application to normal forms in XML. ACM Transactions on Database Systems 29(3), 445–462 (2004)
5. Lee, M., Ling, T., Low, W.: Designing functional dependencies for XML. In: Jensen, C.S., Jeffery, K., Pokorný, J., Šaltenis, S., Bertino, E., Böhm, K., Jarke, M. (eds.) EDBT 2002. LNCS, vol. 2287, pp. 124–141. Springer, Heidelberg (2002)
6. Liu, J., Vincent, M., Liu, C.: Local XML functional dependencies. In: WIDM 2003: Proceedings of the 5th ACM International Workshop on Web Information and Data Management, pp. 23–28. ACM, New York (2003)
7. Yan, P., Lv, T.: Functional dependencies in XML documents. In: Shen, H.T., Li, J., Li, M., Ni, J., Wang, W. (eds.) APWeb Workshops 2006. LNCS, vol. 3842, pp. 29–37. Springer, Heidelberg (2006)
8. Ahmad, K., Ibrahim, H.: Functional Dependencies and Inference Rules XML. In: Internaltional Symposium of Information Technology, pp. 494–499 (2008)
9. Shahriar, M.S., Liu, J.: On defining functional dependency for XML. In: International Conference on Semantic Computing, pp. 595–600 (2009)
10. Hartmann, S., Link, S.: More functional dependencies for XML. In: Kalinichenko, L.A., Manthey, R., Thalheim, B., Wloka, U. (eds.) ADBIS 2003. LNCS, vol. 2798, pp. 355–369. Springer, Heidelberg (2003)
11. Lv, T., Yan, P.: XML constraint-tree-based functional dependencies. In: Proceedings of ICEBE, pp. 224–228 (2006)

Structure and Content Similarity for Clustering XML Documents

Lijun Zhang, Zhanhuai Li, Qun Chen, and Ning Li

School of Computer Science and Technology,
Northwestern Polytechnical University, Xi'an 710072, China
{zhanglijun,lizhh,chenbenben,lining}@nwpu.edu.cn

Abstract. XML has been extensively used in many information retrieval related applications. As an important data mining technique, clustering has been used to analyze XML data. The key issue of XML clustering is how to measure the similarity between XML documents. Traditionally, document clustering methods use the content information to measure the document similarity, the structural information contained in XML documents is ignored. In this paper, we propose a model called Structure and Content Vector Model(SCVM) to represent the structure and content information in XML documents. Based on the model, we define similarity measure that can be used to cluster XML documents. Our experimental results show that the proposed model and similarity measure are effective in identifying similar documents when the structure information contained in XML documents is meaningful. This method can be used to improve the precision and efficiency in XML information retrieval.

Keywords: XML Clustering; Content Similarity; Structure Similarity; SCVM.

1 Introduction

Before retrieving the information with search engine, if the large volume of data can be preprocessed, and the unrelated information with user query can be filtered, then the precision and efficiency of the search engine would be improved. Clustering analysis method can be used to preprocess the XML documents. The XML collection can be partitioned into several clusters using clustering technique, where each cluster represent a category. When the query be processing, the cluster that is most relative with user query would be find firstly, then only those documents in this cluster be searched using search engine. Thus the search space can be reduced and the efficiency of the retrieve can be improved.

A document is represented as a term vector in traditional text mining community, where a term is a word in the document. The difference between XML document and text is that XML is semi-structured data, except content information, it also contains structure information. Some existed XML document clustering methods use the content information only, ignored the structure information. If we take account of the semi-structured property of XML data, use

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 116–124, 2010.

both content and structure information contained in the document to clustering, more information can be acquired and the precision of clustering can be improved.

In this paper, we propose a novel XML document representation method: SCVM(Structure and Content Vector Model).An XML document can be viewed as a two-dimensional tuple composed of structure and content vector. Structure and content vector are represented by terms respectively. Based on this, we define the structure and content similarity measure between XML documents, and use it into clustering. Our experiments show that this strategy is effective and can be used to improve the precision of XML document clustering.

The rest of this paper is organized as follows. In Sect.2 we briefly review some related work in the area of XML document clustering. In Sect.3 we describe the structure and content vector model and define a similarity measure for XML documents based on the model. Section 4 presents the experimental results. The paper is finished with some conclusions and discussions for future work for XML documents clustering in Sect.5.

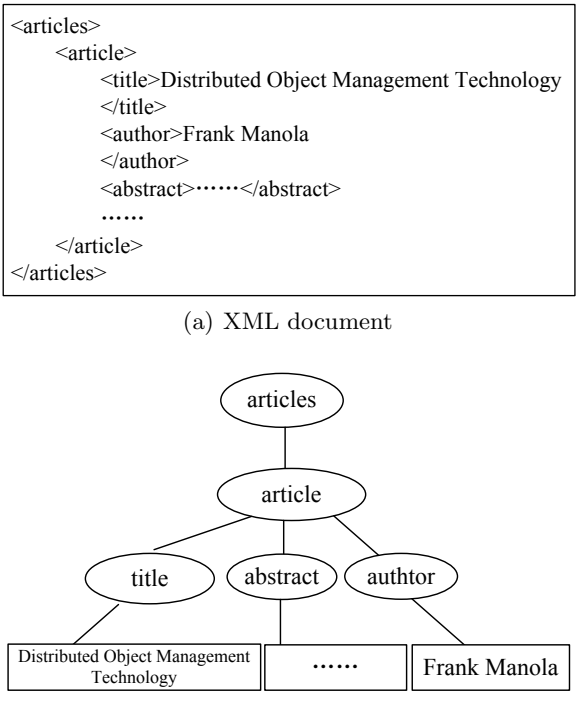
2 Related Work

There are many study on XML documents clustering. The majority methods view XML document as an unordered labeled tree, then compute the structure similarity between documents or schemas based on some tree edit distance[1,2]. However, it is computationally expensive to measure document similarity based on tree edit distance, and it has been proved that the similarity computing is NP-Complete in [3]. Moreover, [1] needs the schema of XML document, but schema is not always existent. Some other methods only use the content information but no structure information in XML documents. An XML document is also viewed as an unordered labeled tree in [4], and is represented as a vector of path extracted from the tree, then the similarity can be computed with this vector. The XML document is also represented as path vector in [5]. The difference with [4] is that only frequently occurring elements are used. [6] view path as feature, though it consider the semantic information, but it doesn't consider the content information contained in text node. Since it ignores the text nodes in the tree when extract path features, so it couldn't cluster the XML documents correctly which are similar in structure but different in content. [7,8] consider the link information except the structure information.

3 Computing of Structure and Content Similarity

3.1 SCVM: Structure and Content Vector Model

An XML document can be represented by an ordered labeled tree which is denoted as $\{V, E, R\}$, where V is a set of label nodes, E is a set of edges from parent nodes to child nodes, and R is the root of the tree. For example, the XML document in Fig.1(a) can be represented as a tree in Fig.1(b).



(a) XML document

(b) XML tree

Fig. 1. XML document and XML tree

We represent the XML document as tree also and extract the structure and content information from the tree. Then the structure and content information can be represented as a vector respectively. Given XML document collection D, every document d_i can be denoted as:

$$d_i =< v\_struct_i, v\_cont_i > \tag{1}$$

where v_struct_i is structure vector, which describes the structure of the document, and v_cont_i is content vector, which describes the content of the document. The two vectors are composed of structure and content terms respectively.

A structure term is a path in the XML tree that from the root node to the parent of leaf node. For example, the structure terms in the XML document shown in Fig.1 include articles/article/title, articles/article/abstract, articles/article/author. Structure term space is composed of all structure terms extracted from all documents in document collection D. Let the size of structure term space is l, then the structure vector of document d_i can be denoted as:

$$v\_struct_i =< stw_{i0}, stw_{i1}, \cdots, stw_{il} > \tag{2}$$

where stw_{ij} is the weight of the jth structure term in d_i.

The word contained in the leaf node(also called text node) is content term of the document. All content terms extracted from all documents in collection D compose content term space. Let the size of content term space is m, then the content vector of document d_i can be denote as:

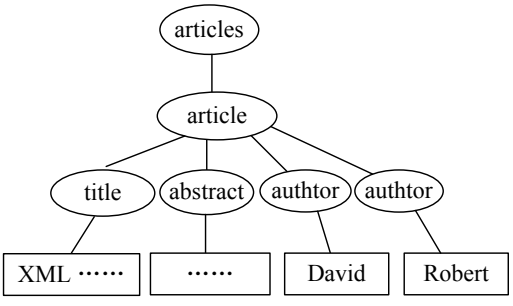
$$v\_cont_i =< ctw_{i0}, ctw_{i1}, \cdots, ctw_{im} > \tag{3}$$

where ctw_{ij} is the weight of the jth content term in d_i.

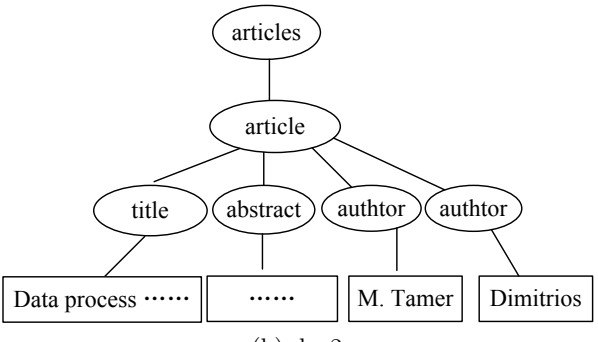
The similarity between XML documents can be computed using structure vector and content vector. Due to consider both structure and content information, if it is used to cluster XML documents, the precision can be improved.

3.2 Structure Similarity

The structure similarity between XML documents can be computed with their structure term vector, the key issue is how to evaluate the weight of each structure term. We observe the more frequent occurrence of a structure term in a pair of XML documents does not mean more similar. For example, though the structure term articles/article/author occurs 2 times in the documents shown in Fig. 2(a) and Fig. 2(b) respectively, but only 1 time in the document shown in



(a) doc1



(b) doc2

Fig. 2. XML document example

Fig.1, it couldn't say that the two documents in Fig.2 are more similar than doc1 and document shown in Fig.1. In fact, according to the content, the document in Fig.1 is more similar with doc2 in Fig.2(b)(Both belong to data management). So just occurring or not in document is taken into account for evaluating the weight of a structure term. The weight can be defined as:

$$stw_{ij} = \begin{cases} 1, & \text{if } st_j \text{ occures in } d_i, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The structure similarity between XML documents d_i and d_j is defined as follows with cosine measure.

$$struct\_sim_{ij} = \frac{v\_struct_i^t \cdot v\_struct_j}{\|v\_struct_i\| \cdot \|v\_struct_j\|} \tag{5}$$

where $\|v\|$ is the Euclidean norm of vector v and v^t is a transposition of vector v.

3.3 Content Similarity

Except for the structure similarity, we should consider content similarity in XML documents clustering. It is mentioned that the content term is the word contained in the text node(include attribute value) of the XML tree in 3.1, so we evaluate the weight of content terms using traditional $tf - idf$ formula.

$$tfidf(ct_j, d_i) = tf(ct_j, d_i) \cdot idf(ct_j) \tag{6}$$

where $tf(ct_j, d_i)$ is the frequency of content term ct_j in XML document d_i. And $idf(ct_j)$ is defined as:

$$idf(ct_j) = \log \frac{|D|}{df(ct_j)} \tag{7}$$

$|D|$ is the size of XML document collection D, $df(ct_j)$ is document number that contain term ct_j. For the weight fall within the range [0,1], we normalize it:

$$ctw_{ij} = \frac{tf(ct_j, d_i) \cdot idf(ct_j)}{\sqrt{\sum_{k=1}^{m}(tfidf(ct_k, d_i))^2}} \tag{8}$$

Same to the structure similarity, we use (5) to evaluate the content similarity between XML documents d_i and d_j.

3.4 XML Document Similarity: Structure and Content Similarity

According to the definitions of structure similarity and content similarity, we can evaluate the document similarity by aggregating them using specified function. In this paper, the document similarity is defined as:

$$sim(d_i, d_j) = (struct\_sim_{ij} + cont\_sim_{ij})/2 \tag{9}$$

We call the document similarity defined by (9) as structure and content similarity.

3.5 Clustering XML Documents Using Structure and Content Similarity

For clustering XML documents using SCVM, preprocess is necessary. Firstly we split each XML document into structure information and content information, then construct the structure and content term space respectively. For content information, filtering the stop words and steming[1] are done before extracting content term. The terms which occur in very few documents and almost all documents are eliminated from the term space, then each document can be represented with SCVM. K-Means algorithm is used to cluster these XML documents.

4 Clustering Results and Analysis

In evaluation of experiment result of structure-based XML documents clustering methods, the classification standard is that the structures of documents are similar or not, and the content is ignored. But the standard of structure and content similarity proposed in this paper is based on the content of the documents. So the work in this paper shouldn't compare with that structure-based clustering methods. To examine the effectiveness of our method, we compare it with traditional content-based clustering methods.

Our comparison is based on two real data sets. The first is Wiki collection from INEX2006. We select 6910 documents which belong to 10 categories. The second data is XML documents collected by a research group of Texas University at Dallas. There are total 101 documents from 20 different sites belonging to 4 categories.

To measure how effectively our proposed method is performed, we use the measure BEP and F_1, which are defined as follows:

$$BEP = \frac{recall + precision}{2}, \quad F_1 = \frac{2 * recall * precision}{recall + precision}$$

Table 1. Clustering result on Wiki collection

Cluster		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	Average
F1	VSM	91.56	91.49	90.14	71.50	94.55	93.64	82.72	87.27	89.95	94.09	88.69
	SCVM	90.95	89.19	88.07	69.02	94.03	93.26	78.39	86.97	88.00	93.36	87.12
BEP	VSM	92.00	91.79	90.16	74.72	94.59	93.72	84.30	87.45	90.23	94.17	89.31
	SCVM	91.37	89.73	88.07	72.83	94.05	93.35	80.92	87.23	88.23	93.41	87.92

Experimental results are shown in Table1 and Table2. In the tables, VSM is the result using traditional vector space model and $tf - idf$ weight evaluation, which make use of content information only, and SCVM is the result using

[1] http://tartarus.org/ martin/PorterStemmer/

Table 2. Clustering result on Texas collection

Cluster		Automobile	Movie	Reference	Software	Average
F1	VSM	100.00	100.00	96.20	93.02	97.31
	SCVM	100.00	100.00	98.77	97.56	99.08
BEP	VSM	100.00	100.00	96.34	93.48	97.45
	SCVM	100.00	100.00	98.78	97.62	99.10

```
<article>
    <name id="2504010">Nap-of-the-earth</name>
    <conversionwarning>0</conversionwarning>
    <body>
        <emph3>Nap-of-the-earth</emph3>(NOE) is flying with electronic
        assistance to ….
        <templateAero_stub NAME="Aero-stub"></templateAero_stub>
    </body>
</article>
```

(a) Wiki document

```
<automobile>
    <manufacturer>Isuzu</manufacturer>
    <model>Isuzu Ascender</model>
    <year>2005</year>
    <engine>
        <type>5.3L 8 Cylinder 300 hp Gas</type>
    </engine>
    <transmission>4 Speed Automatic OD</transmission>
    <feature>
        <safety>ABS Brakes/Driver-Passenger airbags</safety>
        <seat>7</seat>
        <pro>Alarm</pro>
    </feature>
</automobile>
```

(b) Texas document

Fig. 3. Document example in Wiki and Texas collection

method proposed in this paper, which make use of both structure and content information.

We can see that from the results, our method has a better performance on Texas collection than VSM method, the average F_1 and BEP are improved 1.77 and 1.65 respectively. But on Wiki collection, our method is worse than VSM method, the two measurements are reduced 1.57 and 1.39 respectively. Analyzing the documents in the two data sets, we find that the structure information contained in documents of Texas collection is meaningful to whose category, but that in Wiki collection is meaningless. The structure information in Wiki collection is tag for controlling the display format, and they are similar in every category, so it is unmeaning for clustering, and lead to the F_1 and BEP reduced. Fig.3 is an example, Fig.3(a) is a document from Wiki collection, whose structure terms, such as article/body/emph3, etc., are used to control the display format and are unmeaning for document category. Fig.3(b) is a document from Texas collection, whose structure terms, such as automobile/engine/type, etc., can demonstrate it's category in a certain extent.

From the experimental results, we can see that our method works well when the structure information contained in the XML document is meaningful, and SCVM can be used to improve the performance of XML document clustering.

5 Conclusions and Future Work

Based on the characteristic of XML document, we propose a novel model for representing XML document: SCVM(Structure and Content Vector Model), which consider both structure and content information contained in XML document simultaneously. Based on the model, we define the structure and content similarity measure and use it to cluster XML documents. The experimental results show that SCVM has better F_1 and BEP than traditional VSM when the structure information is meaningful to document category. And SCVM can be used to improve the precision of XML document clustering.

Though SCVM take both structure and content information into account, it ignores the semantic information of content term in structure term, such as the content term position in structure term, the structure term number which include a certain content term, etc., how to using these information in XML document clustering is our future work.

Acknowledgments. This work was supported by the National Natural Science Foundation of China under Grant No.60803043 and the National High-Tech Research and Development Plan of China under Grant No.2009AA1Z134.

References

1. Xing, G., Guo, J., Xia, Z.: Classifying XML documents based on structure/Content similarity. In: Fuhr, N., Lalmas, M., Trotman, A. (eds.) INEX 2006. LNCS, vol. 4518, pp. 444–457. Springer, Heidelberg (2007)

2. Dalamagas, T., Cheng, T., Winkel, K.J., Sellis, T.: Clustering XML documents using structural summaries. In: Lindner, W., Mesiti, M., Türker, C., Tzitzikas, Y., Vakali, A.I. (eds.) EDBT 2004. LNCS, vol. 3268, pp. 547–556. Springer, Heidelberg (2004)
3. Zhang, K., Statman, R., Shasha, D.: On the editing distance between unordered labeled trees. Inf. Process. Lett. 42(3), 133–139 (1992)
4. Joshi, S., Agrawal, N., Krishnapuram, R., Negi, S.: A bag of paths model for measuring structural similarity in web documents (2003)
5. Leung, H., Chung, F., Chan, S.C.F., Luk, R.: XML document clustering using common xpath. In: Proceedings. International Workshop on Challenges in Web Information Retrieval and Integration, WIRI 2005, pp. 91–96 (2005)
6. Kim, T.S., Lee, J.H., Song, J.W.: Semantic structural similarity for clustering XML documents. In: Lee, G., Ahn, T.N., Howard, D., Slezak, D. (eds.) International Conference on Convergence and Hybrid Information Technology, Daejeon, South Korea, pp. 552–557. IEEE Computer Soc., Los Alamitos (2008)
7. Yang, J., Cheung, W.K., Chen, X.: Integrating element and term semantics for similarity-based XML document clustering. In: Proceedings of The 2005 IEEE/WIC/ACM International Conference on Web Intelligence, pp. 222–228 (2005)
8. Yang, J., Chen, X.: A semi-structured document model for text mining. Journal of Computer Science and Technology (05) (2002)

pq-Hash: An Efficient Method for Approximate XML Joins*

Fei Li, Hongzhi Wang, Liang Hao, Jianzhong Li, and Hong Gao

The School of Computer Science and Technology, Harbin Institute of Technology
lifei.cscs@163.com, {wangzh,lijzh,honggao}@hit.edu.cn, hl8807@126.com

Abstract. Approximate matching between large tree sets is broadly used in many applications such as data integration and XML de-duplication. However, most existing methods suffer for low efficiency, thus do not scale to large tree sets.

pq-gram is a widely-used method with high quality of matches. In this paper, we propose *pq*-hash as an improvement to *pq*-gram. As the base of *pq*-hash, a randomized data structure, *pq*-array, is developed. With *pq*-array, large trees are represented as small fixed sized arrays. Sort-merge and hash join technique is applied based on these *pq*-arrays to avoid nested-loop join. From theoretical analysis and experimental results, retaining high join quality, *pq*-hash gains much higher efficiency than *pq*-gram.

1 Introduction

For the ability to represent data from a wide variety of sources, XML is rapidly emerging as the new standard for data representation and exchange on the Internet. Given the flexibility of XML model, data in autonomous source which represent the same real world object may not be exactly the same. Thus approximate matching techniques must be applied.

pq-gram is an effective transformation-based tree matching method [2]. In this method, each tree is split into a subtree set (bag) and the *pq*-distance between these sets is used to describe the distance between their corresponding trees. To further increase the efficiency, we propose *pq*-hash as an improvement to *pq*-gram. In our method, trees are first transformed into sets (bags) of *pq*-grams and then hashed into *pq*-arrays: a randomized data structure presented in this paper for faster comparisons. Each *pq*-array has a small fixed size and min-wise hashes [5,10] are applied for the generation of these *pq*-arrays. As a result, two

* Supported by the National Science Foundation of China (No 60703012, 60773063), the NSFC-RGC of China(No. 60831160525), National Grant of Fundamental Research 973 Program of China (No.2006CB303000), National Grant of High Technology 863 Program of China (No. 2009AA01Z149), Key Program of the National Natural Science Foundation of China (No. 60933001), National Postdoctor Foundtaion of China (No. 20090450126), Development Program for Outstanding Young Teachers in Harbin Institute of Technology (no. HITQNJS.2009.052.).

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 125–134, 2010.

trees are considered similar if the numbers in most positions in their corresponding *pq*-arrays equal. We use the sort-merge and hash join technique to further improve the efficiency. Experimental results show that even with high efficiency, the quality of matches based on *pq*-array is good.

To summarize, the main contributions of this paper include:

- A randomized data structure called *pq*-array is developed.
- Based on *pq*-array, we propose a new method called *pq*-hash for efficient approximate matching on large tree sets.
- Theoretical analysis and experimental results confirm that *pq*-hash retains the quality of matches based on *pq*-grams and improves the efficiency significantly when the number of trees is large and trees are huge.

2 Related Work

Most papers that compare XML documents present the XML data as ordered, labeled trees. A well known distance function for trees is the tree edit distance. Many previous works improve the efficiency of tree edit distance computation [15,16,12,8]. However, all of them have at least $O(n^3)$ runtime.

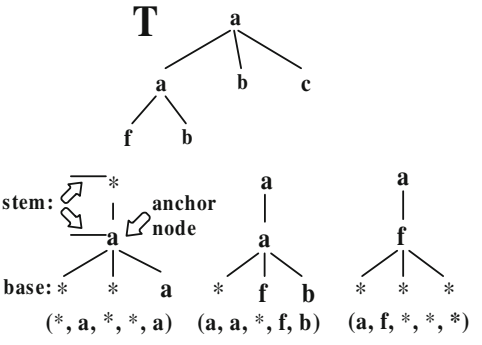
Obviously, the tree edit distance computation is expensive thus do not scale to large trees. Therefore, some previous work transforms trees into other data structures whose similarity is easier to evaluate. In the method of *pq*-gram [2], each tree is transformed into a set of *pq*-grams (a small subtree in a specific shape). By this transformation, the similarity between trees can be evaluated in time $O(n \log n)$. In [1], the *pq*-gram method is extended to evaluate the similarity between unordered trees. The author also provides an optimized join algorithm which in many cases avoids nested-loop join. It has an $O(Nn \log n + Nn \log Nn)$ runtime in the best case, where N is the number of trees in each dataset. However, in many other cases (especially when many tree pairs show high degree of similarity), the runtime is near to the nested-loop join $O(Nn \log n + N^2 n))$. In this paper, we use the Jaccard Coefficient as the metric to evaluate the similarity between bags and give an algorithm which can complete the join in time $O(Nn \log n + N^2))$ in the worst case.

3 Preliminary

pq-gram was introduced by Augsten et al.[2] as an effective method for similarity evaluation between ordered, labeled trees. Intuitively, a *pq-gram* g is a small subtree in a specific shape. It consists of an *anchor node* with its p-1 ancestors and its q consecutive children. The *anchor node* and its p-1 ancestors form the *stem* while the q consecutive children form the *base*.

Definition 1 (*pq*-Bag). *For a tree T, the pq-bag of T is $\{g|\ g$ is a pq-gram of the extended tree of $T\}$. We use the symbol B_T^{pq} to denote the pq-bag of T.*

Example 1. Figure 1(a) shows some 2,3-grams of T. The 2,3-bag of T is shown in Figure 1(b).



(a) some 2,3-grams of T.

pq-bag of T
(*, a, *, *, a)
(a, a, *, *, f)
(a, f , *, *, *)
(a, a, *, f, b)
(a, b, *, *, *)
(a, a, f, b, *)
(a, a, b, *, *)
(*, a, *, a, b)
(a, b, *, *, *)
(*, a, a, b, c)
(a, c, *, *, *)
(*, a, b, c, *)
(*, a, c, *, *)

(b) 2,3-bag.

Fig. 1. Preliminary

The *pq*-distance:$\triangle^{pq}(T_1, T_2) = 1 - 2\frac{B^{pq}(T_1) \bigcap B^{pq}(T_2)}{B^{pq}(T_1) \biguplus B^{pq}(T_2)}$ is used as the distance function based on *pq*-grams.

In their method, each tree is transformed to a large *pq*-bag with $O(n)$ *pq*-grams. $O(n)$ time is needed to compute the *pq*-distance between a pair of sorted *pq*-bags. When join tree sets, all tree pairs in $F_1 \times F_2$ have to be compared. Since there are potential $O(N^2)$ set pairs, the join process may need a lot of time. The goal of this paper is to accelerate the comparisons between these *pq*-bags. Since Jaccard coefficient [14] serves as one of the standard similarity metrics for bags, it is used in this paper as the similarity metric between *pq*-bags.

$$sim(T_1, T_2) \approx Jaccard(B^{pq}_{T_1}, B^{pq}_{T_2})) = \frac{|B^{pq}_{T_1} \bigcap B^{pq}_{T_2}|}{|B^{pq}_{T_1} \bigcup B^{pq}_{T_2}|}$$

4 *pq*-Hash

In this section, a randomized data structure *pq*-array is presented for fast comparison between *pq*-bags. In our method, each *pq*-bag is converted to a fixed sized *pq*-array by two transformation hash functions and a signature hash function. The former transforms a *pq*-bag to a number-set while the latter hashes each large number-set into a small fixed sized array.

4.1 Transformation Hash Function

To hash *pq*-grams easier, each *pq*-gram $g_i \in B^{pq}_T$ is hashed into an integer use the following hash function: $h_1(g_i) = (a_1 * g_i(1) + ... + a_{p+q} * g_i(p+q)) \ mod \ M_1$, where M_1 and a_j are all large prime numbers, $g_i(1)$, ..., $g_i(p)$, ..., $g_i(p+q)$ are the labels of the nodes in g_i in preorder. (for the convenience of hashing, all alphanumeric labels are converted to number labels using Karp-Rabin [11] method)

To compute the Jaccard Coefficient between bags easier, we further hash each number bag into a number set. Let f be the frequency an element w appears

and $h_2(w)=(b_1*w+b_{i+1})\ mod\ M_2$ be a linear hash function. w is hashed into f numbers $w_1, ..., w_i, ..., w_f$, where $w_1 = w$ and $w_{i+1}=h_2(w_i)$.

After the two hash functions, a number set called *pq-set* is generated. We use the symbol S_T^{pq} to denote the *pq*-set of T. According to [10], we have:

$$Jaccard(B_{T_1}^{pq}, B_{T_2}^{pq}) = Jaccard(S_{T_1}^{pq}, S_{T_2}^{pq})$$

4.2 Signature Hash Function

Note that the size of *pq*-sets is linear in the tree size. The merge and intersection between two sorted *pq*-sets need $O(n)$ time. Since there are potential N^2 pairs of sets to be compared, the join process might cost a long time. To compute the Jaccard Coefficient efficiently, we use the min-hash technique [5,10] to transform each *pq*-set into a fixed sized array called *pq*-array. With these *pq*-arrays, the Jaccard coefficient between *pq*-sets can be approximated in $O(1)$ time.

For a *pq*-set S_T^{pq}, let $\pi(x)=(c*x+d) \mathrm{mod}\ M$ be a random linear permutation, where c, d and M are all large prime numbers. Let $S'=\{\pi(x)|x \in S_T^{pq}\}$. Only the minimal element $min \in S'$ is recorded. Thus the set S_T^{pq} is hashed into a min-number denoted by $min_{\pi(T)}$. According to [5], the probability of $min_{\pi(T_1)}=min_{\pi(T_2)}$ equals to the Jaccard coefficient between $S_{T_1}^{pq}$ and $S_{T_2}^{pq}$.

In order to approximate the probability of $min_{\pi(T_1)}=min_{\pi(T_2)}$, k random linear permutations $\pi_1(x)$,...,$\pi_k(x)$ are performed independently. We use the frequency of event $min_{\pi_i(T_1)} = min_{\pi_i(T_2)}$ occurs to approximate that probability. After the k min-hashes, each *pq*-set is hashed to an array with k entries called *pq*-array. We use the symbol A_Π^T to denote the *pq*-array of T. The equal ratio between two *pq*-arrays is the frequency of event $min_{\pi_i(T_1)} = min_{\pi_i(T_2)}$ occurs.

Definition 2 (Equal Ratio). *For a permutation set Π and two trees T_1, T_2, equal ratio $\gamma_{T_1,T_2}^{\Pi}=\frac{|H_{T_1,T_2}|}{|\Pi|}$, where $H_{T_1,T_2} = \{i|A_\Pi^{T_1}[i] = A_\Pi^{T_2}[i]\}$.*

It is proven in Section 5.2 that the probability of $min_{\pi(T_1)}=min_{\pi(T_2)}$ is closely approximated by γ_{T_1,T_2}^{Π} with high probability. γ_{T_1,T_2}^{Π} is computed in O(1) time.

4.3 Algorithms for XML Join

The core of our algorithms is the generation *of* *pq*-arrays. Using the algorithm framework in [2], the hash algorithms are shown in Algorithm 1, 2, 3.

For efficient transformation, two shift regs, *stem* and *base*, are used to store the stem and base of the current *pq*-gram. The operation shift(*reg*,*el*) returns reg with its oldest element dequeued and *el* enqueue. The operation *stem* ∘ *base* returns the *pq*-gram with a stem of *stem* and a base of *base*. After the transformation, each tree is transformed into an array with fixed size.

Using the algorithms above, all the trees are transformed into *pq*-arrays. Although the equal ratio between *pq*-arrays can be computed in $O(1)$ time, nested-loop join between the *pq*-array still takes a long time when N is very large. Instead of computing the equal ratio between each *pq*-array pair independently,

Algorithm 1. Get-*pq*-Array

Input:T, p, q, k; **Output:**$pq - Array_T$
stem:=initialize *stem* register full with *
pq-Bag=**Get-*pq*-Bag**(T, *root*(T), p, q, *pq-Bag*, *stem*)
pq-Set=h_2(*pq-Bag*)
pq-Array$_T$=**MinHash**(*pq-Set*, k)
return *pq-Array*$_T$

Algorithm 2. Get-*pq*-Bag

Input:T, r, p, q, *pq-Bag*, *stem*; **Output:***pq-Bag*
stem:=shift(*stem*, r)
base:=initialize *base* register full filled with *
if r is a leaf node **then**
  pq-number=h_1(*stem* ∘ *base*)
  pq-Bag:=*pq-Bag*⊎*pq-number*
else
  for all children c (from left to right) of r **do**
    base:=shift(*base*, c)
    pq-number=h_1(*stem* ∘ *base*)
    pq-Bag:=*pq-Bag*⊎*pq-number*
    pq-Bag:=**Get-*pq*-Bag**(T, c, p, q *pq-Bag*, *stem*)
  for $j = 1$ to q-1 **do**
    base:=shift(*base*, *)
    pq-number=h_1(*stem* ∘ *base*)
    pq-Bag:=*pq-Bag*⊎*pq-number*
return *pq-Bag*

we check for each min-number in which pairs of *pq*-arrays it appears. We apply sort-merge and hash join technique to optimize this join.

After the process of *pq*-hash, each tree is transformed into a *pq*-array with k numbers. We denote each min-number by a tuple (tree-Id, value, position). Tree-Id records which tree the min-number belongs to while value records the value of that min-number. The position records the min-number's position in its *pq*-array. We put all these tuples in a source together to form a List. Then we sort the List by the value of the min-number. Using the Algorithm 4, the equal ratio between each pair of *pq*-arrays can be computed without nested-loop.

5 Analysis

5.1 Efficiency Analysis

Let $O(n)$ be the tree size and N be the number of trees. According to the algorithms above, the join on F_1 and F_2 based on *pq*-array has three steps:

1. All the trees in F_1 and F_2 are transformed into *pq*-sets.
2. k-MinHashes are used to hash all *pq*-sets into *pq*-arrays.

Algorithm 3. MinHash

Input: pq-Set, k; **Output:** $pq - Array_T$
for all the pq-numbers in pq-set **do**
  for $i = 1$ to k **do**
    $min{=}\pi_i(pq\text{-}number)$
    if $min < pq\text{-}Array[i]$ **then**
      $pq\text{-}Array[i]{=}min$
return $pq\text{-}Array_T$

Algorithm 4. Optimized Join for *pq*-Arrays

Input: two Sorted List, k, τ; **Output:** Join Result
for $List_1$ and $List_2$ **do**
  $List_i \leftarrow \rho_{treeId/treeId_i}$
$List'{=}List_1 \bowtie List_2$
$List''{=}\Gamma_{treeId_1,treeId_2,COUNT/k \rightarrow EqualRatio}(List')$
return $\pi_{treeId_1,treeId_2}(\sigma_{EqualRatio \geq \tau}(List''))$

3. "Optimized join" between these pq-arrays is conducted and the set $\{(T_i, T_j) | T_i \in F_1, T_j \in F_2, \gamma^{\Pi}_{T_i,T_j} \geq \tau\}$ is outputted, where τ is the threshold.

In step 1, since the transformation for a pq-set takes $O(n \log n)$ time, it totally costs $O(Nn \log n)$ time for generation of all the pq-sets in F_1 and F_2.

In step 2, it totally costs $O(kNn)$ time to min-hash all the pq-sets into pq-arrays. Since k is a constant, the time cost in this step is $O(Nn)$.

In step 3, our join algorithm can take advantage of the diversity of trees in forests. In the best case, when no pq-array has equal min-number in the same position, the runtime is $O(N \log N)$. Even in the worst cases, when the joined forests consist of identical copies of the same tree, our join algorithm still have a $O(N^2)$ time complexity. Note that the join results are N^2 tree pairs in the worst cases, thus no algorithm can improve the quadratic runtime.

In sum, our join algorithm has an $O(Nn \log n + N \log N)$ runtime in the best case and an $O(Nn \log n + N^2)$ runtime in the worst case.

5.2 Random Error Analysis

In our method, we use many hash functions (h_1, h_2, k-Minhash). Since all the parameters (a_1,...,a_{p+q}, M_1 in h_1; b_i, M_2 in h_2; c, d, M in each π) are large random prime numbers, the probability of collision is extremely low [9]. The error caused by collision is almost 0 with carefully selected parameters.

The main random error is resulted in by using equal ratio between pq-arrays to approximate the Jaccard Coefficient between pq-sets. For each random linear hash function π, we use P to denote the probability of $min_{\pi(T_1)}{=}min_{\pi(T_2)}$. In our method, min-hash is performed independently for k times. This process is a series of Beronouli trials. By counting the frequency t that event $min_{\pi(T_1)}{=}min_{\pi(T_2)}$ occurs, P is approximated by the equal ratio t/k. We use random error to describe the difference between t/k and P. The Random Error r is $|t/k - P|$.

The random variable in the Beronouli trials is denoted by X. Since Beronouli trials are independent with each other, X fits the binomial distribution. The average random error is computed with the following formula:

$$\overline{r} = \sum_{t=0}^{n} P(X=t) r_P(t) = \sum_{t=0}^{n} \binom{k}{t} P^t P^{k-t} |\frac{t}{k} - P|$$

Intuitively, the average random error decreases with the increase of k. When P and k are fixed, $\overline{r}$ can be computed by the equation above. Quantitatively speaking, when k is larger than 400, no matter what the Jaccard coefficient P is, the average random error of this approximation is smaller than 0.02.

We can also set a threshold ν and use the symbol $P(s)$ to denote the probability that our random error is within ν. We have the following equality:

$$P(s) = \sum_{t=\lceil k(P-\nu)\rceil}^{\lfloor k(P+\nu)\rfloor} P(X=t) = \sum_{t=\lceil k(P-\nu)\rceil}^{\lfloor k(P+\nu)\rfloor} \binom{k}{t} P^t P^{k-t}$$

Quantitatively speaking, when k is larger than 400, the probability that our random error within ν=0.05 is larger than 0.95.

6 Experiments

6.1 Efficiency

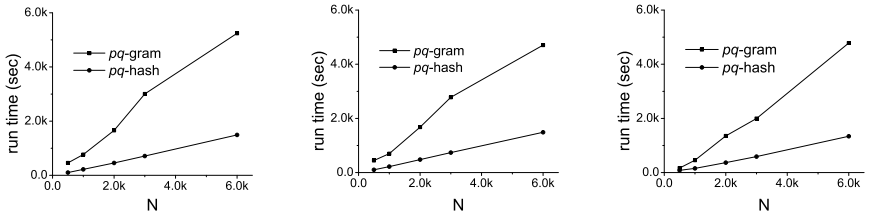
We use the Swissprot[1] (bioinformatics), Treebank[2] (linguistics) and DBLP[3] (bibliography) XML databases to generate large tree sets. Small documents are combined randomly to generate larger trees. In this section, the large trees we deal with have about 5000 nodes on average.

We compare our method with the "optimized join" using *pq*-grams [2,1][4]. We choose our parameters p=2, q=3, k=400. Since these sorted *pq*-bags can be updated incrementally [3], we do not take the time of sorted *pq*-bag generation into account. The results are shown in Figure 2(a)-2(c). Since the size of the List we need to sort-merge and hash join is $O(N)$ while the size of Index in method *pq*-gram is $O(Nn)$, our method outperforms *pq*-gram significantly in efficiency. We also test the impact of k on time cost and show the results in Figure 3(a)- 3(c). The runtime increases linearly with k.

6.2 Accuracy Experiments

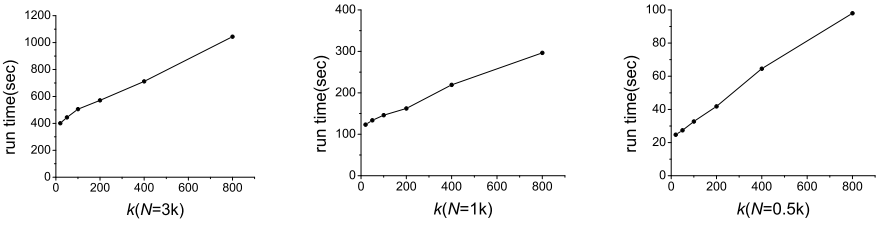
Data Recall Experiments. In this experiment, we add noise to the Swissprot, Treebank and DBLP databases to stimulate the case of spelling mistakes and

[1] Swissprot: http://www.expasy.ch/sprot/
[2] Treebank:http://www.cis.upenn.edu/ treebank/.
[3] DBLP: http://dblp.uni-trier.de/xml/
[4] We have obtained the source code from the authors and we modify their code to apply the "optimize join" on ordered trees.



(a) Efficiency (Swissprot) (b) Efficiency (Treebank) (c) Efficiency (DBLP)

Fig. 2. Efficiency on publicly available XML



(a) Swissport (b) Treebank (c) DBLP

Fig. 3. The impact of k on runtime

missing elements. We randomly relabel and delete a number of nodes on the original documents and approximately join the original and the noisy set.

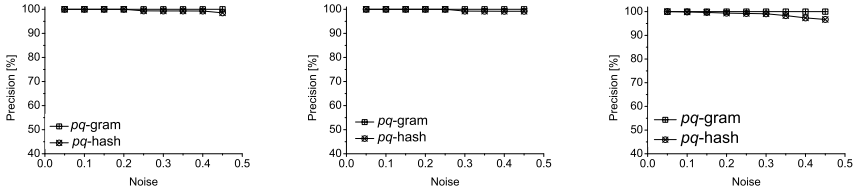
For each original document d, we match if d has only one nearest neighbor d' in the modified set and the reverse is also true. We define the precision as the ratio between correct match and all the matches ($precision=\frac{correct}{correct+incorrect}$). The recall ratio is defined as the ratio between correct match and the number of the documents ($recall=\frac{correct}{N}$).

We compare our method with *pq*-gram [2]. In our method, we choose 400-MinHashes. Figure 4 shows precision and recall ratio for different percentage of changed nodes in different database. In each database, our method behaves nearly as well as *pq*-gram.

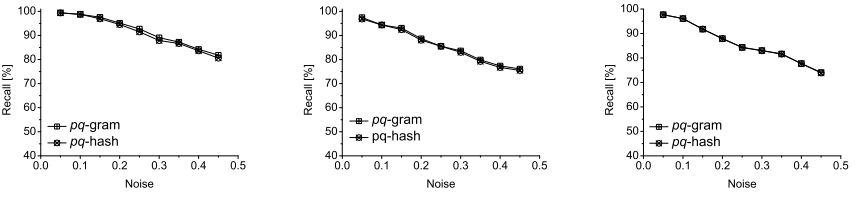
Matchmaking with Real Data. To test the accuracy for real world data, we use an application data from the Municipality of Bozen[5]. The Office wants to integrate data about apartments stored in two databases and display that information on a map. We denote the two sources as R and L. We match T_R and T_L iff T_R is the nearest neighbor of T_L and T_L is the nearest neighbor of T_R.

We compute mappings using the *pq*-distance [2] and the tree edit distance [15]. We also test the impact of k on accuracy. The accuracy of our method is nearly as good as *pq*-gram when we use 400-MinHashes. In the cases both the efficiency and accuracy have to be considered, the recommended value of k is 400.

[5] We get the data from the authors in [2].



(a) Precision (SwissProt). (b) Precision(Treebank). (c) Precision (DBLP).



(d) Recall (SwissProt). (e) Recall (Treebank). (f) Recall (DBLP).

Fig. 4. Recall Tests

	recall	correct	false pos.
edit distance	82.9%	248	9
2,3-gram	79.3%	237	4
2,3-hash(k=400)	78.6%	235	5

Fig. 5. Street Matching Result Using Different Methods

7 Conclusions

In this paper, we propose an effective and efficient method of *pq*-hash for the approximate join of ordered trees. The synopsis of our method is to transform trees into fixed sized *pq*-arrays and use the equal-ratios between the *pq*-arrays to evaluate the similarity between their corresponding trees. Instead of computing the equal-ratio between each pair of *pq*-arrays, we apply the sort-merge and hash join based on these *pq*-arrays. We theoretically analyze the efficiency and random error of our method. Experiments confirm the efficiency and match quality of our method and suggest that our technique is both useful and scalable.

In the future, we will try to combine our theory with more applications like duplication detecting. We are also interested in extending our method to a broader range of hierarchical data like unordered trees, free trees and directed acyclic graphs.

Acknowledgements

Many thanks to Michael H. Böhlen [1,2,3] for his source code and test data; and the Municipality of Bolzano for the street matching data.

References

1. Augsten, N., Böhlen, M.H., Dyreson, C.E., Gamper, J.: Approximate joins for data-centric XML. In: ICDE, pp. 814–823 (2008)
2. Augsten, N., Böhlen, M.H., Gamper, J.: Approximate matching of hierarchical data using *pq*-grams. In: VLDB, pp. 301–312 (2005)
3. Augsten, N., Böhlen, M.H., Gamper, J.: An incrementally maintainable index for approximate lookups in hierarchical data. In: VLDB, pp. 247–258 (2006)
4. Bille, P.: A survey on tree edit distance and related problems. Theor. Comput. Sci. 337(1-3), 217–239 (2005)
5. Broder, A.Z., Charikar, M., Frieze, A.M., Mitzenmacher, M.: Min-wise independent permutations (extended abstract). In: STOC, pp. 327–336 (1998)
6. Cobena, G., Abiteboul, S., Marian, A.: Detecting changes in XML documents. In: ICDE, pp. 41–52 (2002)
7. Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J.D., Yang, C.: Finding interesting associations without support pruning. IEEE Trans. Knowl. Data Eng. 13(1), 64–78 (2001)
8. Demaine, E.D., Mozes, S., Rossman, B., Weimann, O.: An optimal decomposition algorithm for tree edit distance. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 146–157. Springer, Heidelberg (2007)
9. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: VLDB, pp. 518–529 (1999)
10. Haveliwala, T.H., Gionis, A., Indyk, P.: Scalable techniques for clustering the web. In: WebDB (Informal Proceedings), pp. 129–134 (2000)
11. Karp, R.M., Rabin, M.O.: Efficient randomized pattern-matching algorithms. IBM Journal of Research and Development 31(2), 249–260 (1987)
12. Klein, P.N.: Computing the edit-distance between unrooted ordered trees. In: Bilardi, G., Pietracaprina, A., Italiano, G.F., Pucci, G. (eds.) ESA 1998. LNCS, vol. 1461, pp. 91–102. Springer, Heidelberg (1998)
13. Lee, K.-H., Choy, Y.-C., Cho, S.-B.: An efficient algorithm to compute differences between structured documents. IEEE Trans. Knowl. Data Eng. 16(8), 965–979 (2004)
14. Metwally, A., Agrawal, D., Abbadi, A.E.: Detectives: detecting coalition hit inflation attacks in advertising networks streams. In: WWW, pp. 241–250 (2007)
15. Tai, K.-C.: The tree-to-tree correction problem. J. ACM 26(3), 422–433 (1979)
16. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM J. Comput. 18(6), 1245–1262 (1989)

TwigLinkedList: Improvement of TwigList*

Zengqi Gao, Husheng Liao, Hongyu Gao, and Kechao Yang

Beijing University of Technology, Beijing, China
{zengqigao,yakecoco}@gmail.com, {liaohs,hygao}@bjut.edu.cn

Abstract. To deal tree pattern matching (twig query) issue for XQuery processing, *TwigList* [1] proposes a high efficient solution. However, it does not consider the out-of-orderness issue which is required in XQuery. And, like most existing approaches *TwigList* pay little attention to optimization for the parent-child relationship. In this paper, we propose a novel way to handle both the out-of-orderness issue and optimization of parent-child relationship by improving *TwigList*, named *TwigLinkedList*. Detailed performance test shows that our algorithm reaches and exceeds the level of *TwigList*, while reducing redundant intermediate results.

1 Introduction

Tree pattern matching (twig query) is the key issue for XQuery processing. The left side of Fig. 1 is a simple query tree. A twig query tree includes two types of edges: /-edge and //-edges, which represent parent-child (PC) and ancestor-descendent (AD) relationship respectively. Recently, *Twig*^{2}*Stack* [2] proposes a hierarchical-stack encoding scheme to compactly represent the twig results. This encoding scheme can save the structural relationship between XML nodes. Then they propose algorithms to construct this encoding scheme and enumerate results from it. At the same time, Lu Qin proposes *TwigList* algorithm which uses simple lists instead of complex hierarchical-stacks, thus simplifying the algorithm and improving efficiency significantly. Then, based on the *Tag Streaming* [3], Jiang Li and Junhu Wang [4] further improve the efficiency.

Compared with others, *TwigList* is nearly the most efficient one, and it is not dependent on *Tag Streaming* which reduces the generality. But it doesn't provide solution for the out-of-orderness issue and optimization for the parent-child relationship. In this paper, We propose a novel way to handle both the out-of-orderness issue and the parent-child relationship by improving *TwigList*, named *TwigLinkedList*. This method will improve the efficiency while reducing the redundancy of the intermediate results. Detailed performance test shows that our algorithm *TwigLinkedList* reaches and exceeds the level of *TwigList*.

* Supported by: 1) Beijing Municipal Natural Science Foundation (4082003); 2) Discipline and Graduate Students Education Project of Beijing Municipal Commission of Education.

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 135–140, 2010.

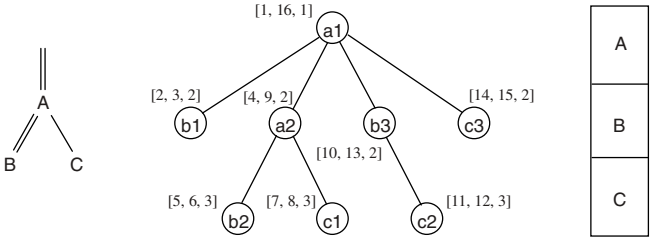


Fig. 1. A sample XML document tree and a twig query

2 TwigList

TwigList uses simple lists to store the structure relationship between XML elements. Consider the twig query "$//A[.//B]/C$" in Fig. 1, Fig. 2 shows the data model of *TwigList*. According to the pointers in this figure, $a2$ is ancestor of $b2$ and $c1$, $a1$ is ancestor of all elements in L_B and L_C.[1]

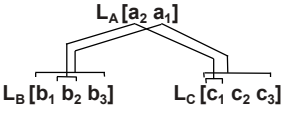


Fig. 2. Data model of TwigList

To construct the lists, *TwigList* uses a stack, S. Elements are pushed into the stack in preorder, and the start pointer will be setted. As the use of the stack S, elements will be poped in postorder, and each of them will be checked to see whether it should be appended to the corresponding list.

3 Out-of-Orderness Issue

The results of expression evaluation in XQuery should be arranged in document order. *TwigList* cannot solve this issue since all XML elements stored in corresponding lists are in postorder. So we need to keep them in preorder by inserting them into the "right" places. Based on property of *TwigList*: All XML elements are visited in preorder, the "right" place is the place after last XML element in the corresponding list when first time visits them.

Algorithm 1 shows the construct algorithm of *TwigLinkedList*. *v.insert* in line 12 is used to save the "right" place. For example, Fig. 3.(a) shows the status before first time visiting of $a1$. Because $a1$'s corresponding list is empty, $a1$'s "right" place is the head of list. Fig. 3.(b) shows the status before the insertion

[1] We assume the XML elements with labels in lower-case letter match the query nodes with labels in the corresponding upper-case letter. For instance, $a1$ matches node A.

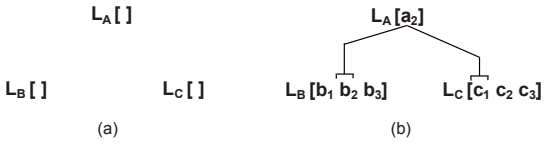


Fig. 3. The data model before visit to XML elmenent $a1$

of $a1$. Then $a1$ will be inserted in the head of corresponding list. The insertion will change the index of elements following v, so we use pointer and linked list instead of index and array, so it is called *TwigLinkedList*.

4 Optimization for Parent-Child Relationship

4.1 Motivation

Twig^2*Stack* handles PC relationship by using hierarchical stack. But it needs more resources to maintain the hierarchical stack. *TwigList* uses simple data structure, but they didn't pay more attention to PC relationship.

We introduced a property above: all XML elements were visited in preorder. PC relationship is just living inside the preorder. For example, all XML elements in Fig. 1 in preorder is as $(a1, b1, a2, b2, c1, b3, c2, c3)$. And the nearest ancestor of each element is its parent.

4.2 TwigLinkedList

We use a list, called *Level List*, to handle the PC relationship. *Level List* is a list P with m elements $\{U_1, ..., U_m\}$, U_i is a query node matched with the last visited XML element of level i $(1 \leq i \leq m)$. The right side of Fig. 1 shows an example of *Level List* before visit to $c2$. A, B and C are query nodes matched with the last visited XML elements in each level. In Algorithm 1, procedure $getQueryNode()$ is called in line 6 to compute the matched query node in twig query of XML element according to the *Level List*.

The procedure $getQueryNode()$ deals with PC relationship and AD relationship separately. When deal with PC relationship, this it returns v's matched query node V_k *iff* the matched query node V_j of v's parent has v kind child node V_k in twig query. For example, in Fig. 1, when visit to $c2$, $c2$ will be filtered because B (the matched query node of $c2$'s parent) doesn't have C-typed child in twig query. Otherwise this procedure will return matched query node according AD relationship by traversing the twig query. In conclusion, we use $getQueryNode()$ to filter XML elements by examining PC relationship.

The *Level List* is also used to create sibling pointer since it stores the father's information of current XML element.

Algorithm 1. The contruct algorithm of TwigLinkedList

Input : A query tree Q with n nodes $\{V_1, ..., V_n\}$, a sequence of XML nodes in preorder X, a list P with m elements$\{U_1, ..., U_m\}$, U_i is a query node matched with the last visited XML element of level i $(1 \leq i \leq m)$

Output: All L_{V_i}, for each $1 \leq i \leq n$

initialize stack S as empty;
initialize U_i as $\emptyset$, for each $1 \leq i \leq m$;
initialize all list L_{V_i} as empty for all $V_i \in V(Q)$ (The length of L_{V_i} is 0);
while $X \neq \emptyset$ **do**
    let v be the top element in X;
    $V_q \leftarrow getQueryNode(v, Q, P, level(v))$;
    remove v from X;
    if $V_q \neq$ null **then**
        $toList(S, Q, reg(v))$;
        foreach child of V_q in query tree Q, V_p **do**
            $v.start_{V_p} \leftarrow length(L_{V_p}) + 1$;
        $v.insert \leftarrow length(L_{V_q}) + 1$;
        $push(S, v)$;
$toList(S, Q, (\infty, \infty))$;

Procedure: $toList(S, Q, r)$

while $S \neq \emptyset \wedge r \not\subseteq reg(top(S))$ **do**
    $v_j \leftarrow pop(S)$;
    let v_j 's type be V_j, found by $getQueryNode()$ before;
    foreach child of V_j in query tree Q, V_k **do**
        $v_j.end_{V_k} \leftarrow length(L_{V_k})$;
    insert v_j into L_{V_j} at $v_j.insert$ if $v_j.start_{V_k} \leq v_j.end_{V_k}$ for V_j 's children, V_k;
    $v_j.lastDescendent \leftarrow length(L_{V_j})$;

Procedure: $getQueryNode(v, Q, P, level)$

if $U_{level-1} \neq$ null **then**
    $V_j \leftarrow U_{level-1}$;
    foreach child of V_j in query tree Q, V_k **do**
        if relation between V_k and V_j is PC $\wedge v$ matchs V_k **then**
            $U_{level} \leftarrow V_k$;
            return V_k;
return $processAD(v, Q, root(Q))$;

Procedure: $processAD(v, Q, V_r)$

$V_k \leftarrow V_r$'s parent in Q;
if relation between V_k and V_r is AD $\wedge v$ matchs V_r **then**
    return V_r;
foreach child of V_r in query tree Q, V_p **do**
    $r \leftarrow processAD(v, Q, V_p)$;
    if $r \neq$ null **then**
        return r;
return null;

Time/Space Complexity. Time complexity of the construct algorithm is the same as *TwigList*, because we do not change the main frame of *TwigList*. The space complexity will be a little more than *TwigList*. Because of *Level List*, our algorithm will spend $O(|L|)$ more space then *TwigList*, where $O(|L|)$ is the depth of XML document tree.

Enumeration of Results. Enumeration algorithm is different with present algorithms. It is modified for the need of XQuery. Normally, we only need part of result. For example, C is the query node that needs to be returned in XPath $//A[.//B]/C$. Detailed enumeration algorithm will not be discussed in this paper.

5 Experimental Evaluations

5.1 Experimental Setup

We implemented our *TwigLinkedList* algorithm and *TwigList* algorithm using Java 1.4.2 and performed experiments on a PC with a Intel(R) Core(TM) 2 Duo T7100-1.79GHz processor and 2GB of main memory.

A set of synthetic and real datasets are used for the experimental evaluation. In particular, the synthetic dataset includes XMark (113MB) [5]. The two real datasets included are DBLP (127MB) and TreeBank (82MB) from [6]. We compared the two twig algorithms in terms of the query processing time and memory usage. Table 2 shows all twig queries used for the experiments.

Table 1. Twig queries used for the experimental evaluation

Query Name	Dataset	Twig Query
DBLP-Q1	DBLP	//dblp/inproceedings[title]/author
DBLP-Q2	DBLP	//dblp/article[author][.//title]//year
DBLP-Q3	DBLP	//inproceedings[author][.//title]//booktitle
XMark-Q1	XMark	/site/open_auctions[.//bidder/personref]//reserve
XMark-Q2	XMark	//people//person[.//address/zipcode]/profile/education
XMark-Q3	XMark	//item[location]/description//keyword
TreeBank-Q1	TreeBank	//S/VP//PP[.//NP/VBN]/IN
TreeBank-Q2	TreeBank	//S/VP/PP[IN]/NP/VBN
TreeBank-Q3	TreeBank	//VP[DT]//PRP_DOLLAR_

5.2 Experimental Result

Fig. 4 depicts the performance results. Overall, our algorithm reached and exceeded the level of *TwigList* both in query processing time and memory usage. The reason is that *TwigLinkedList* can handle the PC relationship very well, can build the sibling pointer very fast.

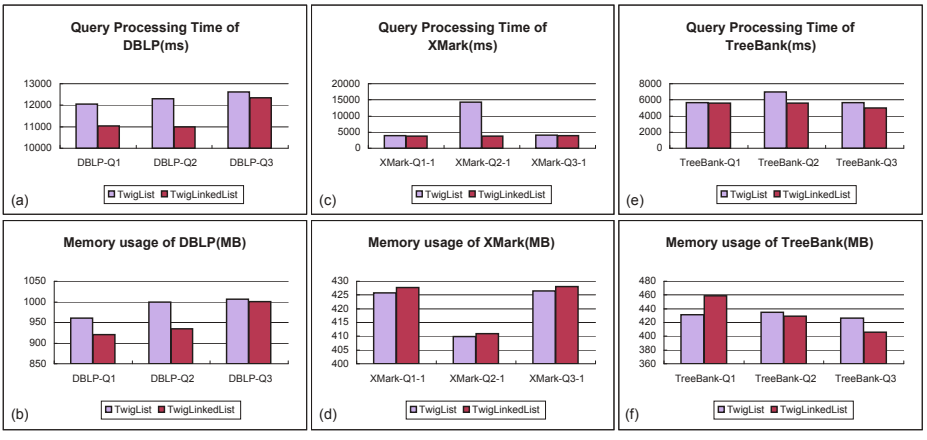


Fig. 4. Result of twig query processing on DBLP, XMark and TreeBank datasets

Because reduced redundant intermediate results are not enough and linked list may cost more space then array, especially for query TreeBank-Q1, the memory usage of *TwigLinkedList* is not leading *TwigList* in all queries.

6 Conclusions

In this paper, we proposed a novel way to handle the out-of-orderness issue and the PC relationship by improving *TwigList*, named *TwigLinkedList*. Performance test showes our algorithm reaches and exceeds the level of *TwigList*.

As the future work, we plan to add early enumeration to save the space. Further more, we plan to study deeply about how to eliminate redundant intermediate results and how to add full support of predication.

References

1. Qin, L., Yu, J.X., Ding, B.: Twiglist: Make twig pattern matching fast. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 850–862. Springer, Heidelberg (2007)
2. Chen, S., Li, H.G., Tatemura, J., Hsiung, W.P., Agrawal, D., Candan, K.S.: Twig^{2}stack: Bottom-up processing of generalized-tree-pattern queries over XML documents. In: VLDB, pp. 283–294. ACM, New York (2006)
3. Chen, T., Lu, J., Ling, T.W.: On boosting holism in XML twig pattern matching using structural indexing techniques. In: Proceedings of SIGMOD, pp. 455–466 (2005)
4. Li, J., Wang, J.: Fast matching of twig patterns. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) DEXA 2008. LNCS, vol. 5181, pp. 523–536. Springer, Heidelberg (2008)
5. Schmidt, A.R., Waas, F., Kersten, M.L., Florescu, D., Manolescu, I., Carey, M.J., Busse, R.: The XML benchmark project. Technical report, CWI (2001)
6. U: XML Repository, http://www.cs.washington.edu/research/xmldatasets/

Compression Algorithms for Structural Query Results on XML Data*

Qing Wang, Hongzhi Wang, Hong Gao, and Jianzhong Li

Institute of Computer Science and Technology, Harbin Institute of Technology,
150001 Harbin, China
21651865@163.com,
{wangzh,honggao,lijzh}@hit.edu.cn

Abstract. XML is an important data form. The size of query results on XML data is very large in some applications and in the distributed XML data management system, the transmission of query results become the bottleneck. For the efficiency of XML management system, the compression method for XML query results is studied in this paper. By exploiting the properties of query results on XML data, two compression methods are proposed. One is a G-bitmap algorithm as a variant of bitmap index. The other is based on tree structure. The experiments on real and manual data demonstrate the effectiveness and efficiency of our compression methods.

Keywords: Bitmap, Structural query, XML, Compression, Bit storage.

1 Introduction

XML (eXtensible Markup Language)[1] is a markup metalanguage designed to enable semantics-aware tagging of World Wide Web information. Because of its scalability and flexibility, XML has become the standard of data representation and exchange on the web. One of important problems of XML data management is how to query XML data efficiently. Among the queries, structural query to matching a pattern in an XML document is an important kind of queries.

In an XML query processing system, the size of query result may be very large, especially for structural queries. When the query is performed in a distributed environment, the transmission cost may be large for such large result set.

In order to reduce the size of query results on large XML data effectively, we propose two applicable compress algorithms for query results on XML data. One is a G-bitmap algorithm, a variant of bitmap index. This method uses a bitmap to represent the query set and then compresses such bitmap. The other method represents the result set as a tree and compression is performed on such tree structure.

* Supported by the National Science Foundation of China (No 60703012, 60773063), the NSFC-RGC of China(No. 60831160525), National Grant of Fundamental Research 973 Program of China (No.2006CB303000), National Grant of High Technology 863 Program of China (No. 2009AA01Z149), Key Program of the National Natural Science Foundation of China (No. 60933001), National Postdoctor Foundtaion of China (No. 20090450126), Development Program for Outstanding Young Teachers in Harbin Institute of Technology (no. HITQNJS.2009.052).

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 141–145, 2010.

There are two distinct contributions in this paper: 1) two effective compression algorithms are proposed for the compression of structural query result of XML data; and 2) theoretical analysis and extensive experiments are performed to demonstrate the benefits of these two compression methods.

The rest part of this paper is organized as follows. Section 2 analyzes the problem and exploits the properties of query results. Section 3 presents two new compression algorithms with performance analysis. Experimental results of the algorithms are shown in Section 4. Section 5 concludes the whole paper.

2 The Compressions Algorithms for XML Query Results

As an observation, we exploit the four properties of a structural query result set S on XML data. P1. Every tuple in S has the same number of nodes. P2. No duplicate nodes are in the same tuple. Otherwise it will lead to the wrong semantic. P3. S has many repetitive nodes. P4. The number of the tuple in S may be very large.
Based on the properties discussed above, two compression algorithms are presented in this section in different aspects.

Firstly we discussed the G-bitmap method. G-bitmap, a bitmap-based compression method, is presented based on the properties of the query result set. The idea is evolved from the bitmap index [2, 3]. The basic idea of this method is to represent each tag with one bit and each tuple is converted to one bit array. Such that the whole result set is converted to a bitmap with each tuple as a row.

The compression phase has two steps. The first is to construct the bitmap from original result set. The second is to compress the bitmap.

In the first step, a bitmap M' is constructed. Each row in M' represents a tuple and each column in M' represents a uniform node in the result set. If the j-th tuple contains the i-th node, then M'i,j=1. Otherwise M'i,j=0. According to P1, all rows in M' have the same number of 1's. In the second step, M' is compressed to a bit array CM'. M' is scanned row by row. In each row r, the number of segments with successive 0-bits is recorded as c. For each c, a number k is computed as following.

$$k = \begin{cases} \lfloor \log_2 c \rfloor + 1 & (c > 0) \\ 1 & (c = 0) \end{cases}.$$

For each segment of successive 0-bits in r, k-1 bit 1 and one bit 0 are appended to CM', and then the binary representation of c is appended to CM'. All rows in M' are compressed one by one in above way. As the result, the bit array CM' is generated.

In step 2, from the start position of M', we record the number of successive 0-bits when we scan the first 1-bits. Number=0. Therefore, c =0, k =1. It means one bit 0 but no bit 1 is to be appended to CM'. Then the binary representation of c (0) is appended to CM'. Currently, CM'=00. Such process is repeated till the last element in the matrix.

In the decompression phase, the compressed bitmap CM' is decompressed to the bitmap M' at first. Then the query set is obtained from M'. The idea of converting CM' to M' is as following. Without loss of generality, we consider that case that the kth bit of CM' is accessed. Following bits in CM' from CM'[k] are scanned till CM'[k']=0 is met. j=k'-k. Then next j bits are visited. i is the integer representation such j bits in binary. We insert the successive i of 0s and one of 1 to the end of current row in M'. Initially, k=0. The above process is started from the bit CM'[k+2j].

We analyze the best compression ratio and the worst compression ratio.

$CR_{Worst} = \frac{2\times(W+1)\times\log_2(N-W) - 2\times W\times\log_2(W+1)}{N}$, where W is the number of 1s in all rows in M'. And N is the number of distinct nodes in the whole result set.

$CR_{Best} = \frac{2\times(\log_2(N-W)+W-1)}{N}$.

Time complexity analyzes. Since the compression and decompression phases both scan data once, the time complexity of compression is $\Theta(NC)$, and the time complexity of decompression phase is $O(NC)$.

We discussed the other method in the following paragraph. we present a tree-structure compression method to represent the result set as a forest and compress it. From the observation in the trees, the pointers occupy the lion's share of the storage space. To avoid the storage of the pointers, every tree is stored as a sequence in a link list and the content of every node is stored in layers.

By merging the common prefix of all tuples, the result set M is converted to a forest. With the P3 that the result set has many repetitive nodes, if many repetitive nodes are stored as a representative, much space will be saved. It is the basic idea of this method.

The compression phase has two steps.

M has following two intuitive rules about invariant.

Rule 1(element transformation) For $t_{i,j} \in t_i, t_{i,k} \in t_i (t_i \in M, j \neq k)$, $t_{i,j}$ and $t_{i,k}$ are swapped and the new row is t_i'. The result constructed from t_i is same as that from t_i'. **Rule 2(row transformation)** For $t_i \in M, t_j \in M$ (i ≠ j), t_i are t_j swapped an a new matrix M' is generated. The result set represented by M' is same as M.

Our method is performed recursively. For the input matrix M, based on the above two rules, transformation on rows and columns are performed to make M be able to be partitioned to multiple sub-matrices, the rows in each of which share same prefix and can be represented as a tree. For each sub-matrices m, the common prefix of rows is stored as the value in this layer. The remaining part of m is partitioned and stored in the method same as M but with deeper layer.

In the second step, the converted forest is stored in layer. As discussed in Step 1, in each layer l, the content of the prefix and the number of nodes sharing such content are to be stored. Deeper layers of l are stored with l consecutively. Such that the storage of the pointers from l to deeper layers are avoided.

The number of nodes in the tree or graph structure XML document is denoted by k. Then the number of the sub-matrices of a layer ranges from 1 to k. In this case, the representation of each layer in variant-length code is more effective than invariant-length code. In the variant-length code, at first the prefix is stored in variant mode. Then we use the next $\left\lceil \frac{\log k}{8} \right\rceil$ bits to represent the number of bytes to store such layer. The following logk-bits represent the number of sub-matrices in this layer. The code of the sub-matrices of this layer is stored in following bits and the last bit identifies the end of the layer, if the last bit equal to 1, it means the end of the layer.

In the decompression phase, firstly we load the tree structure from the file and then decompress the tree-structure to a matrix M. The key step of decompression is decompressing the tree-structure to a matrix M.

The number of tuples in M is denoted as C, and N denotes the number of nodes in every tuple. The definition of the compression ratio was presented in section 2.2, we just analyze the best and the worst compression ratio. Obviously, the best case is that the forest contains just a tree and the paths from the root to leaf differ only in leaves. In the best case,

$$CR_{best} = \frac{2 \times C + 5 \times (N-1)}{sizeof(\text{int}) \times C \times N}$$

And the worst situation is that the first nodes of every row are different. In the worst case,

$$CR_{worst} = \frac{3 \times C \times N}{sizeof(\text{int}) \times C \times N} = 0.75$$

Time complexity analyzes the total time complexity of compression and decompression is $O(C \times C \times N)$.

3 Experiments

In this section, we show experimental results to evaluate the effectiveness and efficiency of compression algorithms. Our experiments focused on two important aspects: (1) the CR discussed in Section 3 for the effectiveness. (2) The run time of compression for the efficiency.

Table 1. All the properties of real data set

File name	File size	R	W
1937-4.txt	53k	1937	4
407313-6.txt	16.1M	407313	4

In the experiments, all algorithms are implemented in C++ with SGI STL. The synthetically data set is generated with following parameters: the number of tuple in the query result set (C); The tuple width (W); The number of nodes(N). The similarity P as the probability that ti,j=ti+1,j. The information of real data is shown in Table 1. The default setting of the synthetically is C=20,000, W=4, N=5,000, P=0.9.

We compare our algorithms with Huffman coding on real data set. The CRs are shown in Table 2.

Table 2. The result on the real data set

R	W	CR of G-bitmap	CR of Tree-structure	CR of Huffman coding
407313	4	0.35	0.13	0.24
1937	4	0.29	0.22	0.31

From the result, we note that the tree-structure storage shows the best effectiveness whenever the C is small or big, and the Huffman coding is worse than tree-structure storage. G-bitmap algorithm is the worst compression method.

This following paper tests the impact of the four parameters referred in section 3 on the effectiveness. We obtained the following conclusions from the below figures .1) the CR of G-bitmap is almost stable with C when The CR of tree-structure increases;2) the CR of G-bitmap decreases with W. However, that of tree-structure is opposite.3) the tree structure storage is almost unchanged with N when G-bitmap is growing; 4) P has no impact on G-bitmap. The compression ratio of tree-structure decreases with P. It is obviously.

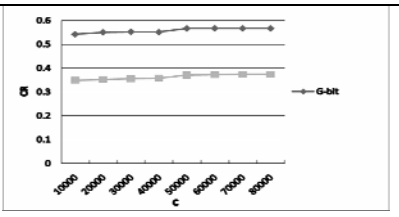


Fig. 1. The trend with change of C

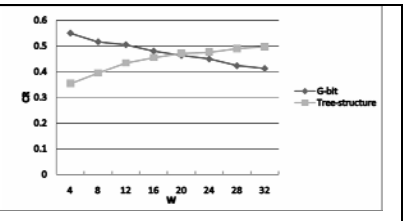


Fig. 2. The trend with change of W

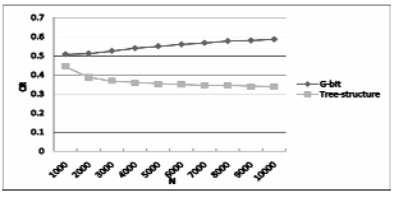


Fig. 3. The trend with change of N

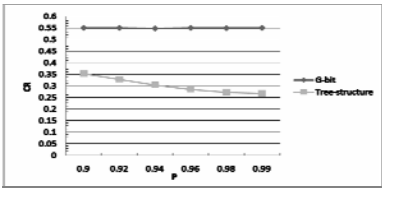


Fig. 4. The trend with change of P

4 Conclusion

The paper proposed two novel compression algorithms to store the structural query result set on XML data. These algorithms make the best usage of the properties of structural query results. They are suitable for both tree-structured and graph-structured XML data. Analysis and extensive experiments are performed to demonstrate the efficiency and effectiveness of these algorithms. From the results, tree-structure method is better in compression ratio while G-bitmap is more efficient. As described in the above sections, we proposed two novel compression algorithms to store the structural query result set on XML data. My future task is how to query on the structural query result set after compression.

References

[1] Lu, Z.: A Survey Of XML Applications On Science And Technology. International Journal of Software Engineering and Knowledge Engineering (IJSEKE) 15(1), 1–33 (2005)

[2] Bookstein, A., Klein, S.T.: Compression of correlated bit-vectors. Inf. Syst (IS) 16(4), 387–400 (1991)

[3] Garcia-Molina, H., Ullman, J.D., Widom, J.: Database System Implementation. Prentice-Hall, Englewood Cliffs (2000)

SeCCX: Semantics-Based Fine Granular Concurrency Control for XML Data

Chuitian Rong[1,2], Wei Lu[1,2], Xiao Zhang[1,2], Zhen Liu[1,2], and Xiaoyong Du[1,2]

[1] Key Labs of Data Engineering and Knowledge Engineering, Ministry of Education, China
[2] School of Information, Renmin University of China, China
{rct682,uqwlu,zhangxiao,liuzhen,duyong}@ruc.edu.cn

Abstract. This paper proposes a new locking protocol, SeCCX, for isolation of concurrent transactions on XML data. This protocol adopts the semantics of operations issued by users. Comparing with previous XML locking protocols, SeCCX has some new features, like richer lock modes, finer granular, lower lock confliction and dynamic lock escalation. The importance is that it locks the objects according to the semantics of user's operation and sets different restrictions on different objects. It provides finer granular concurrency control such that the overall system performance can be improved. We prove that the SeCCX protocol is conflict serializability. We also compare it with other protocols to demonstrate the advantages of SeCCX.

1 Introduction

The eXtensible Markup Language (XML) has a variety of applications, including information storage, data exchange, content management, web services and data integration. For years, XML has become a research hot spot, including indexing, keyword search. Recently, there has been a major focus on concurrency control for XML documents [1,2,3,4,5,6,7] such that we can provide efficient access. Similar to the concurrency control in relational databases, operations in a transaction must be isolated from those of other transactions. Also, these transactions must assure the correctness of data.

Example 1. Consider an XML document tree[1], shown in Figure 1, where details of orderings from customers are recorded and maintained. Assume there exist two transactions: (1) transaction T_1 is to obtain information of the customers with the Xpath expression: "/order/header//name". All nodes that T_1 accesses in the XML document tree are shown in the Figure 2; (2) transaction T_2 is to update the price of the product with the Xpath expression: "/order/price". Similarly, all nodes that T_2 accesses in the XML document tree are shown in the Figure 3. As T_2 only updates the value of the text node shown in Figure 3, and other element nodes are irrelevant to T_2, T_1 and T_2 can be executed concurrently. □

Based on the above example, from the perspective of the operations, transaction T_1 (query) conflicts with transaction T_2 (update) on the same document. However, if we

[1] It was downloaded from the website:
http://www.ibm.com/developerworks/cn/xml/jguru-dom/index.html

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 146–155, 2010.

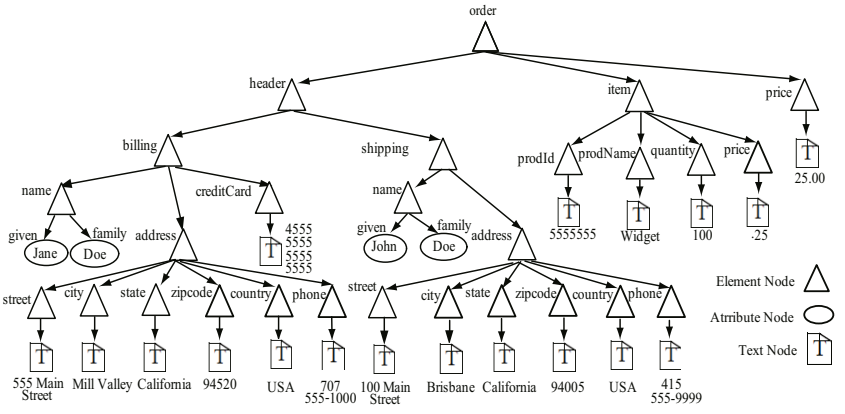


Fig. 1. An XML document tree

further consider the nodes that T_1 and T_2 function on, we find that these two transactions operate on different subtrees, and operations on each node do not conflict with each other. Thus, T_1 and T_2 can be executed concurrently.

Example 2. Consider the same XML document tree shown in Figure 1 again. Assume there exist two transactions: (1) transaction T_3 performs the operations as T_1 does in Example 1; (2) transaction T_4 is to rename the label of element node "billing", which is shown in Figure 2, with the Xpath expression "/order/header/billing". As described above, since operations from T_3 and T_4 on the element node "billing"conflict, T_3 and T_4 cannot be executed concurrently.

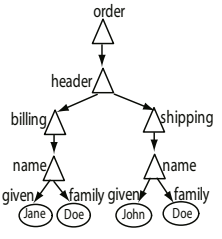


Fig. 2. Subtree for T_1

Fig. 3. Subtree for T_2

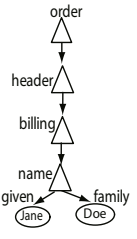


Fig. 4. Subtree for T_4

Intuitively, if two operations on the same node conflict with each other, then one of them must wait until the other transaction submits or rolls back. However, if we further consider the situation in the above example, no matter whether T_4 renames the label "billing" or not, T_3 can still find all nodes that it tries to read information of customers correctly, including the node "billing". As a result, in this paper, we concentrate on how to improve the *throughput of transactions*, i.e., increase the number of submitted transactions per unit of time. To sum up, we make the following contributions:

– We propose a finer granular locking protocol: SeCCX. Existing locking protocols have some limitations: (1) if a node is set to exclusive mode (update operation), then no operation is allowed on the subtree rooted with the node, and all its ancestor nodes [5,6]; (2) coarse granularity of operations results in pseudo conflict and phantom phenomenon [1]; (3) predicates must be introduced to providing finer granular concurrent control. However, such predicates suffer from rather expensive computation cost [2,3]. In this paper, we propose locking protocol based on semantics of operations. Specifically, some other primitive operations are introduced, where user's operations can be mapped to some primitive operations and these primitive operations providing high concurrency.
– In any database management system (DBMS), conflict serializability is an essential requirement, including XDMS [8]. In order to assure the serializability of transations, the acquisition and release of locks must obey the two-phase locking protocol. In this paper, we prove that the schedule of transactions compliance with SeCCX protocol is isolated.

The rest of this paper is organized as following. In Section 2, we give some preliminaries to help readers comprehend this paper, including symbol definitions and operation analysis. In section 3 we presents our new SeCCX protocol, including its lock modes, lock compatibility matrix, protocol rules and conflict detections. In section 4 we analyze SeCCX and make a comparison with other protocols. Section 5 is related work. The last is the conclusion of this study.

2 Preliminary

2.1 Symbols and Definitions

The interfaces for operating XML data are mostly based on Xpath. Let xp denote an Xpath expression. For a given xp, we use $Ex(xp)$ and $Im(xp)$ to denote the nodes that are referred explicitly and implicitly, respectively. For example, in Xpath expression, "/order/header//name", $Ex(xp) = \{$order, header, name$\}$ and $Im(xp) = \{$billing, shipping$\}$. As the precondition of accessing node "name" is the existence of its ancestor,

Table 1. Symbols and Definitions

Symbol	Definition
xp	Xpath expression,like "/order/header//name"
$Ex(xp)$	collection of nodes that accessed explicitly in Xpath expression
$Im(xp)$	collection of nodes that accessed implicitly in Xpath expression
$Rs(xp)$	collection of nodes, each of which is the target of Xpath expression
$Parent(n)$	the parent node of node n
$Ancestors(n)$	collection of ancestors of node n
$IsP(n_1,n_2)$	determine whether the node n_1 is the parent of node n_2
$IsA(n_1,n_2)$	determine whether the node n_1 is one of the ancestors of node n_2
$O(n)$	operation on node n, $n \in Rs(xp)$,$O \in \{R,I,D,Re,U\}$
$< O(n),xp >$	operation in a transaction

Table 2. Operation Analysis

Operation	Object	Effect to Document Tree
query	node or subtree	no
insert	node or subtree	affect one level, only add a subtree or a node
update	node or subtree	affect a node's content or a subtree's structure
rename	node	node name, no impact on the document tree structure
delete	subtree	affect the document tree structure, delete a subtree

{billing, shipping} are the nodes that we access implicitly. $Rs(xp)$ is the final result of the query, it also is a nodes collection. For this expression, $Rs(xp)$ is two nodes named "name" in Figure 2. $Parent(n)$ denotes the parent node of node n. $Ancestors(n)$ denotes ancestor nodes of node n. $IsA(n_1,n_2)$ and $IsP(n_1,n_2)$ are two functions that used for locking mechanism. They are used to verify whether two nodes are in the parent-child or ancestor-descendant relationship. $O(n)$ denotes an operation on node n. For the sake of brevity, symbols and their definitions listed in Table 1 are used throughout this paper.

2.2 Operations

Xpath [9] and XQuery [10] are two important query languages for XML. XUpdate is an XML update language[11]. XQuery Update Facility is a new recommendation of W3C, which defines an update facility that extends the XQuery [12]. Xpath, XQuery and XUpdate all model an XML document as a tree of nodes and the latter two are based on Xpath. The primitive operations that are supported in XUpdate and XUQuery update are shown in column one and column two of Table3, respectively. Based on the analysis of these operations, we propose five primitive operations, which are shown in the last column of Table 3, to design our concurrency protocol. For any given operation from users, we can transform it into one or multiple primitive operations. For better understanding, we list objects on which these primitive operations can perform, and the effect to the document tree by these operations.

Basically, an Xpath expression includes structural constraints and predicates. The Xpath expression consists of a location path [9], which in turn consists of a sequence of one or more location steps separated by the symbol '/' or '//', where the content between two successive symbols is called a location step. The universal form of a location step is: $Axis :: Node[Predicate]$, where Axis specifies the node relationships (e.g. parent, child or sibling, etc.) between the nodes referred in successive location steps. The location path and location steps are considered as structural constraints. In contrast, predicate is used to filter the results from nodes satisfying the structural constraint $Axis :: Node$ of a location step. The result of an Xpath expression is a set of nodes satisfying the structural constraint and predicates in each location step of the Xpath expression [1].

Xpath-like query and update languages are navigational and they are similar to regular expression. Some important observations are made from the Xpath's access behavior that may be useful in designing an efficient concurrency protocol for Xpath-like operations.

Observation 1. *Each axis in the location steps specifies the direction of navigating in the document tree. The element(node) in the last step is the objects on which operations occur.*

Observation 2. *The nodes that occur explicitly in the location steps have different significance from the nodes that were implicitly referred in Xpath expression. The explicitly occurred nodes in the Xpath expression can not be updated by any other transaction concurrently, they must keep the same until the current transaction commits. Thus, we don't need to put the same strict restrictions on the nodes accessed implicitly as the nodes accessed explicitly.*

3 SeCCX Protocol

This section describes the SeCCX protocol, including its lock modes, lock compatibility matrix and protocol rules. Followed by conflict detection used during locking process and isolation analysis of SeCCX.

3.1 Lock Modes

SeCCX has nine lock modes, denoted by P, IR, R, IX, Rn, II, N, D and SX. Before executing an operation, specific lock modes must be applied and acquired. These nine lock modes are explained in detail in the following.

P lock: This lock is a sharing lock designed for the node in the $Im(xp)$. While executing a location step, nodes that are accessed implicitly are locked by this lock mode.

IR lock: For any query operation $< R(n), xp >$ in a transaction, before executing the query we must get IR locks on the nodes in $Ancestors(n) - Im(xp)$, i.e., we set IR lock on the nodes in the Ancestors(n) but not in the $Im(xp)$. Also, we set IR locks on the nodes in $Ex(xp) - Ancestors(n)$.

R lock: For any query operation $< O(n), xp >$ in a transaction, any node in $Rs(xp)$ should be locked by R lock mode before we do the query operation physically.

IX lock: For any update operation $< O(n), xp >$ in a transaction, $O \in \{I, D, U, Re\}$, $n \in Rs(xp)$, each node in the $Ancestors(n) - Im(xp)$ will be locked by IX lock. Also, we should set IX lock on the nodes in $Ex(xp) - Ancestors(n)$.

II lock: This is used for insert operation. Nodes in $Rs(xp)$ are set to II lock. This lock mode is compatible with P, IR,and IX, i.e., it permits an insert operation on the node although there is a query or update operation under this node on the same subtree.

D lock: This is used for delete operation. For any node $n \in Rs(xp)$, nodes in $Parent(n)$ are set to D lock, not the node to be deleted as we should do some operations on the node's parent.

Rn lock: It is set on the node in $Rs(xp)$ when the operation is rename.

N lock: This is used for preventing phantom phenomenon, it is set on the new inserted or updated node or new subtree's root.

SX lock: It is subtree exclusive mode and used for locking escalation. Once a transaction is set with this lock successfully, it will access the subtree exclusively.

Table 3. XML operations

XUpdate	XQuery Update	Our Porposed
query	Query	Query
insert-before	Insert	Insert
insert-after	Delete	Delete
append	Replace	Update
update	Rename	Rename
remove	Transform	

Granted lock

Requested lock	–	P	IR	R	IX	SX	Rn	II	N	D
P	+	+	+	+	+	–	+	+	–	–
IR	+	+	+	+	+	–	–	+	–	–
R	+	+	+	+	–	–	–	–	–	–
IX	+	+	+	–	+	–	–	+	–	–
SX	+	–	–	–	–	–	–	–	–	–
Rn	+	+	–	–	–	–	–	–	–	–
II	+	+	+	–	+	–	–	–	–	–
N	+	–	–	–	–	–	–	–	–	–
D	+	–	–	–	–	–	–	–	–	–

Fig. 5. SeCCX lock compatbility matrix

Algorithm 1. *bool LockProcessForDelete(Node)*

input : *Node*: The node to be locked
output: *bool*, whether locking process is succeed

```
1  foreach node_i in Ancestors(Node) do
2      if IsP(node_i, Node) then LockXMLNode(node_i, D);
3      else
4          if node_i ∈ Ex(Node) then LockXMLNode(node_i, IX);
5          else LockXMLNode(node_i, P);
6      end
7  end
8  foreach node_i in Ex(Node)-Ancestors(Node) do
9      LockXMLNode(node_i, P);
10 end
11 return true;
```

Given two lock modes, we can check whether they conflict with each other base on the lock compatibility matrix that is described in Figure 5.

3.2 Protocol Rules

In this section, we propose three protocol rules which must be obeyed when using SeCCX.

- **2PL rule.** For any lock mode in SeCCX, its acquisition and release must obey the strong strict two phase locking protocol.
- **Locking rule.** If there is a locking request, the locking manager will apply specific lock mode on some nodes according to operation type and locking protocol. As there are relationships between different nodes, this protocol requires that when we lock a specific node we must have set proper locks on its ancestor nodes. All the locks are set from root node to the target node (current node). While, when the transaction is completed the locks are released from bottom to top. In Algorithm 1, we give the pseudo code of locking process for delete operation. The locking process for other operations is similar to deletion.

Algorithm 2. *bool LockEscalation(Node)*

```
   input  : Node: The root node of the subtree
   output: bool, whether lock escalation is done
 1 LockCount ← LockNum(Node);
 2 NodeCount ← DescendantNum(Node);
 3 if LockCount/NodeCount < Threshold then return false;
 4 else
 5     if IsLockedbyMe(Node,IX) then
 6         LockXMLNode(Node,SX);
 7         //release all the lock on the subtree exclude the new one
 8         LockReleaseSubtree(Node);
 9     else
10         LockXMLNode(Node,R);
11         LockReleaseSubtree(Node);
12     return true;
13 end
```

– **Escalation rule.** If a transaction already has some locks on a subtree, the lock manager will compute the proportion between the lock number and the nodes number in the subtree(line 3). If the proportion value gets the predefined threshold the lock manager will trigger a lock escalation process(line 4-11). The lock manager will give the transaction a lock on the subtree's root node, and release the locks on the nodes that belong to the subtree. The mode of the lock that set on the subtree's root is determined by the locks mode that set on the nodes of the subtree. If there is any exclusive lock mode under the root node the root node will be set *SX* lock(line 6). If there are only share mode locks we set the root node *R* lock(line 10). The details of lock escalation is shown in Algorithm 2. This algorithm calls other functions, such as *LockXMLNode*, which completes the locking process on a node; *LockReleaseSubtree* is used to release the locks in a subtree's nodes; *LockNum* and *DescendantNum* is to get statistics.

3.3 Conflict Detection

Conflict detection is processed during the locking process. The result of conflict detection determines the lock's grant. This section gives the details of conflict detection. We define a transaction T as a sequence of operation pairs $< O(n), xp >$, O is the symbol of the five primitive operations in Table 2. We just use the prefix of the operation name such as $:R, I, U, D$ and Re.

Let T_1 and T_2 be any two transactions in a schedule, xp_1 and xp_2 be any two Xpath expressions, $n_1 \in Rs(xp_1)$, $n_2 \in Rs(xp_2)$. If their operations and the objects of the operations satisfy any of the following conditions, there is a conflict.

– $< \mathbf{R(n_1)}, \mathbf{xp_1} > \in \mathbf{T_1}, < \mathbf{D(n_2)}, \mathbf{xp_2} > \in \mathbf{T_2}$
  If $Ancestors(n_1) \subset Ancestors(n_2)$ or vice versa, there is a confliction.
– $< \mathbf{R(n_1)}, \mathbf{xp_1} > \in \mathbf{T_1}, < \mathbf{I(n_2)}, \mathbf{xp_2} > \in \mathbf{T_2}$
  If $Ancestors(n_1) \subset Ancestors(n_2)$, there is a confliction.

- $< \mathbf{R}(\mathbf{n_1}), \mathbf{xp_1} > \in \mathbf{T_1}, < \mathbf{U}(\mathbf{n_2}), \mathbf{xp_2} > \in \mathbf{T_2}$
  If $Ancestors(n_1) \supset Ancestors(n_2)$ or vice versa, there is a confliction.
- $< \mathbf{R}(\mathbf{n_1}), \mathbf{xp_1} > \in \mathbf{T_1}, < \mathbf{Re}(\mathbf{n_2}), \mathbf{xp_2} > \in \mathbf{T_2}$
  If $(Ancestors(n_1) \supset Ancestors(n_2)$ and $n_2 \in Ex(xp))$ or $Ancestors(n_1) \subset Ancestors(n_2)$, there is a confliction.
- $< \mathbf{D}(\mathbf{n_1}), \mathbf{xp_1} > \in \mathbf{T_1}, < \mathbf{U}(\mathbf{n_2}), \mathbf{xp_2} > \in \mathbf{T_2}$
  If $Ancestors(n_1) \supset Ancestors(n_2)$ or vice versa, there is a confliction.
- $< \mathbf{Re}(\mathbf{n_1}), \mathbf{xp_1} > \in \mathbf{T_1}, < \mathbf{O}(\mathbf{n_2}), \mathbf{xp_2} > \in \mathbf{T_2}, \mathbf{O} \in \{\mathbf{D}, \mathbf{U}\}$
  If $Ancestors(n_1) \supset Ancestors(n_2)$ or $(Ancestors(n_1) \subset Ancestors(n_2)$ and $n_1 \in Ex(xp_2))$, there is a confliction.
- $< \mathbf{I}(\mathbf{n_1}), \mathbf{xp_1} > \in \mathbf{T_1}, < \mathbf{O}(\mathbf{n_2}), \mathbf{xp_2} > \in \mathbf{T_2}, \mathbf{O} \in \{\mathbf{D}, \mathbf{U}\}$
  If $Ancestors(n_1) \supset Ancestors(n_2)$, there is a confliction.

3.4 Isolation of SeCCX

Isolation theorems are usually called the serialization theorems. Isolation is used for consistency with the ACID terminology. The acronym CAPS (consistency, atomicity, persistence, serializability) is an alternative mnemonic [13]. In this section we analyze the Isolation of SeCCX. According to [13] we have two theorems that can help us to prove the isolation or serializability.

Locking Theorem: *If all transactions are well-formed and two-phase, then any legal history will be isolated.*

Wormhole Theorem: *A history is isolated if, and only if, it has no wormhole transactions.*

Our SeCCX protocol adheres to strong strict two-phase locking protocol that is described in section 3.2. Specifically, for a given transaction, after releasing a lock, it cannot apply for any lock. Furthermore, when a transaction commits or is rolled back, all its locks need to be released. According to the locking rule in section 3.2, we have three requirements which are listed below.

- For read operation in a transaction, any object that belongs to this operation is locked by share mode locks until the transaction completes.
- For update operation in a transaction, we also require to lock all objects to the exclusive mode which is the target of this update operation. These locks are kept until the transaction is committed.
- For any lock acquired in a transaction must be released at the end of the transaction.

All three requirements described above assure the transaction is well formed according to [13]. Thus, if a transaction adheres to our SeCCX protocol must be well-formed and two-phase. According to the locking theorem, the schedule(history) produced under the SeCCX protocol is isolated. If the schedule of the concurrent transactions is isolated, then there is no wormhole among these concurrent transactions according to the wormhole theorem. In graphical terms, there is no cycle in the dependency graph, i.e., he schedule or history is equivalent to a serial schedule since no transaction is both before and after another transaction [13].

There are four definitions of users for the degree of isolation, and each degree of isolation has different requirements for locking protocols. We have given detailed descriptions about SeCCX in Section 3; any transaction that adheres to this protocol is well-formed and two-phase. Thus, schedules produced under this protocol can provide Degree 3 isolation.

4 Related Work

Because of the particular character of XML documents, including flexible data structure and navigational query language, the concurrency control mechanism in RDBMS cannot be applied in XML databases. As a result, a variety of new locking-based protocols for XML databases have been proposed. In summary, these protocols can be partitioned into three categories, including Xpath-based protocols [1],DataGuide-based protocols [2,3,4] and DOM-based protocols [5,6,7].

For Xpath-based protocols, XLP [1] doesn't support rename operation directly. Instead, it use delete and insert operations to realize this operation, this will cause pseudo conflicts. As a result, rename operation and read operation on the same subtree in different transactions can not be done concurrently. Further, it allows phantoms.

For DataGuide-based protocols in [2,3,4]. The locks of this kind protocol are set on the node of DataGuide not the document tree. Unfortunately, the node in the DataGuide usually corresponds to two or more nodes in the document tree. If want to access a node or a subtree, we must use predicate to differentiate the nodes which have the same names. As determination of predicate is NP, all these kind protocols suffer from expensive computing. It also causes pseudo conflict [3].

For DOM-based protocols in [5,6], those transactions in examples can not be done concurrently, either. These kind protocols are DOM API oriented, and they all designed for DOM like operations. The concurrency for Xpath like operations is not evaluated yet. They all do not permit an update operation occurring above another transaction on the same subtree.

In summary, all those protocols proposed before didn't utilize the semantics of the user's operation and they all treated the nodes in the document tree as the same. In fact, the nodes accessed by a transaction usually don't have the same significance to the current transaction. The other protocols don't permit an update transaction occurring with other transactions on the same subtree regardless of the type of the other transactions. Although some protocols allow the above types of operations to some extent, still, phantom may happen. In SeCCX, we differentiate the nodes into two different collections based on semantics of operations and set different restrictions on them; it has richer lock modes. Thus, SeCCX can provide finer granularity concurrency control than others.

5 Conclusions and Future Work

In this paper, we analyze operations on XML data and its effects to the document tree. Further, we analyze the semantics of user's operation and give out detailed analysis on operation conflictions. Based on our observations and detailed analysis, we propose the

SeCCX protocol for XML collaboration processing. In SeCCX, we give different restrictions on different nodes by utilizing the semantics of operations; in locking process we set different lock modes on nodes accessed by operation explicitly and implicitly respectively according to SeCCX. The serializable assurance and isolation characteristic of SeCCX is proved. Finally, we give the comparisons with other proposed protocols for XML.

At the moment, a hybrid system that provides uniform management for relational and XML data is under our development. In the near future we will integrate SeCCX into our hybrid system and test it in real applications.

Acknowledgements

This work is partly supported by National 863 High-Tech Program(No. 2009AA01Z149), National Science Foundation of China(No. 60873017) and Graduate Research Fund Project of Renmin University of China(No.10XNH097).

References

1. Jea, K., Chen, S.: A high concurrency XPath-based locking protocol for XML databases. Information and Software Technology 48(8), 708–716 (2006)
2. Grabs, T., Böhm, K., Schek, H.: XMLTM: efficient transaction management for XML documents. In: Proceedings of the Eleventh International Conference on Information and Knowledge Management, p. 152. ACM, New York (2002)
3. Pleshachkov, P., Chardin, P.: A Locking Protocol for Scheduling Transactions on XML Data. In: Proceedings of the Spring Young Researcher's, Citeseer, vol. 7, p. 90204 (2004)
4. Pleshachkov, P., Kuznetcov, S.: SXDGL: Snapshot Based Concurrency Control Protocol for XML Data. In: Barbosa, D., Bonifati, A., Bellahsène, Z., Hunt, E., Unland, R. (eds.) XSym 2007. LNCS, vol. 4704, p. 122. Springer, Heidelberg (2007)
5. Haustein, M., Harder, T.: taDOM: A Tailored Synchronization Concept with Tunable Lock Granularity for the DOM API. In: Kalinichenko, L.A., Manthey, R., Thalheim, B., Wloka, U. (eds.) ADBIS 2003. LNCS, vol. 2798, pp. 88–102. Springer, Heidelberg (2003)
6. Helmer, S., Kanne, C.C., Moerkotte, G.: Evaluating lock-based protocols for cooperation on XML documents. ACM SIGMOD Record 33(1), 58–63 (2004)
7. Mathis, C.: Storing, indexing, and querying XML documents in native database management systems. Ph.D. dissertation, University of Kaiserslautern, München (2009)
8. Helmer, S., Kanne, C., Moerkotte, G.: Timestamp-based Protocols for Synchronizing Access on XML Documents. In: Galindo, F., Takizawa, M., Traunmüller, R. (eds.) DEXA 2004. LNCS, vol. 3180, pp. 591–600. Springer, Heidelberg (2004)
9. Xpath, W3C Recommendation, http://www.w3.org/TR/xpath20
10. XQuery, W3C Recommendation, http://www.w3.org/XML/Query
11. XUpdate, W3C Recommendation, http://www.w3.org/TR/xpath20
12. XQuery Update Facility, W3C Recommendation, http://www.w3.org/TR/xquery-update-10
13. Gray, J., Reuter, A.: Transaction processing: concepts and techniques. Morgan Kaufmann Pub., San Francisco (1993)

XRCJ: Supporting Keyword Search in XML and Relation Co-occurrence

Song Zhang and Xiaoyong Du

Key Lab of Data Engineering and Knowledge Engineering, Ministry of Education
School of Information, Renmin University of China
zsyuyizhang@gmail.com,duyong@ruc.edu.cn

Abstract. Keyword search is a proven, user-friendly way to query html documents in the World Wide Web. With the trend of integrating IR technologies into DB community, state-of-the-art studies have introduced keyword search into XML and relation. In this paper we introduce keyword search into a new circumstance called *XRC* (XML and relation co-occurrence) where XML and relational data exist at the same storage. By exploring semantic relationships between data from relation and XML, we enable users who intend to visit data from XML and relation to access information in an easier manner. To achieve this target, we propose a novel operation called *XRCJ* which connects data from XML and relation together when answering search query. We also improve traditional inverted index and design a mixed inverted index. Through conducting extensive experiments on synthetic data from real data set DBLP, we find that our method can achieve a good performance in XML and relation co-occurrence circumstance.

1 Introduction

Keyword search is a proven, user-friendly way to query html documents in the World Wide Web. As it emerged as one of the most effective paradigm to query information, a user no longer needs to understand the underlying knowledge of data or to learn a complex query language. Besides the success of html-based keyword search, state-of-the-art techniques also applied keyword search to XML and relational database [1.2.3.4.5.6.7.8.9]. These works make querying data from XML and relational database easier than before by using a flexible interface into which user can input keywords. And corresponding systems automatically return top-k relevant results to the user. Moreover, novel methods have been proposed to make the result more precise according to the user's intentions.

XML has emerged as an easy-exchanging data format. More and more companies have chosen XML as an exchange and storage format. However because there are also some data which must be stored in a relational database due to its advantages: well-defined query interface, low redundancy, data consistency, etc, so when facing these two kinds of storage formats, the issue of searching information over XML and relation co-occurrence (*XRC*) circumstance remains a big challenge (See example XML documents and relational data in Figure 1 and Figure 2).

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 156–165, 2010.

In the traditional manner of querying over *XRC*, a user not only has to master the query language like SQL and XQuery, (s)he also needs to learn the underlying structure of data from relation and XML. To release users from these needless burdens, intuitively, introducing keyword search in *XRC* circumstance is an ideal solution. Despite the existing study on XML keyword search or on relational one, they have not considered the particularity of *XRC*, so the technique they developed cannot be easily adapted to a mixed XML and relation circumstance.

In a setting where data is stored in both forms of XML and relation, the result of a keyword search is not simply a tuple, or a fragment of XML. Moreover, it is not always a connection tree of several tuples that collectively cover the keyword or a SLCA or MLCA semantic fragment of XML. In such condition, a result can be a connected subgraph composed of tuples from relation and elements from XML together that collectively satisfy the query. Therefore, the key issue to solve the problem in *XRC* circumstance is to develop a method to combine data from different sources together to form a result of the keyword search. Note that a relation tuple is not naturally related to an XML element or XML fragment. One way appeals to use semantic relation between XML and relation to connect XML and relation together for answering a query.

As far as we know, very few studies have targeted to design an ideal method to solve the above problems in *XRC*. In this paper, we propose *XRCJ* which represents the XML-relation-circumstance-join. In an *XRC* situation where data is stored in form of both XML and relation, one main challenge is how to combine the data from different sources together. To solve this challenge, we first model XML and relation as graph and then use *XRCJ* to connect data from XML and relation offline. Finally we create a novel inverted index for connected XML-relation graph to fulfill keyword search in *XRC* circumstance.

The rest of the paper is organized as follows. Section 2 describes related works which mainly concern how to integrate keyword search into XML or relation and how to make it work well. Section 3 describes the query model and result definition in a *XRC* circumstance. We propose a new operation that is used to link the relation and XML meaningfully. Section 4 describes more details about the operation and proposes an advanced inverted index that fulfills the implement of query execution. Section 5 describes experiment we conducted on synthetic data from real dataset DBLP and section 6 makes a conclusion of this paper.

2 Related Work

A plethora of works have focused on integrating keyword search on relational and XML database. DBXploer[1], DISCOVER-I[7], DISCOVER-II[6], BANKS-I[2] are systems built on top of relational database. DISCOVER and DBXploer use tuple trees as query result of keyword search. BANKS identifies query result using approximation of Steiner tree problem. In XML part, subtrees rooted at LCAs have been proposed as query result. As an extension, Smallest LCA[12], XSeek [13] have been proposed to make result of KS query more specific and accurate. SLCA can avoid false positives compared to LCAs at the expense of false negatives. XSeek [13] aims at returning nodes which are explicitly inferred from keywords or constructed according to entities in the data. XRANK[4] and XSEarch[3] enable keyword search over XML documents. XRANK [4] presents a ranking method when returning query

result, while XSEarch [3] focuses on semantics and ranking of results. Unlike all the above existing works, this paper mainly study keyword search on the circumstance of the combination of XML and relational data storage.

3 Query Model and Result Definition

In this section, we briefly describe the query model and then define search results for the keyword search in a XRC situation. Then we introduce a novel operation called XRCJ to implement keyword search.

3.1 Query Model

The reason why the query in setting of *XRC* is a unique one compared to the keyword search which previous works have studied is that it requires the combination of XML and relational data to form one query result. Studies focusing on the XML or relation have put their emphasis on whether the result is relevant enough to the input keyword and whether the result fits the user's intention enough. In context of *XRC*, things are more complex because of the involvement of both XML and relation. In the real world , relationship between data from XML and relation can be classified into three situations, 1) data from XML and relation has no overlap, which means that we cannot acquire any semantic relation between them, 2) data from them has overlapping to an certain extent, which means that we can discover semantic relationship using overlap information, 3) data from XML and relation has so many duplications that only considering data from XML or relation alone can answer user's query properly. Facing these complex issues, one naïve solution to integrate KS over data from both XML and relation is to use technique already proposed to process the keyword search on XML and relation separately, and combine the result of the two queries as a united one. However these methods cannot address the redundancy of the result accurately. Meanwhile, due to the uncertainty of the semantic relationship between the data from XML and relation, simply processing keyword search over data from XML and relation separately cannot capture the relationship between them and therefore fail to produce result that satisfy user's query properly. Because it is very likely that only tuples from relation and elements from XML together can cover all keywords (assuming the query semantic is *conjunctive*) while tuples derived from XML or elements from XML alone cannot answer the query, so the combination of the results from XML and relation cannot answer the query properly too.

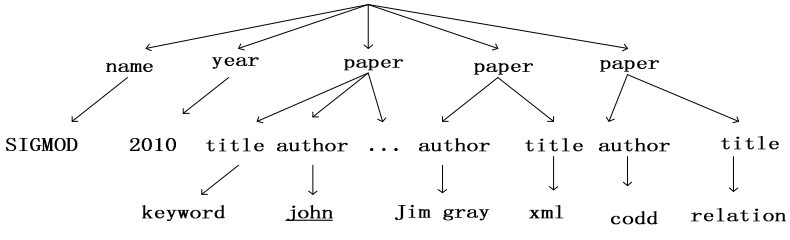


Fig. 1. An example XML document

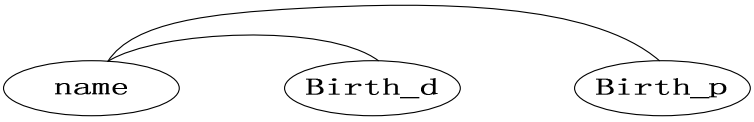


Fig. 2. An example schema of a relational database

Tables 1~3. Three example simple relational tables

Id	Name
..0	John
..1	John

Id	Time
..0	1988-2-22
..1	1978-6-13

Id	Place
..0	Hong kong
..1	Shanghai

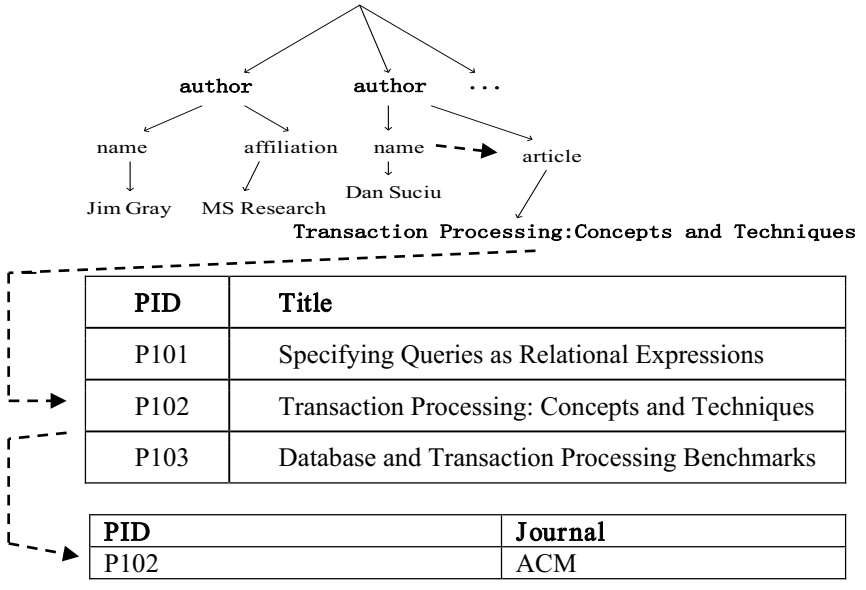
Based on the above discussion, we propose a novel query model within the *XRC* condition. We first define the input of keyword search as: $Q=(K_1,K_2,..K_q)$, where k_i denotes a keyword that users input. The data graph which stored in the form of both XML and relation that potentially provides query results are defined as search space $S=(n_1,n_2,n_3..n_s)$ where n_i denotes the basic unit in the search space S. In XML, n_i may represent an element which only has a text value child, while in relation, n_i may represent a tuple in tables. Obviously the search space is a directed graph with n_i being vertex and references between n_i being edges. In XML databases, reference may be a containment edge which links a parent node and its child node, or id-reference between elements. In relation databases, reference may be a link from a foreign key to a primary key. In the search space, nodes can be simply classified into two parts: one part is from XML and another is from relation. At the beginning, there is no edge between these two parts as data node from XML and relation will not refers to each other in a structure way naturally.

Secondly, we define a function T(S), which receives a search space as input then outputs a more connected one with an operation called *XRCJ* which will be discussed in following section. Thirdly function F accepts the query Q and a refined search space S, and then return a query result RS.

3.2 Result Definition

In this subsection, we will analyze the exact definition of query result in *XRC* , we also will introduce an operation *XRCJ* used to extend the original search space to a more connected one by connecting data node from XML and relation through semantic relationship between them.

In a relational data graph, a query result is assembled tuples that cover all or part of the input keyword. In contrast, a query result upon XML data tree may be a subtree of the entire tree, and the result subtree may satisfy the semantic of SLCA or ELCA[9,12]. However in a *XRC* situation, due to the specificity of mixed existence of data from XML and relation, neither assembled tuples or subtrees works well. Therefore we proposed a novel result format: mixed tuple element subgraph(MTES). Formally, MTES has two components, one is XML part and another is relation part, two parts are connected through semantic relationship between them (explore the overlap



PID	Title
P101	Specifying Queries as Relational Expressions
P102	Transaction Processing: Concepts and Techniques
P103	Database and Transaction Processing Benchmarks

PID	Journal
P102	ACM

Fig. 3. Example of connection between data from XML and relation

information of them). It is requested that result must be so specific that would not overwhelming user when returned to user, meantime it must contain enough information a user could need.

To meet these requirements, we first put forward the notion of information unit. The intuition behind it is that when a user issue keyword search, (s)he just wants the relevant information which is related to his input keyword to be returned. That is to say information that is not relevant will not arouse user's interest. What a user wants to get from a keyword search system can be classified into two types: one is those information in database that cover the keywords he just inputs, we called them complete or detail information, the other is those information that has linkage relationship with complete or detail information the user input, we called extensive information. Due to the property of relational database that information is split when doing normalization, we define tuples connected through foreign-primary relationship as information unit of relation part in query result in *XRC***.** In order to make the information unit of relation part to the minimal extent while containing enough information, size of relation part (number of nodes) of information unit is defined no larger than that of schema graph. Based on the same consideration , as only leaf nodes in a XML subtree contain actual information and sibling leaf nodes together make up a whole information unit (nodes with id-reference relation are considered as siblings), we define sibling leaf nodes of one parent as information unit of XML part of result in *XRC***.** In order to return enough information and make the result as specific as possible, the information unit in MTES is defined by following three rules formally:

(1) Nodes of MTES represent tuple from relation, or element that only has one text value child from XML.

(2) Edges of MTES represent foreign-primary relationship from relation, or sibling (including id-reference) relationship from XML, and semantic relationship between relation and XML.
(3) Size of relational part of MTES is constrained by the schema of relation.

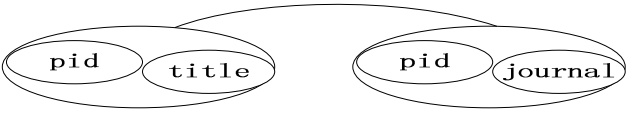
The relation constraints are based on the intuition that schema of one relational database represents the basic information relationship between tuples. So tuples related according to schema can form a minimal information unit for a result. The XML part's constraints is based on the intuition that sibling element that only has a text child can contribute to their parent's information as a whole. Therefore they form a minimal information unit in XML database. In Figure 3, an example of MTES can be described as: name->article->Title->Journal.

4 Connecting XML and Relation

In order to enable the challenging query in *XRC* situation, we use the operation *XRCJ* to join data from XML and relation together. Following subsections mainly concern detail information about how *XRCJ* works. Then we will introduce novel inverted index which implements *XRCJ*.

4.1 Connection Method

To capture the semantic relationship between data from XML and relation, one method is to scan all relational tuples (or XML elements) and for each tuple(element), scan all the corresponding XML elements (relation tuples), check whether there are elements in XML(tuples in relation) that contain same information as corresponding tuple(element) contains. If there is such relation-XML pair that contains same information, we say they have semantic relationship between them. However this algorithm requires huge number of scan operations and comparisons which would cost unnecessary computing resources. Assuming that there are n tuples in relation database, m elements which only has text value child in XML database, the complexity of scan operation and comparisons would be O(n*m). To improve efficiency of *XRCJ* and reduce this huge number of computing cost**,** we propose a refined algorithm which uses schema of relation and the DTD of XML.



<!ELEMENT Authors (author|place…)*>
<!ELEMENT authors(article|name|affiliation|…)*>
<!ENTITY %field "article|name|affiliation">

Fig. 4. Example of schema of one relational database and schema of one XML document

To avoid scanning all of data from XML and relation when connecting data from two sources together, we explore information stored in schema. Using schema information, we can easily find out whether there are semantic relationships between data from XML and relation, if exist, which part of data from XML and relation may have semantic relationship can be easily identified too. For an example, by analyzing the example schema in Figure 4, we can infer that the attribute "title" from relation and "article" from XML element "author" could possibly have overlap information between them. Therefore we can reduce both the search range and searching cost. If there is no semantic relationship between data from XML and relation at all, we have to use naïve method to implement keyword search in circumstance of *XRC*.

After we have located those data from XML and relation that have semantic relationship, however, to link them together and record it for generation of ultimate result, there is still a challenge in front of us. According to the unique property of semantic-relation-attribute, we first classify them into two kind: 1) the value of semantic-relation-attribute is unique, 2) The value of semantic-relation-attribute has repetition.

Consider the first kind, in order to record the semantic linkage, we construct a virtual node in refined search space after identifying semantic relationship between data node. The virtual node here is stored in relation as a table named semantic relation table with three attribute: first attribute is the id of semantic-relation-attribute$^{!}$ of relation, second attribute$^{\#}$ is the id of semantic-relation-attribute of XML. The third attribute is a boolean value that indentify the validation of the semantic relationship current tuple represents. The third attribute here is set to 1.

For the second kind, due to the existence of repetition of value of semantic-relation-attribute, simply connecting semantic-relation-attribute from XML and relation together would create false semantic relationship .For example, considering the situation in Figure 1, there are two name value "john" in relation ,however in XML part, there is only one author named "john" . By using method mentioned above, "john" from XML could connect with two authors named" john" from relation , although one of the authors named "john" in relation did not write the article recorded in XML ,it is against fact!

As things are, when we confront the second kind of condition, we set the third attribute of the current tuple of semantic relation table to 0. Thus when we use the semantic relation table to connect XML and relation index together, we first check the validation of semantic relationship represented by current tuple. If it is positive, we connect them using algorithm discussed below .If the validation is negative, which means there exist ambiguous semantic relationship between data from XML and relation , we left XML and relation as original one. Detailed operations when facing negative validation of semantic relationship will be discussed in following subsections.

4.2 Indexing

Next we will discuss mixed inverted index in more detail from practical aspect. Follow the inverted index style information retrieval, we propose an advanced inverted index called mixed inverted index.

According to the definition of the query result, mixed inverted index is composed of two parts: one is relation inverted index part, another is XML inverted index part, these two part are connected through semantic relationship between them. In relation

part, inverted index is a sequence of tuple id with connector "," which represents a foreign-primary key relationship between adjacent ids. Each tuple id represents the location of a tuple in corresponding table in relational database, and also represents information stored in corresponding tuple. In XML part, inverted index is also a sequence of element id with connector",". Here element id is the identification number of the element in whole XML database. We can use existing labeling schema to present the id as a dewey code does. Each adjacent element id represents the sibling relationship between them. Besides, element represented here must have and only have a text value children.

The core method of constructing mixed inverted index is: first, build an index table where each tuple is a sequence of id, and this id sequence represent information unit and query result in *XRC* .Second, using this index, traverse information unit represent by those ids and parse sentences in that information unit , record terms and id sequence. Then after the traversal and parse work is done, the table record term and id sequence is created, that is the mixed inverted index.

Using schema of relation, we can easily extract information unit in relation by connecting tuples that has foreign-primary key relationship, then we take down the ids of the tuples. In the same way we can take down the ids of elements in XML through its schema in dtd files. After that using semantic relation table created by the *XRCJ* operation, we can connect the XML and relation part as one index. It should be noted that as the existence of ambiguous semantic relationship between data from XML and relation, on occasion, id sequence from XML and relation cannot be connected through semantic relation table. In this case , when we find the validation of semantic relationship is negative by checking corresponding attribute in semantic relation table, we connect id sequence of XML(or relation) with"-1,-1" to achieve formal conformance ,which means that id sequence of XML (relation)part alone make a ultimate index. An example inverted index will be:" 0,56,70,0.34.1.0,0.34.8.1.0".

5 Experiment

We have designed and performed experiments to evaluate the *XRCJ* operation's performance. In this section we describe the experiments and present the result.

5.1 Experiment Setup

We run experiments on synthetic data, thus data are generated from real-world XML data set from DBLP. As the limitation of space we only report results which present executing time via the growing size of experiment data when doing *XRCJ* operation. All experiment are performed on the Intel(R) Core(TM)2 Duo CPU E6750 2.66GHz PC with 2.00GB RAM 250G hard disk, running windows XP with Java language.

5.2 Experiment Data

The original data file from data set of DBLP is at size of 20M. We generate different size of relation tables and XML documents from raw XML data file. The following table shows the size of tables in relational data and number of nodes in XML data.

Table 4. Different sizes of relation tables and different number of nodes in XML document

File	Size of relation(MBytes)	Num-of-node-in-XML
1	1.11	847
2	2.39	1671
3	4.59	3103
4	9.92	6303
5	19.2	9779

5.3 Experimental Performance

Finally we describe the performance of our *XRCJ* operation. With the growing size of relation tables, the elapsed time has been shown in Figure 5 which is obtained by fixing number of nodes in XML to 4728 and varying the size of table which is used to be connected with corresponding XML document. The results show that our proposed algorithm is scalable well and robust with varied data size.

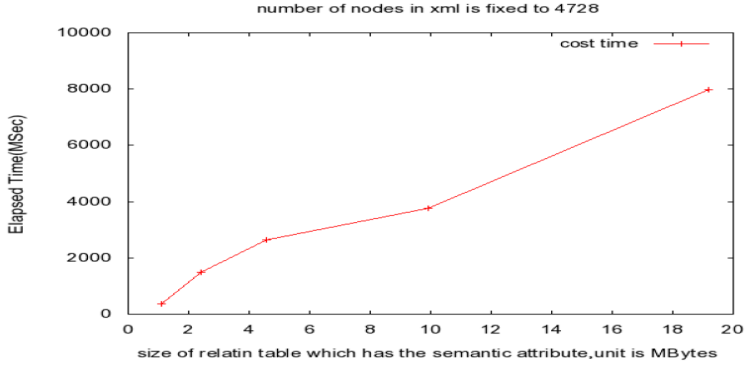


Fig. 5. Performance of the algorithm with varied size of data

6 Conclusion

In this paper, we have analyzed a new circumstance where data stored in both form of XML and relation. In order to enable user to search information in a more convenient manner within this circumstance, we propose a novel format of query result which consists of data from XML and relation. We propose our method to clear the way and a new operation which can connect data from XML and relation together. From a practical aspect, we designed our novel inverted index to support our operation and gave an efficient method to create this novel inverted index. Finally an experimental study was performed and verified the scalability of our approach.

Acknowledgements

This paper was partially supported by 863 National High-Tech Research Plan of China (No: 2009AA01Z133, 2009AA01Z149), National Science Foundation of China

(NSFC) (No.60903056), Key Project in Ministry of Education (No: 109004) and SRFDP Fund for theDoctoral Program (No.20090004120002).

References

1. Agrawal, S., Chaudhuri, S., Das, G.: Dbxplorer: A system for keyword-based search over relational databases. In: ICDE, pp. 5–16 (2002)
2. Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., Sudarshan, S.: Keyword searching and browsing in databases using banks. In: ICDE, pp. 431–440 (2002)
3. Cohen, S., Mamou, J., Kanza, Y., Sagiv, Y.: Xsearch: A semantic search engine for XML. In: VLDB (2003)
4. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: Xrank: Ranked keyword search over XML documents. In: SIGMOD, pp. 16–27 (2003)
5. He, H., Wang, H., Yang, J., Yu, P.: Blinks: Ranked keyword searches on graphs. In: SIGMOD (2007)
6. Hristidis, V., Gravano, L., Papakonstantinou, Y.: Effcient ir-style keyword search over relational databases. In: VLDB, pp. 850–861 (2003)
7. Hristidis, V., Papakonstantinou, Y.: Discover: Keyword search in relational databases. In: VLDB (2002)
8. Hristidis, V., Papakonstantinou, Y., Balmin, A.: Keyword proximity search on XML graphs. In: ICDE, pp. 367–378 (2003)
9. Li, G., Feng, J., Wang, J., Zhou, L.: Effcient Keyword Search for Valuable LCAs over XML Documents. In: CIKM (2007)
10. Luo, Y., Lin, X., Wang, W., Zhou, X.: Spark: Top-k keyword query in relational databases. In: SIGMOD (2007)
11. Markowetz, A., Yang, Y., Papadias, D.: Keyword search on relational data streams. In: SIGMOD (2007)
12. Xu, Y., Papakonstantinou, Y.: Effcient keyword search for smallest lcas in XML databases. In: SIGMOD, pp. 527–538 (2005)
13. Liu, Z., Chen, Y.: Identifying return information for XML keyword search. In: SIGMOD (2007)

Formal Verification of Stochastic Timing Behavior in Web-Based Business Process Collaboration*

Haiyang Hu[1,2,3], Jianen Xie[1], and JiDong Ge[2]

[1] College of Computer Science, Hangzhou Dianzi University,
Hangzhou, China 310018
huhaiyang@hdu.edu.cn
[2] State Key Laboratory for Novel Software Technology, Nanjing University,
Nanjing, China 210089
[3] College of Computer Science and Information Engineering,
Zhejiang Gongshang University, Zhejiang, China 310018

Abstract. Web services-based systems have gained the attention of researchers, which enable business process system to automatically discovery and invocation suitable Web services at run time. In this paper, for stochastic- timing software business process collaboration based on Web services, we take into account the problem of verifying the behavioral compatibility in the cross-organizational system. We use the extended markovian process algebra to model the behavior of the business processes, and address the issue of formally verification of whether the behavior of system design can meet the requirements of behavioral compatibility from the aspect of stochastic timing. Based on analyzing the bisimulation relation of the parallelism of business processes, we present a several theorems for check the behavioral compatibility in the collaboration system.

1 Introduction

Recent years, the technology of Business Process Management has been focused on by software scientists and engineers. It can help enterprise managers to design, maintain, and improve large enterprise systems on the basis of business process models [1]. Usually, a business process model is a flow-oriented representation by coordinating a set of activity, which aims at achieving certain goals, such as functions and performance. Examples of software business processes include design methods, change-control procedures, and testing strategies.

With the scale of organization becoming large, more business processes are carried out across different organizations distributed globally. The challenges of managing and improving this cross-organizational business process collaboration have come to the forefront of software researchers. In response, new methods and tools have been

* This paper is supported by National Natural Science Foundation of China granted by Nos. 60873022 and Nos. 60903053, the open fund provided by State Key Laboratory for Novel Software Technology of Nanjing University, and the Natural Science Foundation of Zhejiang Province of China under Grant No.Y1080148.

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 166–176, 2010.

presented to support the business process collaboration from various aspects [8][9][10]. In these methods, the web-based business process system has gained the attention of researchers. In developing the Web-based business process system, there are four stages [11]: business process selection, Web service discovery and match-making, Web service provisioning and business process system invocation. During the design such a system, one of the major goals in developing such systems is the verification that whether the behavior of business process collaboration satisfy the specified stochastic timing requirements.

In this paper, for time-constraint business process collaboration, we use the exponentially timed kernel of process algebra $EMPA_{et}$ (Extended Markovian Process Algebra) [2] [8] to address the issue of formally verification of whether the behavior of system design can meet the requirements of behavioral compatibility from the aspect of stochastic timing. The rest of the paper is organized as follows: in Section 2, definition and operation semantic of EMPA_{et} are given. In Section 3, we use $EMPA_{et}$ to model the stochastic timing behavior of business processes. In Section 4, we put analysis on the stochastic-timing collaboration of business process, which is specified by a set of synchronization on the shared activities; a set of verification rules is presented as well. An example for specifying the verification process is presented in Section 5. In section 6, the conclusion of this paper is given.

2 Definition of Extended Markovian Process Algebra

In this paper, we use the process algebra EMPA_{et} [2] [8] to specify the stochastic-timing behavior of a business process, which can effectively model the synchronization of activities among business process, capture their similarity and deduce the behavioral compatibility of the collaboration process. Definition of EMPA_{et} is given as follows.

Denote the set of actions by $Act = AType \times ARate$ where $AType$ is the set of types including τ (the internal action which can not be observed by external observers) and $ARate = \Re_+ \cup Inf \cup \{*\}$, with $Inf = \{\infty_{l,w} \mid l \in \mathrm{N}_+ \wedge w \in \Re_+\}$, is the set of rates. In this paper, we use $a, b, ...$ to denote the variables in $AType$, and λ, μ for $ARate$. Now, we present the definition of labeled transition system (LTS), which is adopted in the definition of the operation semantics for EMPA_{et}.

Definition 1. A labeled transition system (LTS) is a tuple:

$$(S, U, \longrightarrow, s_0)$$

where S is a set of states, U is a set of labels, $\longrightarrow \subseteq S \times U \times S$ is called transition relation, and $s_0 \in S$ is called the initial state.

Definition 2. The process terms of exponentially timed kernel EMPA_{et} is generated by the following syntax:

$$E ::= \underline{0} \mid < a, \overline{\lambda} > .E \mid E / L \mid E[\varphi] \mid E + E \mid E \parallel_S E \mid A$$

where the null term "$\underline{0}$" is the term that can not execute any actions; the term "$< a, \overline{\lambda} > .E$" represents that it can execute an action a with the rate $\overline{\lambda}$ and then

behaves as term E; the term E/L represents that any action $a \in L$ executed in E is turned into τ; the term $E[\varphi]$ behaves as term E except that the action a is replaced by $\varphi(a)$; the term $E_1 + E_2$ behaves as either term E_1 or E_2 depending on the action of E_1 or E_2 being executed first; the term $E_1 \|_S E_2$ represents that for those actions of E_1 and E_2 not in S, $E_1 \|_S E_2$ asynchronously execute them, while for any action in actions S, $E_1 \|_S E_2$ synchronously executes it.

Note that in EMPA_{et} the synchronization of E_1 executing a given action a by rate λ with E_2 executing a by rate μ can be completed if and only if $\min(\lambda, \mu) = *$. The operation semantics of EMPA_{et} are given in the following table.

$$\frac{(<a,\lambda>,E') \in Melt(Select(PM(E)))}{E \xrightarrow{a,\lambda} E'}$$

$$\begin{aligned}
PM(\underline{0}) &= \phi \\
PM(<a,\lambda>.E) &= \{|\,(<a,\lambda>,E)\,|\} \\
PM(E/L) &= \{|\,(<a,\lambda>,E'/L)\,|\,(<a,\lambda>,E') \in PM(E) \wedge a \notin L\,|\} \oplus \\
&\quad \{|\,(<\tau,\lambda>,E'/L)\,|\,\exists a \in L.(<a,\lambda>,E') \in PM(E)\,|\} \\
PM(E[\varphi]) &= \{|\,(<\varphi(a),\lambda>,E'[\varphi])\,|\,(<a,\lambda>,E') \in PM(E)\,|\} \\
PM(E_1 + E_2) &= PM(E_1) \oplus PM(E_2) \\
PM(E_1 \|_S E_2) &= \{|\,(<a,\lambda>,E_1' \|_S E_2)\,|\,a \notin S \wedge (<a,\lambda>,E_1') \in PM(E_1)\,|\} \oplus \\
&\quad \{|\,(<a,\lambda>,E_1 \|_S E_2')\,|\,a \notin S \wedge (<a,\lambda>,E_2') \in PM(E_2)\,|\} \oplus \\
&\quad \{|\,(<a,\gamma>,E_1' \|_S E_2')\,|\,a \in S \wedge \exists \lambda_1,\lambda_2 \\
&\quad (<a,\lambda_1>,E_1') \in PM(E_1) \wedge (<a,\lambda_2>,E_2') \in PM(E_2) \wedge \\
&\quad \gamma = Norm(a,\lambda_1,\lambda_2,PM(E_1),PM(E_2))\,|\} \\
PM(A) &= \text{if } A \stackrel{\Delta}{=} E \\
Norm(a,\lambda_1,\lambda_2,PM_1,PM_2) &= Split(\lambda_1, 1/(\pi_1(PM_2))(<a,*>)) \text{, if } \lambda_2 = * \\
&\quad \text{or } Split(\lambda_2, 1/(\pi_1(PM_1))(<a,*>)) \text{, if } \lambda_1 = *
\end{aligned}$$

$$Split(*,p) = * \qquad Split(\lambda,p) = * \quad Split(\infty_{l,w},p) = \infty_{l,w \cdot p}$$

In this table, PM denotes the set of all potential moves.

Definition 3. A partial function $Rate : \varsigma \times AType \times APLev \times \mathrm{P}(\varsigma) \longrightarrow ARate$ can be defined by

$$Rate(E,a,l,C) = Aggr\{|\,\lambda\,|\,\exists E' \in C.E \xrightarrow{a,\lambda} E' \wedge PL(<a,\lambda>) = l\,|\}$$

Definition 4. An equivalence relation $B \subseteq \varsigma \times \varsigma$ is a strong extended Markovian bisimulation (strong EMB) if and only if, whenever $(E_1,E_2) \in B$, then for all $a \in AType$, $l \in APLev$, and $C \in \varsigma / B$

$$Rate(E_1,a,l,C) = Rate(E_2,a,l,C)$$

Definition 5. Let $\sim_{EMB}$ be the union of all the strong EMBs. Then $\sim_{EMB}$ is the largest strong EMB. We say that $E_1, E_2 \in \varsigma$ are strongly extended Markovian bisimulation equivalent if and only if $E_1 \sim_{EMB} E_2$.

To eliminate the nodeterminism from $EMPA_{et}$ so as to ensure that $\sim_{EMB}$ is a congruence, it holds that: i) if $\exists a \in S, E_1 \xrightarrow{a,*} E_1' \wedge E_1 \xrightarrow{a,*} E_1'' \wedge E_2 \xrightarrow{a,\lambda} E_2' \wedge \lambda \neq *$, then it must holds $E_1' \sim_{EMB} E_1''$; and ii) if $\exists a \in S, E_1 \xrightarrow{a,\lambda} E_1' \wedge \lambda \neq * \wedge E_2 \xrightarrow{a,*} E_2' \wedge E_2 \xrightarrow{a,*} E_2''$, then it must holds $E_2' \sim_{EMB} E_2''$.

Deadlock in the process reflects that a process is in a blocked state where the process is not terminated successfully, and it can't continue to execute any observable action. The deadlock-free process can be formally defined as following:

Definition 6. A term E is said to be deadlock free to an external observer, if and only if, for $E \neq \underline{0}$, there exist no such an observable action a that $E \xrightarrow{a}$.

3 System Model

In a cross-organizational business process system, the business processes collaborates with each other through some certain interfaces which include a set of shared activities. We present a formal model for the business process under collaboration as follows:

Definition 7. A business process W based on Web services is defined by such a tuple:

$$W =< I_W, A_W, C_W, P_W >$$

where:

- I_W contains a set of interfaces through which W collaborates with other business processes. For each $ite \in I_W$, it contains a set of activities shared by W and some other business process;
- $A_W = Act_W \times ARate$ is the set of activities executed in W, where Act_W is the set of activities including τ and $ARate = R_+ \cup \{*\}$ is the set of rates at which activities are executed. A_W includes two disjoint parts, external activities and internal activities, which are denoted by A_W^E and A_W^I respectively.
- C_W is a set of collaborations between W and other business processes. Each collaboration $con_i \in C_W$ expresses a collaboration of W and another business process W_i, and it is defined by a tuple $con_i =< ite_i, ins_i, pl_i >$ where ite_i is one interface of W, ins_i is the instance of W_i, and pl_i is the location of W_i.
- The stochastic-timing behavior of W is denoted by P_W, which is in the form of . $EMPA_{et}$ and defined by a tuple $< S_W, A_W, \Gamma_W >$, where S_W is a finite set of states, and $\Gamma_W \subseteq S_W \times A_W \times S_W$ is a finite set of transitions between states, such that $\Gamma_W = \{ (s,(a,\mu),s') \mid s,s' \in S_W, a \in A_W \wedge s \xrightarrow{a,\mu} s' \wedge \mu = \lambda, * \}$.

Suppose now two business processes $W_i =< I_{W_i}, A_{W_i}, C_{W_i}, P_{W_i} >$ and $W_j =< I_{W_j}, A_{W_j}, C_{W_j}, P_{W_j} >$ collaborate with each other through an interface $ite_{(W_i,W_j)}$, then their collaboration behavior can be specified by $P_{W_i} \|_{ite_{(W_i,W_j)}} P_{W_j}$. During the running of the system, W_i and W_j synchronously execute the actions in $ite_{(W_i,W_j)}$. So, we can present the notion of behavioral compatibility for business process collaboration as following:

Definition 8. For two business processes $W_i =< I_{W_i}, A_{W_i}, C_{W_i}, P_{W_i} >$ and $W_j =< I_{W_j}, A_{W_j}, C_{W_j}, P_{W_j} >$ collaborate with each other through an interface $ite_{(W_i,W_j)}$, W_i and W_j are

behavioral compatibility, if and only if, there exists such a set of actions $a_1, a_2, ..., a_n \in A^E_{W_i} \cup A^E_{W_j}$, that it holds that $P_{W_i} \|_{ite_{(W_i,W_j)}} P_{W_j} \xrightarrow{a_1,\lambda_1} ... \xrightarrow{a_n,\lambda_n} \underline{0} \| \underline{0}$.

Now, the definition of cross-organizational business process system which may be constituted by multiply business processes, can be represented as follows:

Definition 9. A cross-organizational business process system, MWS, is a tuple

$$< V_M, E_M, I_M, T_M, P_M >$$

Where

- $V_M = \{W_1, W_2, ... W_n\}$ is a set of software business processes contained in MWS;
- E_M is the set of interfaces through which the business processes collaborate with each other;
- I_M is the set of interfaces provided by MWS for its external environment;
- T_M is a set of all collaboration relations in MWS, $T_M = \bigcup_{1 \le i \le n} C_i$;
- P_M is the behavior of MWS, which is defined by $P_M = P_{W_1} \|_{ite_{(W_1,W_2)}} P_{W_2} \|_{ite_{(W_1,W_3)} \cup ite_{(W_2,W_3)}} P_{W_3} \| ... \|_{ite_{(W_1,W_3)} \cup ... \cup ite_{(W_{n-1},W_n)}} P_{W_n}$.

A state s_M of P_M is defined in the domain $S_{W_1} \times S_{W_2} \times ... \times S_{W_n}$, $1 \le i \le n$. Then, the state transition of P_M can be given by:

Definition 10. Suppose s_M and s_M' are two different states of P_M, with $s_M = (s_{W_1}, ..., s_{W_n})$ and $s_M' = (s_{W_1}', ..., s_{W_n}')$. When one of the two following conditions are satisfied, P_M transits from s_M to s_M' by executing an action a at rate λ, i.e., $s_M \xrightarrow{a,\lambda} s_M'$:

- there exist an activity $(a,\lambda) \in A^I_{W_i}$ and a state transition $(s_{M_i}, (a,\lambda), s_{M_i}') \in \Gamma_{W_i}$; while for all other W_j $(1 \le j \le n, j \ne i)$, it holds $s_{W_j} = s_{W_j}'$;
- there exist an interface $ite_{(W_i,W_j)}$, an action $a \in ite_{(W_i,W_j)}$, and two state transitions $(s_{W_i}, (a,\mu), s_{W_i}') \in \Gamma_{W_i}$ and $(s_{W_j}, (a,\gamma), s_{W_j}') \in \Gamma_{W_j}$, where $\mu \in R_+ \wedge \gamma = *$ or $\mu = * \wedge \gamma \in R_+$, and at the same time, for all other $W_k (1 \le k \le n, k \ne i, j)$, it holds that $s_{W_k} = s_{W_k}'$.

The deadlock-free behavior of P_M, which specifies that the synchronously executing activities through the interfaces will not make any one of the business process enter into a dead-lock state. In this way, the deadlock-free behavior of a collaboration system can be formally defined by:

Definition 11. For a MWS constituted by n business processes, $\{W_1, ..., W_n\}$, its behavior P_M is deadlock-free, if and only if, there exists a set of activities $a_1, ..., a_m \in \bigcup_i A^E_{W_i}$ and it holds that $P_M \xrightarrow{a_1,\lambda_1} ... \xrightarrow{a_m,\lambda_m} \underline{0} \| ... \| \underline{0}$.

4 Stochastic-Timing Behavioral Compatibility

Suppose a business process W_i collaborates with W_j through an interface $ite_{(W_i,W_j)}$. To preserve behavioral compatibility, W_i only focuses the behavior of W_j shown on $ite_{(W_i,W_j)}$. The same to W_j, it only cares for whether the corresponding collaborating behavior of W_i shown on $ite_{(W_i,W_j)}$ meets its requirements. So, if W_i can meet the requirements of W_j, and simultaneously, the behavior of W_j also meets the requirements needed by W_j, then they will collaborate with each other successfully.

Theorem 1. Suppose two business processes $W_i =< I_{W_i}, A_{W_i}, C_{W_i}, P_{W_i} >$ and $W_j =< I_{W_j}, A_{W_j}, C_{W_j}, P_{W_j} >$ collaborate with each other through an interface $ite_{(W_i,W_j)}$. Their behavior of collaboration is compatible, if and only if, there exists a relation $B \subseteq S_{W_i} \times S_{W_j}$, such that, whenever $(P_{W_i}, P_{W_j}) \in B$, for any activity $a \in ite_{(W_i,W_j)}$, then

- If $P_{W_i} / D1 \xrightarrow{\tau^m} \xrightarrow{a,\mu} \xrightarrow{\tau^n} P_{W_i}' / D1$, then $(P_{W_i} \|_{ite_{(W_i,W_j)}} P_{W_j}) / D2 \xrightarrow{\tau^u} \xrightarrow{a,\gamma} \xrightarrow{\tau^v} (P_{W_i}' \|_{ite_{(W_i,W_j)}} P_{W_j}') / D2 \wedge \gamma = \mu \in R_+ \wedge (P_{W_i}', P_{W_j}') \in B$, where $D1 = A_{W_i} \setminus ite_{(W_i,W_j)}$, $D2 = (A_{W_i} \cup A_{W_j}) \setminus ite_{(W_i,W_j)}$, $m,n,u,v \in N_+ \cup \{0\}$;
- If $P_{W_j} / D3 \xrightarrow{\tau^m} \xrightarrow{a,\mu} \xrightarrow{\tau^n} P_{W_j}' / D3$, then $(P_{W_i} \|_{ite_{(W_i,W_j)}} P_{W_j}) / D2 \xrightarrow{\tau^u} \xrightarrow{a,\gamma} \xrightarrow{\tau^v} (P_{W_i}' \|_{ite_{(W_i,W_j)}} P_{W_j}') / D2 \wedge \gamma = \mu \in R_+ \wedge (P_{W_i}', P_{W_j}') \in B$, where $D3 = A_{W_j} \setminus ite_{(W_i,W_j)}$

$m,n,u,v \in N_+ \cup \{0\}$; ■

The theorem 1 gives a condition of a partial order behavioral compatibility between two business processes collaboration. Under this scene, if each process can meet the requirements of the other one, they will collaborate with each other compatibly. The conditions presented in theorem 1 are very strict. During the collaboration process, if one business process only needs some activities provided by the other one through the interface, i.e., there is one process executing its activities in the interfaces in a passive manner throughout the process of collaboration, then, some looser conditions can be given, which are shown in theorem 2.

Theorem 2. Suppose two business processes $W_i =< I_{W_i}, A_{W_i}, C_{W_i}, P_{W_i} >$ and $W_j =< I_{W_j}, A_{W_j}, C_{W_j}, P_{W_j} >$ collaborate with each other through an interface $ite_{(W_i,W_j)}$. P_{W_j} is determinism. Their behavior of collaboration is compatible, if and only if, for any behavior trace of P_{W_i}, σ_1, if it holds that $\sigma_1 / ite_{(W_i,W_j)} \equiv \xrightarrow{a_1,\lambda_1} ... \xrightarrow{a_n,\lambda_n}$, where $\lambda_i \in R_+ \wedge a_i \in ite_{(W_i,W_j)}$, then there must exists a behavior trace of P_{W_j}, σ_2, and it holds that $\sigma_2 / ite_{(W_i,W_j)} \equiv \xrightarrow{\tau^m} \xrightarrow{a_1,*} \xrightarrow{\tau^u} ... \xrightarrow{a_n,*} \xrightarrow{\tau^v} \wedge m,n,u,v \in N_+ \cup \{0\}$ ■

Let $trace(P_{W_i})$ be the set of all possible traces executed by P_{W_i} with ignoring the influence of rate λ. From theorem 2, it is explicitly that if $trace(P_{W_i}) / ite_{(W_i,W_j)} \cap trace(P_{W_i}) / ite_{(W_i,W_j)} = \phi$. The collaboration behavior of P_{W_i} and P_{W_j} can not be compatible.

For a collaboration system composing multiply business processes, each process may interact with several other processes simultaneously. To preserving the behavioral compatibility in this system, we need to verify the behavior of collaboration among all the interfaces, and ensure that each process can avoid entering into its

deadlock state, i.e., it can accomplish all the activities successfully. Denoted by $P[i]$ the term $P_{W_1} \|_{ite_{(W_1,W_2)}} P_{W_2} \|... \|_{ite_{(W_1,W_{i-1})} \cup ... \cup ite_{(W_{i-2},W_{i-1})}} P_{W_{i-1}} \|_{ite_{(W_1,W_{i+1})} \cup ... \cup ite_{(W_{i-1},W_{i+1})}} P_{W_{i+1}} \|... \|_{ite_{(W_1,W_n)} \cup ... \cup ite_{(W_{n-1},W_n)}} P_{W_n}$, then $P_M \equiv P_{W_i} \|_{I_i} P[i]$, where $I_i = \bigcup_{1 \le k \le n, k \ne i} ite_{(W_k,W_i)}$. The rule for verifying the behavioral compatibility in the system is presented in theorem 3.

Theorem 3. Suppose now a *MWS* composing n business processes, $\{W_1,...,W_n\}$ ($W_i =< I_{W_i}, A_{W_i}, C_{W_i}, P_{W_i} >$). Its behavior P_M is compatible, if and only if, for each W_i, there exists a relation $B \subseteq S_{W_i} \times (S_{W_1} \times ... \times S_{W_{i-1}} \times ... \times S_{W_{i+1}} .. \times S_{W_n})$, such that, whenever $(P_{W_i}, P^i) \in B$, for any activity $a \in I_i$, then

- If $P_{W_i}/D4 \xrightarrow{\tau^m} \xrightarrow{a,\mu} \xrightarrow{\tau^n} P_{W_i}'/D4$, then $(P[i] \|_{I_i} P_{W_i})/D5 \xrightarrow{\tau^u} \xrightarrow{a,\gamma} \xrightarrow{\tau^v} (P[i]' \|_{I_i} P_{W_i}')/D5 \ \wedge \gamma = \mu \in R_+ \ \wedge (P_{W_i}', P[i]') \in B$, where $D4 = A_{W_i} \setminus I_i$, $D5 = \bigcup_{1 \le k \le n} A_{W_k} \setminus I_i$, $m,n,u,v \in N_+ \cup \{0\}$;
- If $(P[i] \|_{I_i} P_{W_i})/D5 \xrightarrow{\tau^u} \xrightarrow{a,\gamma} \xrightarrow{\tau^v} (P[i]' \|_{I_i} P_{W_i}')/D5$, then $P_{W_i}/D4 \xrightarrow{\tau^m} \xrightarrow{a,\mu} \xrightarrow{\tau^n} P_{W_i}'/D4 \ \wedge \gamma = \mu \in R_+ \ \wedge (P_{W_i}', P[i]') \in B$, where $m,n,u,v \in N_+ \cup \{0\}$; ■

We present an algorithm to check whether the behavior of two Web-based business process is compatible according to the above theorems.

```
Initializestack( stack);
while ( s_W1 != φ && s_W2 != φ ) do
{for each a ∈ AType && a ∈ Ite do
     for each l ∈ APLev do
      if ( there are s_W1' and s_W2' satisfying that s_W1 -a-> s_W1', s_W2 -a-> s_W2'
              and Rate(s_W1, a, μ, C) = Rate(s_W2, a, γ, C) ,where μ ∈ R+ ^ γ = * or
              μ = * ∧ γ ∈ R+ )
      then   { s_W1 = s_W1'; s_W2 = s_W2' };
      else
              if (notempty( stack) ) then < s_W1 , s_W2 > = pop( stack);
              else if ( empty( stack)) then return nocompatible;
  for each a ∈ AType && a ∉ Ite do
     for each l ∈ APLev do
        if ( there are s_W1' satisfying that s_W1 -a-> s_W1' )
       then { push(< s_W1 , s_W2 >, stack); s_W1 = s_W1'; };
  for each a ∈ AType && a ∉ Ite do
     for each l ∈ APLev do
        if ( there are s_W2' satisfying that s_W2 -a-> s_W2' )
       then { push(< s_W1 , s_W2 >, stack); s_W2 = s_W2';};
}
```

5 An Example

In this paper, we extend the e-commerce example presented in [12] to address the characteristics of the rules presented in this paper. In this example, people buy books on an e-commerce system which is composed by the collaboration of three Web service-based business processes, *BookShop* (W_{BS}), *BookBroker* (W_{BB}) and *Bank* (W_{BA}). In this system, the system runs as follows: firstly, *BookShop* registers at *BookBroker*. Next, when a person wants to buy some books, he calls the activity *getABook* provided by *BookBroker*. *BookBroker* then call the activity *inStock* to inquire *BookShop* whether the books needed by the person are in stock. If *BookShop* stores the books, *BookBroker* orders the book by calling the activity *Order*, makes *BookShop* deliver books to the person by executing the activity *deliver*, and puts the money of the books into *BookShop's* bank account by executing the activity *deposit* provided by *Bank*. Definitions of the three business processes and their interfaces are presented in Fig.1-5.

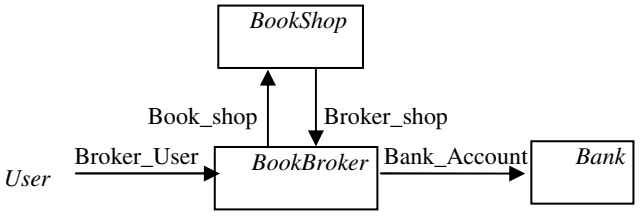


Fig. 1. An e-commerce instance

Shown in Fig.1, W_{BS} and W_{BA} only collaborate with W_{BB}. So, we can separately verify the behavioral compatibility of W_{BS} and W_{BB}, and W_{BA} and W_{BB} to ensure the behavioral compatibility in the whole system. For the collaboration of W_{BS} and W_{BB}, the interface through which they interact with each other is $ite_{(W_{BS},W_{BB})}$ =*Book_shop* ∪ *Broker_shop*={ *inStock, inStock_r, order, deliver, deliver_r* , *register*, *unregister*}. Based on theorem 1, it holds that whenever $P_{W_{BB}} / D1 \xrightarrow{\tau^m} \xrightarrow{a,\mu} \xrightarrow{\tau^n} P_{W_{BB}}' / D1$, then $(P_{W_{BS}} \|_{ite_{(W_{BS},W_{BB})}} P_{W_{BB}}) / D2 \xrightarrow{\tau^u} \xrightarrow{a,\gamma} \xrightarrow{\tau^v} (P_{W_{BS}}' \|_{ite_{(W_{BS},W_{BB})}} P_{W_{BB}}') / D2 \wedge \gamma = \mu$, where $a \in$ { *register, unregister, inStock, inStock_r, order, deliver, deliver_r* }, $D1$ ={ *login, logout, deposit, get balance, withdraw, withdraw_r, deposit_r* }, $D2$ = { *login, logout, deposit, deposit_r, getbalance, withdraw, withdraw_r*}, and at the same time, it holds that whenever $P_{W_{BS}} / D3 \xrightarrow{\tau^m} \xrightarrow{a,\mu} \xrightarrow{\tau^n} P_{W_{BS}}' / D3$, then $(P_{W_{BS}} \|_{ite_{(W_{BS},W_{BB})}} P_{W_{BB}}) / D2 \xrightarrow{\tau^u} \xrightarrow{a,\gamma} \xrightarrow{\tau^v} (P_{W_{BS}}' \|_{ite_{(W_{BS},W_{BB})}} P_{W_{BB}}') / D2 \wedge \gamma = \mu$, where $D3 = \phi$. For the collaboration of W_{BA} and W_{BB}, the interface through which they interact with each other is $ite_{(W_{BA},W_{BB})}$ = *BankAccount* ={ *login, getBalance, deposit, deposit_r, withdraw, withdraw_r, logout*}. Based on theorem 1, it holds that whenever $P_{W_{BB}} / D4 \xrightarrow{\tau^m} \xrightarrow{a,\mu} \xrightarrow{\tau^n} P_{W_{BB}}' / D4$, then $(P_{W_{BA}} \|_{ite_{(W_{BA},W_{BB})}} P_{W_{BB}}) / D5 \xrightarrow{\tau^u} \xrightarrow{a,\gamma} \xrightarrow{\tau^v} (P_{W_{BA}}' \|_{ite_{(W_{BA},W_{BB})}} P_{W_{BB}}') / D5 \wedge \gamma = \mu$, where $a \in$ { *login, getBalance, deposit, deposit_r, withdraw, withdraw_r, logout* }, $D4$ ={ *register, unregister, inStock, inStock_r, order, deliver, deliver_r* }, $D5$ = { *register, unregister, inStock, inStock_r, order, deliver, deliver_r*},

An interface provided by W_{BA} : *BankAccount*

interface BankAccount { *login, getBalance, deposit, deposit_r, withdraw withdarw_r logout*};

An interface provided by W_{BS} : *Book_Shop*

interface Book_Shop { *inStock, inStock_r, order, deliver, deliver_r* };

An interface provided by W_{BB} : *Broker_Shop*

interface Broker_Shop {*register, unregister*};

An interface provided by W_{BB} : *Broker_User*

interface Broker_User{ getABook};

Fig. 2. Interfaces provided by the business processes W_{BA}, W_{BS} and W_{BB}

BookShop: $W_{BS} =< I_{W_{BS}}, A_{W_{BS}}, C_{W_{BS}}, P_{W_{BS}} >$

$I_{W_{BS}}$: { *Book_Shop*, *Broker_Shop* };

$A_{W_{BS}}$: { <*register*, 1.5>, <*unregister*, 2.0>, <*inStock*, *>, <*inStock_r*, 1.0>, <*order*, *>, <*deliver*, *>, <*deliver_r*, 2.0> };

$C_{W_{BS}}$: {< *Broker_Shop*, *BookBroker_Inst*, *BookBroker _Host* >};

$P_{W_{BS}}$: { $P[Shop]_{Init} \triangleq$ <*register*, 1.5>.$P[Shop]_1$

$P[Shop]_1 \triangleq$ <*inStock*, *>. <*inStock_r*, 1.0>.$P[Shop]_2$

$P[Shop]_2 \triangleq$ *order.deliver.deliver_r*.$P[Shop]_1$ + <*unregister*, 2.0>.$P[Shop]_{Fina}$

$P[Shop]_{Fina} \triangleq \underline{0}$ } ;

Fig. 3. Definition of W_{BS}

Bank : $W_{BA} =< I_{W_{BA}}, A_{W_{BA}}, C_{W_{BA}}, P_{W_{BA}} >$

$I_{W_{BA}}$: { *BankAccount*};

$A_{W_{BA}}$: { <*login*, * >, <*getBalance*, * >, <*getBalance_r*, 1.0 >, <*deposit*, * >, < *deposit_r*, 2.0 >, <*withdraw*, * >, <*withdraw_r*, 1.5 >, < *logout*, * > };

$C_{W_{BA}}$: ϕ ;

$P_{W_{BA}}$: { $P[Bank]_{Init} \triangleq$ <*login*, * >.$P[Bank]_1$

$P[Bank]_1 \triangleq$ <*getBalance*, * >. <*getBalance_r*, 1.0 >.$P[Bank]_1$+ <*deposit*, * >. < *deposit_r*, 2.0 >.$P[Bank]_1$ +<*withdraw*, * >.<*withdraw_r*, 1.5 >.$P[Bank]_1$+ < *logout*, * >.$P[Bank]_{Fina.}$

. $P[Bank]_{Fina.} \triangleq \underline{0}$ } ;

Fig. 4. Definition of W_{BA}

$BookBroker$: $W_{BB} =< I_{W_{BB}}, A_{W_{BB}}, C_{W_{BB}}, P_{W_{BB}} >$

$I_{W_{BB}}$:{ $Broker\_Shop$, $Book\_Shop$, $BankAccount$};

$A_{W_{BB}}$: { <*inStock*, 1.5>, < *inStock_r*, *>, <*order*, 1.0>, <*login*, 2.0>, <*deposit*, 1.5>, <*deposit_r*, *>, <*logout*, 1.5>, <*deliver*, 1.5>, <*deliver_r*, *>, <*register*,*>, <*unregister*, *>};

$C_{W_{BB}}$: {<*Book_shop*, *BookShop_Inst*, *BookShop_Host* >, < *BankAccount*, *Bank_Inst*, *Bank_Host* >};

$P_{W_{BB}}$: {

$P[Broker]_{Init} \triangleq <register,*>.P[Broker]_1$

$P[Broker]_1 \triangleq <getABook,*>.P[Broker]_2$

$P[Broker]_2 \triangleq <inStock, 1.5>.< inStock\_r, *>.<getABook\_r,1.0>.P[Broker]_3$

$P[Broker]_3 \triangleq <order, 1.0>.P[Broker]_4 + <unregister, *>.P[Broker]_{Fina}$

$P[Broker]_4 \triangleq <login, 2.0>.<deposit, 1.5>. <deposit\_r, *>.<logout, 1.5>.<deliver, 1.5>.<deliver\_r, *>.P[Broker]_1$

$P[Broker]_{Fina} \triangleq \underline{0}$ }▯

Fig. 5. Definition of W_{BB}

and at the same time, it holds that whenever $P_{W_{BA}} / D6 \xrightarrow{\tau^m} \xrightarrow{a,\mu} \xrightarrow{\tau^n} P_{W_{BA}}' / D3$, then $(P_{W_{BA}} \|_{ite(W_{BA}.W_{BB})} P_{W_{BB}}) / D5 \xrightarrow{\tau^u} \xrightarrow{a,\gamma} \xrightarrow{\tau^v} (P_{W_{BA}}' \|_{ite(W_{BA}.W_{BB})} P_{W_{BB}}') / D5 \wedge \gamma = \mu$, where $D6 = \phi$.

6 Conclusion

Current research works for the behavioral compatibility mostly appear in the area of component-based software systems. In [3], the authors specify the behavioral compatibility between two components by using π calculus, but they didn't taken into account the scenario under which the new components may change its external behavior, nor did they take into account the scenario where a system may contain multiple components and the stochastic-timing constraints.

In [6], the authors focused on behavioral inheritance in component-based software system by Petri Nets. They presented several rules for testing whether the components can be suitable for the requirements of system by comparing the relations between the component's external behavior and the specifications of the system. They defined a type of inheritance called project inheritance. Through this type of inheritance, the system can ensure that the component replaced by a new one can meet the requirements of the system without influencing its external behavior. But in [6], the author didn't take into consideration the scenario under which a new component has its external requirement for the environment.

The similar works were also presented for ensuring the feasibility of component replacement in [4], [5] and [7]. In these works, the author studied the behavioral subtype relationship between objects by using CSP, and presented three behavioral

subtypes: weak subtype, safe subtype and optimization subtype. These three behavioral subtypes can be used to verify object substitutability in the object-based system, but it has not yet consider the case where software entities replaced may have their external request for stochastic-timing constraints.

In this paper, for stochastic-timing software business process collaboration based on Web services, we take into account the problem of verifying the behavioral compatibility in the collaboration system. We use the extended markovian process algebra to model the behavior of the business processes, and address the issue of formally verification of whether the behavior of system design can meet the requirements of behavioral compatibility from the aspect of stochastic timing. Based on analyzing the bisimulation relation of the parallelism of business processes, we develop a set of theorems to check the compatibility in the collaboration system.

References

1. Vincenzo, A., Reidar, C., Alfonso, F.: Assessing process-centered software engineering environments. ACM Trans. Softw. Eng. Methodol. 6(3), 283–328 (1997)
2. Bernardo, M.: Theory and Application of Extended Markovian Process Algbra.[Ph.D] Dottorato di Ricerca in Informatica, University di Bologna, Padova, Venezia (1999)
3. Canal, C., Pimentel, E., Troya, J.M.: Compatibility and inheritance in software architectures. Science of Computer Programming 41, 105–138 (2001)
4. Wehrheim, H.: Relating State-based and Behaviour-oriented Subtyping. Nordic Journal of Computing 9(4), 405–435 (2002)
5. Schrefl, M., Stumptner, M.: Behavior-Consistent Specialization of Object Life Cycles. ACM Transaction on Software Engineering and Methodology 11(1), 92–148 (2002)
6. Van Der Aalst, W.M.P., Toorn, R.A.: Component-based software architectures: a framework based on inheritance of behavior. Science of Computer Programming 42, 129–171 (2002)
7. Wehrheim, H.: Checking Behavioural Subtypes via Refinement. In: FMOODS 2002: Formal methods for Open Object-Based Distributed Systems, pp. 79–93. Kluwer, Dordrecht (2002)
8. Bernardo, M., Ciancarini, P., Donatiello, L.: Architecting families of software systems with process algebras. ACM Transaction on Software Engineering and Methodology 11(4), 386–426 (2002)
9. Canal, C., Pimentel, E., Troya, J.: Specification and refinement for dynamic software architectures. In: Software Architecture, pp. 107–126. Kluwer Academic Publisher, Netherlands (1999)
10. Allen, R., Garlan, D.: A formal basis for architectural connection. ACM Transaction on Software Engineering and Methodology 16(3), 213–249 (1997)
11. Seabastian, S., Terry, R.P., Nicholas, R.J.: Flexible provisioning of Web service workflows. ACM Transaction on Internet Technology 9(1), 201–245 (2009)
12. Hu, H., Lu, J., Ma, X., Tao, X.: Research on Behavioral Compatibility of Components in Software Architecture Using Object-Oriented Paradigm. Journal of Software 17(6), 1276–1286 (2006)

An Efficient Blind Ring Signature Scheme without Pairings

Jianhong Zhang[1], Hua Chen[1], Xue Liu[1], and Chenglian Liu[2]

[1] College of Science, North China University of Technology, Beijing 100041, P.R.China
jhzhangs@gmail.com
[2] Department of Mathematics, Royal Holloway, University of London, UK
cheng-lian.liu@gmail.com

Abstract. Anonymous technology can realize the privacy-preserving of the user's information, thus it plays an important role in the e-commerce application. As two important cryptographic techniques, blind signature and ring signature can realize anonymity. Thus, it makes that they play very important roles in realizing the anonymity of user's information. To fit with the demand in real life, we propose a blind ring signature scheme based on algebraic curves by combining blind signature and ring signature. And we give formal security proof of anonymity, blindness and unforgeability of our scheme in the random oracle model. By comparing our scheme with Herranz et al's scheme which is state-of-the-art, in terms of computational cost and the signature length, we show that our scheme is more efficient, since no pairing computations are involved in the whole signing and verifying phase.

Keywords: blind ring signature, unforgeability, blindness, pairings, anonymity, random oracle.

1 Introduction

In recent years, authentication procedures have been increasing to ensure security of the individual. Simultaneously, users are becoming increasingly worried about infringement of the privacy, as they fear authorities are tracking their whereabouts and activities. However, allowing complete anonymous access for reasons of privacy could result in an alarming amount of crime. It has long been desired to develop technologies that both fulfill security and privacy needs. On the other side, the privacy preserving of a user's transaction information is also need to be provided. For example, every time you make a telephone call, purchase goods using a credit card, subscribe to a magazine or pay your taxes, that information goes into a database somewhere. Furthermore, all these records can be linked so that a malicious adversary can reveal the actual identity of the user. Thus, the protection of the transaction information is also very important.

Blind signature was introduced by D.Chaum [7], which can provide the anonymity of the signed message, thus, it can realize the private protection

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 177–188, 2010.

of user's transaction information. Since it was introduced, it attracts many researcher's interests. Blind signature schemes[4,5,6,7,8,9,10] have been widely applied in real life, most prominently in anonymous voting and anonymous e-cash.

Informally, blind signature allows a user to obtain signatures from an authority on any document, in such a way that the authority learns nothing about the message that is being signed. The most important property of blind signature differing from the other signatures is *blindness*, which requires that after interacting with various users, the signer S is not able to link a valid message-signature pair (m, δ) obtained by some user, with the protocol session during which δ was created. The other property is unforgeability, requires that it is impossible for any malicious user that engages in k runs of the protocol with the signer, to obtain strictly more than k valid message-signature pairs. The basic idea of most existing blind signatures is that the requester randomly chooses some random factors and embeds them to the message to be signed. The random factors are kept in secret so the signer cannot recover the message. Upon the blinded signature returned by the signer, the requester can remove the random factor to obtain a valid signature. Up to now, many variants of blind signature are proposed, such as ID-based blind signature schemes [10,5] , proxy blind signatures [5,2].

Ring signature is a group-oriented signature with privacy concerns. It was introduced by Rivest, Shamir and Tauman[21] to provide anonymity for the message signer. In a ring signature scheme, the message signers form a ring of any set of possible signers and himself. The actual signer can then generate a ring signature entirely using only his secret key and the others' public keys without the assistance or even awareness of the other ring members. However, the generated ring signature can convince an arbitrary verifier that the message was indeed signed by one of the ring members while the real signer's identity is totally anonymous to the verifier. Thus, ring signatures have the following main properties: anonymity, spontaneity, and unlinkability. Since the introduction of ring signature, several related ring signature schemes[11,12] and concepts have been proposed, including convertible ring signature [3] and linkable ring signature [15] Recently, this notion has been extended by Naor by introducing Deniable Ring Authentication [13]: it is possible to convince a verifier that a member of an ad-hoc subset of participants is authenticating a message without revealing which member has issued the signature, and the verifier V cannot convince any third party that message m was indeed authenticated.

The original motivation for constructing blind ring signature came from e-vote, e-cash and untraceable payments. To satisfy the demand of real life scenarios, we must make a single e-bank system more scalable by supporting many banks, and adding some other properties like strong anonymity of the signing banks, unlinkability of two different signatures, revocation facilities. Because blind signature can provide the anonymity of the signed message and ring signature can provide the anonymity of the signer's identity. Obviously, combining blind signature and ring signatures is able to bring solutions to the above scenarios of e-banking or e-auctions. Indeed ring signatures provide more spontaneity to the design of such systems. Motivated the above idea, blind ring signature

also can been applied to e-bid or e-vote in order to protect privacy of signer and the signed message.

In the paper, we propose a blind ring signature scheme based on algebraic curves by combining blind signature and ring signature. And we give formal security proof of anonymity, blindness and unforgeability of our scheme in the random oracle model. By comparing our scheme with Herranz's scheme in terms of computational cost and the signature size, we show that our scheme is more efficient, no pairing computation is required in the signing and verifying phase.

2 Preliminaries

This section gives some cryptographic primitives and security notions of blind ring signature.

Definition 1. *Discrete Logarithm (DL) Problem: Let* $\mathbb{G}_1$ *be a additive group of prime order* q *and* P *be a generator of* $\mathbb{G}_1$*. The DL problem to the base* P *means the following problem:*
Given $P, Q \in \mathbb{G}_1$*, where* $Q = xP$ *for some unknown* x*, its goal is to find* x*.*

The DL problem is believed to be difficult and also to be the hard direction of a one-way function.

2.1 Blind Ring Signature

In the following, we consider blind ring signature notion and some security requirements. In general, a secure blind ring signature (BRS) consists of the following algorithms:

1. **Setup of system parameters:** BRS.Setup is a probabilistic algorithm which takes as input k and outputs system parameters (which include a description of the signature space and message space, hash functions, etc.);
2. **Key generation:** BRS.KeyGen is a probabilistic algorithm which takes as input the system parameters and outputs a signing key pair (pk_j, sk_j) for a user U_j . The value pk_j is made public, where the value sk_j is secretly stored by user U_j .
3. **Blind ring signature generation:** BRS:Sign is an interactive two-party protocol which is initialized by a client C. This client chooses a message M and a ring $U = (U_1, \cdots, U_n)$ of users, and engages an interaction with some of the members U_π of the ring, who can use his secret key sk_π as part of the input. We denote as If_C the secret inputs that client C uses, and as T_{sig} the values that are obtained by the signer, during this interaction.
   At the end, the private output O_C for the client is a valid ring signature δ for the message M and the ring of users U.
4. **Verification of a blind ring signature:** BRS:Verify is a deterministic algorithm which takes as input a message M, a ring of users $U = (U_1, \cdots, U_n)$ their public keys $(pk_1, \cdots, pk_n)$ and a bit string . The output is 1 if the signature is valid, and 0 otherwise.

The security of a secure blind ring signature scheme should satisfy the following four primitive properties:

1. **Correctness** means that a verifier always accepts as valid a signature that has been properly generated by a honest client and a honest signer in the corresponding ring of users
2. **Anonymity** means that the client has no information about which member of the ring has actually participated in the interactive blind ring signature generation.
3. **Blindness** intuitively means that the users in the ring obtain no information about the message that they are actually signing for the client. In some sense, blindness also means unlikability.
4. **Unforgeability** means that a client is not able to produce $l+1$ valid and different ring signatures if he has queried for at most l executions of the blind ring signature protocol.

In the above security properties of blind ring signature, blindness and unforgeability are two important security properties. Let us recall the definition of the two properties [8].

Blindness is defined by a game played between a challenger and an adversary. This adversary $\mathcal{B}$ simulates the dishonest behavior of a ring of users who try to distinguish which of two different messages m_0 and m_1 is being signed in an interactive execution of the signing protocol with a client. The game is as follows:

- Setup: the adversary $\mathcal{B}$ chooses a user set U and a security parameter k. The challenger runs the setup protocol of the blind ring signature scheme with input k, as well as the key generation protocol for each user $U_j \in U$. The adversary $\mathcal{B}$ is given all the resulting information: the public common parameters, the public and secret keys of all users in the user set U.
- Challenge: the adversary chooses a user list $U_L = \{U_1, \cdots, U_t\}$, and two different messages m_0 and m_1. The challenger chooses at random one bit $b \in \{0, 1\}$ and initializes the interactive blind ring signature protocol with message m_b and a user list U_L as inputs. The adversary $\mathcal{B}$ chooses some user U_π and plays the role of the signer in the protocol . In the process, the adversary can obtains the information $Info_{sig}$ which the signer obtains from an execution of the signing protocol.
- Guess: Finally, the adversary $\mathcal{B}$ outputs its guess b'.

We say that such an adversary $\mathcal{B}$ succeeds if $b = b'$. The advantage of the adversary is defined by

$$Adv^{Bld}_{BRS,A}(k) = 2Pr[b = b'] - 1$$

A blind ring signature scheme satisfies the blindness property, if for any PPT adversary, the function $Adv^{Bld}_{BRS,A}(k)$ is negligible in k.

Unforgeability is a primitive property of digital signature. For unforgeability of blind ring signature, it should satisfy $(l, l+1)-$unforgeability of blind signature. Thus, the unforgeability of blind ring signature is similar to that of blind

signature. It is defined by the following game played between the challenger and the adversary:

- Setup: the adversary $\mathcal{A}$ chooses a user set U_L and a security parameter k. The challenger runs the setup protocol of the blind ring signature scheme with input k, as well as the key generation protocol for each user $U_j \in U_L$. It gives to the adversary $\mathcal{A}$ the resulting common parameters and all user's public keys pk_j in user set U_L, and keeps secretly all user' secret keys sk_j of user set U_L .
- Queries: the forger $\mathcal{A}$ makes different queries to the challenger:
  1. Hash queries: in the scheme, hash function H is assumed to behave as a random oracle [4] in the security proof, then the challenger must answer q_h queries of the adversary to this oracle, providing it with consistent and totally random values.
  2. Blind ring signature queries: an adversary can make at most q_s blind ring signature queries with $(M_i, R_{i_1}, \cdots, R_{i_t})$. The challenger must answer with a valid blind ring signature δ_i for this pair message/ring of users. All these queries can be made in an adaptive way; that is, each query may depend on the answers obtained to the previous queries.
- Forgery: the adversary $\mathcal{A}$ outputs a list of $q_s + 1$ tuples (M_i, U_{L_i}, δ_i). We say that $\mathcal{A}$ succeeds if:
  1. the $q_s + 1$ bind ring signatures are valid;
  2. and $(M_i, U_{L_i}) \neq (M_j, U_{L_j})$, for all indices $1 \leq i, j \leq q_s + 1$ such that $i \neq j$.

3 Our Blind Ring Signature Scheme without Pairings

In the section, we propose a novel blind ring scheme without pairings by combining ring signature and the blind signature [1]. It can provide the anonymity of the signer and the signed message. The protocols of the new scheme are described below.

Setup and key generation. On input a security parameter k, an additive group $\mathbb{G}_1$ of prime order $q > 2^k$, generated by some element P. A hash function $H : \{0,1\}^* \times \mathbb{G}_1^n \to \mathbb{Z}_q$. All these parameters are common and public.

Each user U_i chooses his secret key $x_i \in \mathbb{Z}_q$ at random; the corresponding public key is $Y_i = x_i P \in \mathbb{G}_1$.

Blind ring signature generation. If the client wants to obtain a blind ring signature on a message M with respect to a ring $U = \{U_1, \cdots, U_n\}$, then it needs to execute the following steps:

1. A certain member in the ring U, say U_π, where $\pi \in \{1, \cdots, n\}$, for $i = 1$ to n and $i \neq \pi$, randomly chooses $(c_i, s_i) \in \mathbb{Z}_q$ to compute $R_i = s_i P + c_i Y_i$.
2. choose at random $k \in \mathbb{Z}_q$ to compute $R_\pi = kP$
3. Then U_π sends $(R_1, \cdots, R_n)$ to the Client.
4. Upon receiving $(R_1, \cdots, R_n)$ from the ring, the Client firstly chooses $l \in \mathbb{Z}_q$,then for all $i \in \{1, 2, \cdots, n\}$, he randomly chooses $v_i, u_i \in \mathbb{Z}_q$ to compute $\hat{R}_i = lR_i + v_i P + u_i Y_i$. Finally, compute

$$c = H(M, \hat{R}_1, \cdots, \hat{R}_n)$$

Finally, the Client sends $c' = c/l$ to the ring.

5. After receiving c', the member U_π of the ring computes

$$c_\pi = c' - \sum_{i=1, i\neq\pi}^{n} c_i \ , \ s_\pi = k - c_\pi x_\pi$$

Then, for $i = 1$ to n, the member U_π sends (s_i, c_i) to the Client.

6. On receiving all $\{(c_1, s_1), \cdots, (c_n, s_n)\}$, for $i = 1$ to n, the Client computes

$$c'_i = l \cdot c_i + u_i \ , \ s'_i = l \cdot s_i + v_i$$

7. Finally, the resultant blind ring signature of message M made by the ring $U = (U_1, \cdots, U_n)$ is $((c'_1, s'_i), \cdots, (c'_n, s'_n))$.

Verification of a blind ring signature. Given a blind ring signature $((c'_1, s'_i), \cdots, (c'_n, s'_n))$ of message M made by the ring, a verifier can execute as follows:

1. for $i = 1$ to n, compute $R'_i = s'_i P + c'_i Y_i$
2. check whether the following equation holds:

$$\sum_{i=1}^{n} c'_i = H(M, R'_1, \cdots, R'_n) \tag{1}$$

If the signature satisfies the above equation (1), it means that the signature is valid.

4 Security Analysis

In this section, we will discuss the correction and security of our signature scheme. Firstly, we show that our signature scheme satisfies completeness.

Theorem 1. *(Completeness.) If the signer of the ring and the Client follow the above protocol, then the output will be accepted by the verification procedure.*

Proof. From the protocol, we have that

For $i = 1$ to n and $i \neq \pi$

$$R'_i = s'_i P + c'_i Y_i = l(s_i P + c_i Y_i) + v_i P + u_i Y_i = l R_i + v_i P + u_i Y_i$$

When $i = \pi$, we have

$$\begin{aligned} R'_\pi &= s'_\pi P + c'_\pi Y_\pi \\ &= l(s_\pi P + c_\pi Y_\pi) + v_\pi P + u_i Y_\pi \\ &= l((k - c_\pi x_\pi)P + c_\pi Y_\pi) + v_\pi P + u_i Y_\pi \\ &= lkP + v_\pi P + u_i Y_\pi \end{aligned}$$

Thus, we can know that

$$\begin{aligned} H(M, R'_1, \cdots, R'_n) &= H(M, \hat{R}_1, \cdots, \hat{R}_n) = c \\ &= \sum_{i=1, i\neq\pi}^{n} c_i + (c - \sum_{i=1, i\neq\pi}^{n} c_i) = \sum_{i=1, i\neq\pi}^{n} c_i + c_\pi \\ &= \sum_{i=1}^{n} c_i \end{aligned}$$

4.1 Unlinkability of the Scheme

Because our scheme is a blind ring signature, obviously, ring signature can ensure that the anonymity property holds unconditionally: that is to say, even if a client has unlimited computational resources, he cannot obtain any information about which member has actually participated in the interactive protocol to compute a blind ring signature. Note that unconditional anonymity directly implies the other property: **unlinkability** , which means that nobody (including the client) will be able to distinguish if two different interactive executions of the blind ring signature protocol have been performed by the same member of the ring or not. In effect, if a scheme is linkable, then there exists a polynomial-time linking algorithm which takes as input two executions of the blind ring signature protocol and outputs 1 if and only if the same member of the ring has participated in both executions. If this holds, then a client with unlimited resources who tries to break the anonymity of some execution of the protocol can act as follows: (1) he obtains all the secret keys of the members of the ring; (2) for each member U_i of the ring, the client uses the obtained secret key to run by himself a new interactive execution of the blind ring signature protocol; (3) the client applies the linking algorithm to this last execution and to the initial execution whose anonymity he is trying to break; (4) if the output of the linking algorithm is 1 for user U_i, then this user was the one who participated in the initial (target) execution.

4.2 Blindness of the Scheme

To show the proposed scheme satisfying unconditional blindness, we only prove that the probability distribution of the information $Info_{sig}$ that the member U_π of the ring obtains in an execution of the signing protocol is exactly the same for any possible message. In the case of our scheme, we have $Info_{sig} = (s_1, c_1, \cdots, s_n, c_n, R_1, \cdots, R_n, c')$, where each pair (s_i, c_i) is produced by random selection of the member U_π of the ring.

In affect, the value $c' = \frac{H(M, lR_1 + v_1P + u_1Y_1, \cdots, lR_n + v_nP + u_nY_n)}{l}$ follows a completely random and uniform distribution in $\mathbb{Z}_q$, independently of the message M, because all the values $(R_1, \cdots, R_n)$ are randomized by the random numbers $l, u_1, \cdots, u_n, v_1, \cdots, v_n \in \mathbb{Z}_q$. For a blind ring signature $((c'_1, s'_i), \cdots, (c'_n, s'_n))$ and $(c_1, s_1, \cdots, c_n, s_n)$ in $Info_{sign}$, we know that they satisfy $c'_i = l \cdot c_i + u_i, s'_i =$

$l \cdot s_i + v_i$. Each pair (c_i, s_i) in Inf_{sig} is randomized by random numbers (u_i, v_i). Thus, in any case, their probability distribution does not depend on the signed message M. Furthermore, given a blind ring signature on message M, the signature also may been produced by the other members of ring. Therefore, the scheme achieves perfect blindness.

4.3 Unforgeability of the Scheme

In the subsection, we are going to prove that our scheme is unforgeable in the random oracle model,and the security of the scheme is related to the difficulty of solving the discrete logarithm problem. We denote as q_1 the number of queries that an adversary $\mathcal{A}$ against the unforgeability of our scheme can make to the (random) oracle which models the behaviour of the hash function $H(\cdot)$.

Theorem 2. *If there exists a $(q_s, q_s{+}1; q_h)$-forger $\mathcal{A}$ against the unforgeability of our blind ring signature scheme without pairings, which succeeds with probability ϵ, then there exists a $(q_h; q_1)$-Adversary $\mathcal{B}$ which can solve the discrete logarithm problem with probability $\varepsilon > \frac{\epsilon^2}{66V_{n,t}}(1 - \frac{q_s+1}{q})$.*

Proof. Assume there is a $(\epsilon, t, q_s, q_h)-$adversary $\mathcal{A}$ exists. We are going to construct another PPT $\mathcal{B}$ that makes use of $\mathcal{A}$ to solve the discrete logarithm problem with probability at least ϵ' and in time at most t'.

$\mathcal{B}$ is given a problem instance as follow: Given a group $\mathbb{G}_1$, a generator $P \in \mathbb{G}_1$, $Q = xP \in \mathbb{G}_1$ is a random element of group $\mathbb{G}_1$. It is asked to output the discrete logarithm x of Q to the base P. In order to use $\mathcal{A}$ to solve for the problem, $\mathcal{B}$ needs to simulates a challenger and the oracles (the hash oracle and the signing oracle) for $\mathcal{A}$. $\mathcal{B}$ does it in the following way.

In the setup phase, let $\mathcal{U}$ be a user set. For each user $U_i \in \mathcal{U}$, $\mathcal{B}$ randomly chooses a value $\alpha_i \in \mathbb{Z}_q$ ant sets the corresponding public key as $Y_i = \alpha_i Q$. Randomly choose a hash function $H(\cdot)$. Finally, $\mathcal{B}$ sends to $\mathcal{A}$ all the system parameters $(q, \mathbb{G}_1, P, H(\cdot))$ and the public keys Y_i of all the users U_j in the user set $\mathcal{U}$. Note that hash function H acts as a random oracle.

Hash Oracle: when the adversary $\mathcal{A}$ makes at most q_h queries on Hash oracle with $(M, \hat{R}_1, \hat{R}_2, \cdots, \hat{R}_n)$, $\mathcal{B}$ firstly checks whether there exists an item $(M, \hat{R}_1, \hat{R}_2, \cdots, \hat{R}_n, c)$ in the H-list which is initially empty. If it exists, then $\mathcal{B}$ returns $c = H(M, \hat{R}_1, \hat{R}_2, \cdots, \hat{R}_n)$ to the adversary $\mathcal{A}$. Otherwise, $\mathcal{B}$ randomly chooses $c \in \mathbb{Z}_q$ and returns it to $\mathcal{A}$. Finally, $\mathcal{B}$ adds $(M, \hat{R}_1, \hat{R}_2, \cdots, \hat{R}_n, c)$ in the H-list.

Blind ring signature oracle: When an adversary $\mathcal{A}$ makes at most q_s queries On **Blind Ring Signature Oracle** with $(M, U' = (U_{i_1}, \cdots, U_{i_t}))$, where U' is a subset of user set $\mathcal{U}$. $\mathcal{B}$ responses as follows:

1. it randomly chooses a user $U_\pi \in U'$. For $j \in \{i_1, \cdots, i_t\}$, $\mathcal{B}$ randomly selects $\hat{s}_j, \hat{c}_j \in \mathbb{G}_1$ to compute
$$\hat{R}_j = \hat{s}_j P + \hat{c}_j Y_j$$

2. Then, $\mathcal{B}$ checks whether there exists a $t+1-$ tuple $(M, \hat{R}_{i_1}, \cdots, \hat{R}_{i_t})$ in the $H-$list. If it exists, then abort it. Otherwise, $\mathcal{B}$ sets $H(M, \hat{R}_{i_1}, \cdots, \hat{R}_{i_t}) = \sum_{j=1}^{t} c_{i_t}$. And add it in the $H-$list. The failure of simulation happens with probability at most q_h/q. Hence, the simulation is successful q_s times with probability at least $(1-q_h/q)^{q_s} \geq 1 - q_h q_s/q$.
3. Finally, $\mathcal{B}$ returns $(\hat{s}_{i_1}, \hat{t}_{i_1}, \cdots, \hat{s}_{i_t}, \hat{c}_{i_t})$ to $\mathcal{A}$ as the blind ring signature on message M.

Output: Eventually, $\mathcal{A}$ outputs a forged blind ring signature $(s_1^*, c_1^*, \cdots, s_m^*, c_m^*)$ on message M^* with the probability ϵ under the user set $U^* \subseteq \mathcal{U}$, where M^* is not queried on the blind ring signing oracle. And the signature satisfies

$$\begin{aligned}
\hat{R}_j &= s_j P + c_j Y_j, \text{where} j \in \{i_1, \cdots, i_t\} \\
c_\pi &= H(M, \hat{R}_{i_1}, \cdots, \hat{R}_{i_t}) - \sum_{j=1, j\neq\pi} c_{i_j} \\
s_\pi &= k - c_\pi \alpha_\pi x
\end{aligned}$$

Applying forking lemma [18,8], by replaying the same random numbers with different random oracle, we can obtain another valid forgery blind ring signature $(s'^*_1, c'^*_1, \cdots, s'^*_m, c'^*_m)$ with probability $\varepsilon > \frac{\epsilon^2}{66 V_{n,t}}$ where for $i = 1$ to n and $i \neq \pi$, $s_i^* = s'^*_i$ and $c_i^* = c'^*_i$. And the signature also satisfies

$$\begin{aligned}
\hat{R'}^*_j &= s'^*_j P + c'^*_j Y_j, \text{where} j \in \{i_1, \cdots, i_t\} \\
c'^*_\pi &= H(M, \hat{R'}^*_{i_1}, \cdots, \hat{R'}^*_{i_t}) - \sum_{j=1, j\neq\pi} c'^*_{i_j} \\
s'_\pi &= k - c'^*_\pi \alpha_\pi x
\end{aligned}$$

Thus, we have the following relation

$$\begin{aligned}
s'_\pi - s_\pi &= k - c'^*_\pi \alpha_\pi x - (k - c_\pi \alpha_\pi x) = c_\pi \alpha_\pi x - c'^*_\pi \alpha_\pi x \\
&= (c_\pi \alpha_\pi - c'^*_\pi \alpha_\pi) x
\end{aligned}$$

Finally, we know

$$x = \frac{s'_\pi - s_\pi}{c_\pi \alpha_\pi - c'^*_\pi \alpha_\pi} = \alpha_\pi^{-1} \frac{s'_\pi - s_\pi}{c_\pi - c'^*_\pi}$$

Then, the discrete logarithm Q to the base P is solved.

Obviously, the environment of $\mathcal{A}$ is perfectly simulated by $\mathcal{B}$. Since the hash function $H(\cdot)$ is assumed to behave as a random function, the probability that $\mathcal{A}$ obtains a valid blind ring signature for (M^*, U^*) without asking for the value $H(\cdot)$ is $1/q$. Therefore, we have that with probability $1 - \frac{q_s+1}{q}$ the adversary $\mathcal{A}$ has queried the random oracle with $(M, R_{i_1}, \cdots . R_{i_t})$ for the $q_s + 1$ forged pairs. This means that, for $i = 1, \cdots, q_s + 1$, we have $H(M_i, R_{i_1}, \cdots, R_{i_t}) = c_i$, where c_i is a random number of Z_q. Thus, $\mathcal{B}$ can solve the discrete logarithm with probability $\varepsilon > \frac{\epsilon^2}{66 V_{n,t}}(1 - \frac{q_s+1}{q})$, where $V_{n,t}$ denotes the number of $t-$ permutations of n elements, that is $V_{n,t} = n(n-1)\cdots(n-t+1)$. □

4.4 Efficiency Analysis

Up to now, only few blind ring signature schemes have been proposed. In 2005, Chan *et.al* proposed the first one [15]. But their scheme is obscure and it is unclear who actually engages the different protocols. In 2006, Wu *et.al* give a static blind ring signature based on RSA. In the same year, J. Herranz *et.al* gave an efficient blind ring signature scheme[8] based on pairings, it was state-of-the-art. In the following, we compare our scheme with Herranz's scheme [8] in terms of signature size and the required computational cost of signing and verifying. Pairing operator is the most time consuming operation in Elliptic Curve Cryptosystems (ECC).

For convenient comparison, we instantiate pairing-based schemes using Barreto -Naehrig curves [17] with 160- bit point representation. and we include the following presentation, the notion $|\mathbb{G}_1|$ denotes the bit length of an element in $\mathbb{G}_1$, $|q|$ be the binary length of an element in $\mathbb{Z}_q$, P_m be scalar multiplication on the curve and P_e be pairing computation. The hash function maps a string to an element in $\mathbb{G}_1$ used by the scheme [10] usually requires "Maptopoint operation" [22]. As discussed in [22], Maptopoint operator (denoted by H)is so inefficient that we can't neglect it. However, the hash function used in our scheme which maps a string to an element in Z_q is very efficient so that it can been neglected. In the following, the detail comparison is shown in table1.

Table 1. Comparison of our scheme with Herranz*et al*'s scheme [8]

Scheme	Size	PK size	verifying cost	Signing cost
Herranz *et al*'s scheme	$\|q\| + n\|\mathbb{G}_1\|$	$n\|\mathbb{G}_1\|$	$(n+1)P_e$	$(4n-1)P_m + (n+1)P_e$
Our scheme	$2n\|q\|$	$n\|\mathbb{G}_1\|$	$2nP_m$	$(4n-1)P_m$

PK denotes the length of all user's public key.

From Table 1, we know that Herranz's scheme has higher computational cost than our scheme in term of generation and verification of signature. We note that the computation of the pairing is the most time-consuming in pairing based cryptosystems. Although there have been many papers discussing the complexity of pairings and how to speed up the pairing computation [9,20], the computation of the pairing still remains time-consuming.Thus, our scheme is an efficient and usable scheme.

5 Conclusion

Anonymous technology is an important tool to realize the privacy-preserving of the user and plays an important role in the e-commerce application. Blind signature can provide the anonymity of the signed message, ring signature can realize the anonymity of the actual identity of the signer. Such that they are widely applied in the protection of the user sensitive information . To satisfy

the requirement in real life, we propose a blind ring signature scheme based on algebraic curves by combining blind signature and ring signature. And we give formal security proof of anonymity, blindness and unforgeability of our scheme in the random oracle model. By comparing our scheme with Herranz's scheme in terms of computational cost and the signature size, we show that our scheme is more efficient, no pairing computations are required in the signing and verifying phase. But the length of the signature depends on the number of member. Thus, it is an open problem to construct a blind ring signature with constant size.

References

1. Abe, M., Okamoto, T.: Provably secure partially blind signatures. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 271–286. Springer, Heidelberg (2000)
2. Zhang, B., Xu, Q.: Certificateless proxy blind signature scheme from bilinear pairings. In: WKDD 2009, pp. 573–577 (2009)
3. Lee, K.-C., Wen, H.-A., Hwang, T.: Convertible tures from a variety of keys. In: lEE Proceedings - Communications, ASIACRYPT. Lect. Notes ring signature, vol. 152(4), pp. 411–414 (2005)
4. Bellare, M., Rogaway, P.: Random Oracles are Practical: a Paradigm for Designing Efficient Protocols. In: Proceedings of CCS 1993, pp. 62–73. ACM/Academic Press (1993)
5. Pan, H., Ke, K., Gu, C.: Efficient ID-based Proxy Blind signature scheme from Pairings. In: CIS 2008, pp. 390–394 (2008)
6. Cramer, R., Shoup, V.: A Practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
7. Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptology - Crypto 1982, pp.199–203. Springer, Heidelberg (1983)
8. Herranz, J., Laguillaumie, F.: Blind Ring Signatures Secure Under the Chosen-Target-CDH Assumption. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 117–130. Springer, Heidelberg (2006)
9. Duursma, I.M., Lee, H.-S.: Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 111–123. Springer, Heidelberg (2003)
10. Zhang, F., Kim, K.: Efficient ID-based blind signature and proxy signature from bilinear pairings. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 312–323. Springer, Heidelberg (2003)
11. Bender, A., Katz, J., Morselli, R.: Ring Signatures: Stronger Definitions, and Constructions without Random Oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 60–79. Springer, Heidelberg (2006)
12. Herranz, J., Saez, G.: Forking lemmas in the ring signatures' scenario. Technical Report 067, International Association for Cryptologic Research (2003), http://eprint.iacr.org/2003/067.ps
13. Naor, M.: Deniable Ring Authentication. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 481–498. Springer, Heidelberg (2002)
14. Boldyreva, A.: Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2003)

15. Chan, T.K., Fung, K., Liu, J.K., Wei, V.K.: Blind Spontaneous Anonymous Group Signatures for Ad Hoc Groups. In: Castelluccia, C., Hartenstein, H., Paar, C., Westhoff, D. (eds.) ESAS 2004. LNCS, vol. 3313, pp. 82–84. Springer, Heidelberg (2005)
16. Joux, A., Nguyen, K.: Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. Journal of Cryptology 16, 239–247 (2003)
17. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
18. Pointcheval, D., Stern, J.: Security Proofs for Signature Scheme. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 387–398. Springer, Heidelberg (1996)
19. Steinfeld, R., Zheng, Y.: A Signcryption Scheme Based on Integer Factorization. In: Okamoto, E., Pieprzyk, J.P., Seberry, J. (eds.) ISW 2000. LNCS, vol. 1975, pp. 308–322. Springer, Heidelberg (2000)
20. Galbraith, S.D., Harrison, K., Soldera, D.: Implementing the Tate pairing. In: Fieker, C., Kohel, D.R. (eds.) ANTS 2002. LNCS, vol. 2369, pp. 324–337. Springer, Heidelberg (2002)
21. Rivest, R.L., Shamir, A., Tauman, Y.: How to Leak a Secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
22. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)

Improving the Throughput of Wireless Mesh Networks for Web Services*

Hua Hu[1,2], Zheng Zhang[2], and Haiyang Hu[1,2]

[1] College of Computer Science, Hangzhou Dianzi University
[2] College of Computer Science and Information Engineering, Zhejiang Gongshang University

Abstract. With the development of e-business, web-based application model has been widely used; web service has become one of the technologies which can change the software application model. Through web services, various information resources and services in the Internet are integrated, people can use these high quality services at any time, any place. However, with the further development of web services, high performance communicate networks are needed. Traditional wireless networks due to its access mode characteristics and rate constraints, cannot satisfy the demand for mobile access services, but also limits the efficient implementation of web services. In order to make network support web service better, this paper has studied the next generation networks, wireless mesh network. Given multi-channel and multiple network interface cards wireless mesh network physical topology through the extended multiple network interface cards connect graph we joint channel allocation, interface assignment and present a fix channel allocation algorithm by using the linear programming to improve the overall network performance.

1 Introduction

As we all know, with the continuous development of e-business, web-based application model has been widely used; web service has become one of the technologies which can change the software application model. Through web services, various information resources and services in the Internet are integrated, people can use these high quality services at any time, any place. Web service is a service-oriented architecture technology; through a standard web protocols to provide services. Nowadays with the further development of web services, high performance networks are needed. Current wireless networks cannot well satisfy web services' requirements for bandwidth, network performance and network throughput; at the same time it also can't support mobile access to Internet. The emergence of wireless mesh network [1] is a good solution to these problems.

Wireless mesh networks consist of two types of nodes: mesh routers and mesh clients; mesh routers constitute the backbone of the wireless mesh network. Through the

* This paper is supported by National Natural Science Foundation of China granted by Nos. 60873022 and Nos. 60903053, the open fund provided by State Key Laboratory for Novel Software Technology of Nanjing University, and the Natural Science Foundation of Zhejiang Province of China under Grant No.Y1080148."

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 189–197, 2010.

gateway, the backbone of wireless mesh networks is connected with Internet. Mesh clients use the interfaces which are provided by the backbone of wireless mesh network to access the Internet. In the backbone, most mesh routers' position is fixed. Since these mesh routers use the uninterrupted power, there is no need to take the issue of power constraint into account. When a WMN uses multiple channels and network interface cards, the network capacity will be improved more, note that each network interface card within a router should be assigned to a distinct orthogonal frequency channel. In the IEEE 802.11x standards, 802.11b and 802.11a [2] [3] offer 3 and 12 orthogonal frequency channels respectively. In a multi-channel network, it allows simultaneous communication with an interference links if they are tuned on different orthogonal frequency channels.

Compared with the traditional wireless networks, the wireless mesh networks greatly improve the throughput of the network, due to its features of multi-channel and multiple network interface cards. Whereas, if we don't assign channels, interfaces properly and schedule the network carefully, the channel allocation and schedule of confusion network will seriously undermine the performance of wireless mesh networks, which leads to a lower throughput.

In this paper, so as to make wireless mesh networks support web services better, we have studied how to assign channel with each mesh router's network interface card in multi-channel wireless mesh networks which have multiple network interface cards to minimize the interference among mesh routers. After then, we propose an optimization problem, using linear programming to improve the overall network throughput.

The rest of the paper is organized as follows. We discuss related work in section 2. Our system model is described in section 3. Our proposed algorithm is presented in section 4. We present optimal schemes for throughput maximization in section 5. Our simulation results in section 6. Finally the paper is concluded in section 7.

2 Related Work

Many researches on channel allocation have done for wireless mesh networks. In [4] the authors proposed the Connected Low Interference Channel Assignment (CLICA) algorithm for MC-WMNs. In [5] the authors formulate the logical topology design, interface assignment, channel allocation, and routing as a joint linear optimization problem; through the solution of this problem the wireless mesh network achieves a good performance at throughput and end-to-end delay. In [6], the authors studied interference-aware topology control and QoS routing in IEEE802.11-based multi-channel wireless mesh networks, and present a novel definition of co-channel interference. In [7] the authors formulate the joint channel allocation, interface assignment, and media access control problem as a cross-layer non-linear mixed-integer network utility maximization problem. In [8] the authors mathematically formulate the joint channel assignment and routing problem, taking into account the network resources. In [9] the authors use the multi-radio conflict graph to present an interference-aware channel assignment algorithm and protocol for multi-radio wireless mesh network that address interference problem. In [10], the authors studied the joint rate control, routing and scheduling in multi-channel wireless mesh network and then

using linear and convex programming to solve the optimization problem. In [11], the authors studied the problem gateway placement for throughput optimization in multi-hop wireless mesh network and proposed a novel grid-based gateway deployment method. In [12], the authors studied bandwidth allocation in multi-channel multi-hop wireless mesh network; they using the optimization to solve the bandwidth allocation problem.

3 System Model

In this paper, we focus on the backbone of wireless mesh network, in which most mesh routers are stationary, so the physical topology of the network is fixed. In our network, we assume that there are C(C>1) orthogonal frequency channels, and that each mesh router has been equipped with K (K>1) network interface cards. It is clear that different mesh routers have different channels. To avoid self-interference, any two network interface cards at the same mesh router should be tuned on different channels. Since most mesh routers use the uninterrupted power, we suppose that they have the same power, transmission range r and interference range r_i ; sometimes r_i is 2 to 3 times of r [13]. Since mesh routers send signal in broadcast forms, they form two round ranges R^u and R_i^u with radius r and r_i respectively. As the network physical topology is fixed, we assume that the distance between any two mesh routers is already known.

From [13], we define the communication and interference among mesh routers respectively.

Definition 1(Communication): Given any two mesh routers u and v , if their distance $d(u,v) \le r$ and the NIC in each mesh router are tuned on the same channel, we say that there is a potential link between them.

Definition 2(Interference): Given any four mesh routers u , v , x , y , when $d(u,v) \le r$, $d(x,y) \le r$ and x is covered by R_i^u or y is covered by R_i^y (at the same time u is covered by R_i^x or v is covered by R_i^y); if the channel which x and y use is the same to that of the u and v , we say link u , v and link x , y are interfered with each other.

Using definition 1, we construct a connect graph of the wireless mesh network, denoted as $G(N,E)$ in which node $v \in V$ corresponds to a mesh router and $e \in E$ between node u and v corresponds to a potential link. In our connect graph, the meaning of a potential link is that mesh router u and v are within each other's transmission range. In figure1 there is a connect graph, of which nodes are represented as mesh routers, edges between nodes mean that there is a potential link.

In our work we use the distance information within mesh routers in physical topology to extend the multiple network interface cards connect graph. In the extended graph, it was called the distance to the first multiple network interface card connect graph (DMCG), denoted as $G'(N,L)$. To be specific, when some mesh routers are within each other's transmission ranges, we use the distance to measure the priority to

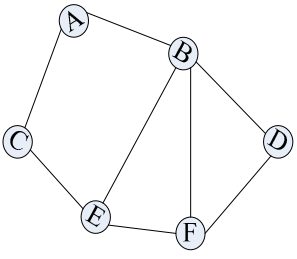


Fig. 1. Connection graph

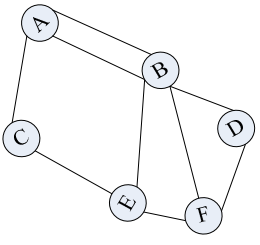


Fig. 2. Distance to the first multiple network interface card connection graph

connect. According to the distance and the number of a mesh router's network interface card we get a link with these network interface cards, in which some network interface card may be shared among links. In figure2, (mesh routers c and d are equipped with two network interface cards, others' equipped with three network interface cards), it was a DMCG converted from figure1.

Each network interface card in the mesh router has the priority to connect the least mesh router, until they have no other network interface cards. In G', $n \in N$ still corresponds to a mesh router but $l \in L$ corresponds to a link between network interface cards, sometimes the number of the links between mesh routers are more than that of network interface cards within a mesh router, so some network interface cards are shared by links. Note that multi-links using the same network interface card can't work concurrently, and a network interface card is assigned to only one channel, these links using the same network interface card should be assigned the same channel and work on different time slot. In figure2, we can see mesh router B has two network interface cards are shared, the links on these network interface cards should not work concurrently.

4 Channel Allocation Algorithm

In this section we present our novel channel assignment algorithm depend on the distance to the first multiple network interface card connection graph. In this algorithm our main purpose is to improve the utilization of channel resources, network interface cards and to avoid interference mostly. In the backbone of wireless mesh network, most mesh routers are stationary, so the distance between them is fixed. During the process of the router discovery and switches of channels among mesh routers, dynamic channel allocation algorithm may lead to the deafness problem,

synchronization problem or long channel switching delay[14][15], so we propose a static channel allocation algorithm, which is mainly used in the backbone of WMN. As we can see, the backbone of wireless mesh network mainly provides the Internet access for people, so most of the traffic is aggregated at the gateway nodes. In [9], we know that the flow structure in mesh routers is similar to a tree: gateway node is considered as the root, and mesh routers the leaf nodes, then most traffic of the leaf nodes are aggregated at root node. As the mesh router which is far away from gateway node is in the transmission process, then it will pass through more nodes, thereby leading to the more serious channel interference and network capacity loss; at last, in order to balance the overall network performance, we assign channels to the network interface cards by using depth-first traversal and make the mesh router, which is far away from gateway node, have the priority to get the channel. During the depth-first traversal process, our channel allocation algorithm depends on the distances among the mesh routers, which is called DDFT-CA (distance, depth-first traversal channel allocation).

Algorithm1 DDFT-CA Algorithm:

Input: the distance to the first multiple network interface cards connect graph

1: Depth-first traversal of this graph, get the depth-first traversal list L and the interference among mesh routers.

2: for all list $l \in L$, do

Choose one node

3: for all network interface cards which are equipped with mesh router do

If it has not have assigned channel, according to the interference information choose one orthogonal frequency channel which will not lead to interference and have not assigned to other network interface cards in the same mesh router. If the network interface card is connected with multi-links, assign same links the same channel

4: end for

5: end for

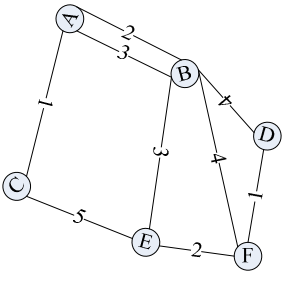


Fig. 3. Logical topology

With using DDFT-CA algorithm, we get figure3, in which there are five orthogonal frequency channels in total, from figure2. The label associated with each edge indicates the channels assigned to that link. From the figure3, we see some links share the NIC, such as the link between mesh routers A and B share the NIC with the link between mesh routers B and E.

5 Throughput Optimization in Wireless Mesh Network

We still need deal with the flow problem after the channels have been completely allocated, because sometimes the number of flows will compete network interface card or interfere with each other. So in this section, we study how to use the most possible network interface card scheduling to improve throughput and propose a linear optimization on mesh routers without gateway nodes. We assume that there are p flows through each mesh router v and that do not consider that whether the flows transmit into the router v or not; Each flow have a flow f^p and the p flows are denoted as the set of p_v. The total flow in mesh router v is:

$$r_v = \sum_{p \in p_v} f^p \tag{1}$$

Our optimization goal is:

$$\max \sum_{v \in V} r_v \tag{2}$$

Due to the limitation of network interface cards and the number of orthogonal frequency channels, all flow p may not pass through mesh router v at the same time. We define the follow notations:

$X^t_{p,k} = 1$, if flow p uses the k-th network interface card which is equipped with mesh router in slot t; t = 1, 2… T; where T is the period of the schedule. $X^t_{p,k} = 0$, Otherwise.

In order to maximize the overall network throughput; we should make each mesh router v satisfying the following constraints:

NIC-using constraints:

Since each mesh router is equipped with multiple network interface cards, if packets that belong to the same flow use more than one network interface card to transmit concurrently, this may cause packets to arrive out of order. To avoid this issue, only one NIC could be used by flow p, that is

$$\sum_{k \in K} X^t_{p,k} \le 1 \qquad \forall p,t \tag{3}$$

Note that when we add both sides of (3) for all slots and divide by T, then we would have:

$$\sum_{k \in K} \frac{\sum_{t \in T} X^t_{p,k}}{T} = \sum_{k \in K} \frac{f^p_k}{c_k} \le 1 \qquad \forall p \in p_v \tag{4}$$

In which f^p_k is the flow p on k-th network interface cards.

NIC constraints:

In each mesh router, it has equipped multiple network interface cards, but the network interface cards which the different flows using concurrently should less than the number of NIC which was equipped in mesh router. That is

$$\sum_{p \in P_v} \sum_{k \in K} \frac{f^p_k}{c_k} \le v_k \qquad \forall p \tag{5}$$

Where v_k is the number of network interface cards which are equipped with mesh router v .

In the above we use static algorithm assign channel to network interface cards, so in each mesh router the number of orthogonal frequency channels is no more than the number of each mesh router's NIC, then it is

$$\sum_{p \in P_v} \sum_{k \in K} \frac{f_k^p}{c_k} \le v_c \qquad \forall p \tag{6}$$

Where v_c is the number of orthogonal frequency channels in mesh router v .

NIC capacity:

Because each NIC capacity is a constant value, and each flow which use the NIC, it should less than the capacity, that is

$$f_k^p \le c_k \qquad \forall p \in p_v \tag{7}$$

Now our optimization formulates follows:

$$\max \sum_{v \in V} \sum_{p \in p_v} f^p$$

Subject to: (4) (6) (7).

6 Simulation

We consider a wireless mesh network with 10 mesh routers randomly located in 900*900 m^2 regions, in which each mesh router is equipped with 3 network interface cards and 6 orthogonal frequency channels are used. The transmission range and corresponding interference range of each mesh router are set to 250m and 500m respectively. Each connection is generated with a randomly chosen source-destination pair. First through definition 1 we construct the connection graph figure4, in which there are ten mesh routers, from the graph we can get the connection information of each mesh router.

After using definition 2 we get the depth-first traversal list "AJEBCIDHGF", and the interference information, then we uses the channel allocation algorithm which we proposed above for each mesh router's network interface card assign channel. In figure5 we get the logical topology, in which dashed line is the link that shares a network interface card with other link.

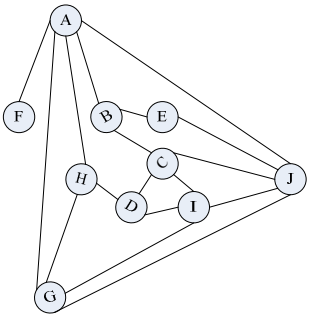


Fig. 4. Connection graph

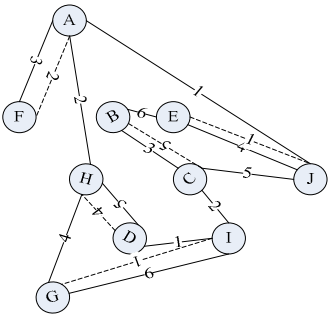


Fig. 5. Logical topology

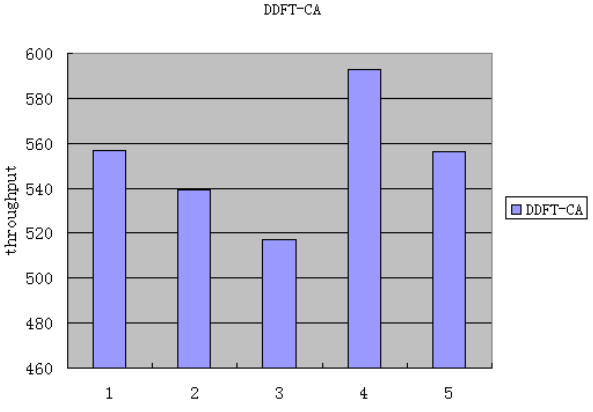


Fig. 6. The mean throughput

At last we solve the linear programming. After running, the results are shown in figure6. Through figure6 we can see, our scheme achieves a good performance at throughput.

7 Conclusion

In this paper, we have studied the network of web services. We joint channel allocation, interface assignment in multi-channel multiple network interface card wireless mesh networks, and proposed a static channel allocation algorithm and a linear programming for maximum throughput the overall wireless mesh network. Simulation results showed that our scheme achieves a good performance at throughput in wireless mesh network. Also it will support web services better.

References

1. Akyildiz, I.F., Wang, X., Wang, W.: Wireless mesh networks: a survey. Elsevier J. Computer Networks 47(4), 445–487 (2005)
2. IEEE 802.11 Working Group, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (1997)

3. IEEE 802.11a Working Group, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 1: High-speed Physical Layer in the 5 GHz band (1999)
4. Marina, M., Das, S.: A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks. In: Proc. Broadnet 2005 (October 2005)
5. Mohsenian-Rad, A.H., Wong, V.W.S.: Joint Logical Topology Design, Interface Assignment, Channel Allocation, and Routing for Multi-Channel Wireless Mesh Networks. IEEE Transactions on Wireless Communications 6(12) (December 2007)
6. Tang, J., Xue, G., Zhang, W.: Interference-aware topology control and QoS routing in multi-channel wireless mesh networks. In: Proc. ACM MobiHoc 2005 (May 2005)
7. Mohsenian-Rad, A.H., Wong, V.W.S.: Joint Channel Allocation, Interface Assignment and MAC Design for Multi-Channel Wireless Mesh Networks. In: Proc. IEEE INFOCOM 2007 (2007)
8. Alicherry, M., Bhatia, R., Li, L.E.: Joint Channel Assignment and Routing for Throughput Optimization in Multi-radio Wireless Mesh Networks. In: Proc. of Mobicom 2005 (2005)
9. Ramachandran, K., Almeroth, K., Belding-Royer, E., Buddhikot, M.: Interference-Aware Channel Assignment in Multi-Radio Wireless Mesh Networks. In: IEEE Infocom 2006 (2006) (to appear)
10. Tang, J., Xue, G., Zhang, W.: Cross-Layer Design for End-to-End Throughput and Fairness Enhancement in Multi-Channel Wireless Mesh Networks. IEEE Transactions on Wireless Communications 6(10) (October 2007)
11. Li Yu Wang, F., Li, X.-Y.: Gateway Placement for Throughput Optimization in Wireless Mesh Networks. In: Proc. IEEE ICC (2007)
12. Tang, J., Xue, G., Zhang, W.: Maximum Throughput and Fair Bandwidth Allocation in Multi-Channel Wireless Mesh. In: Proc. of IEEE INFOCOM (2006)
13. Raniwala, A., Chiueh, T.: Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In: Proc. IEEE INFOCOM, pp. 2223–2234 (March 2005)
14. Kyasanur, P., Vaidya, N.H.: Routing and interface assignment in multi-channel multi-interface wireless networks. In: Proceedings of WCNC 2005, pp. 2051–2056 (2005)
15. So, J., Vaidya, N.H.: Routing and channel assignment in multi-channel multi-hop wireless networks with single network interface, UIUC Technical Report (2005), `http://www.crhc.uiuc.edu/wireless/groupPubs.html`

Author Name Disambiguation for Citations on the Deep Web

Rui Zhang, Derong Shen, Yue Kou, and Tiezheng Nie

College of Information Science and Engineering, Northeastern University,
110004 Shenyang, China
anna_zhangrui@126.com,
{shenderong,kouyue,nietiezheng}@ise.neu.edu.cn

Abstract. Name ambiguity is a critical problem in many applications, in particular in the online bibliographic digital libraries. Although several clustering-based methods have been proposed, the problem still presents to be a big challenge for both data integration and cleaning process. In this paper, we present a complementary study to the author name disambiguation from another point of view. We focus on the common names, especially non-canonical ones. We propose an approach of automatic access to authors' personal information over Deep Web, and compute the similarity of every two citations according to the following features: co-author name, author's affiliation, e-mail address and title. Then we employ Affinity Propagation clustering algorithm to attributing the resembling citations to the proper authors. We conducted experiments based on five data sources: DBLP, CiteSeer, IEEE, ACM and Springer LINK. Experiments results show that significant improvements can be obtained by using the proposed approach.

Keywords: Name Disambiguation, AP Clustering, Similarity.

1 Introduction

In scientific bibliography systems, Web page has become the main carrier for publications. There are several digital libraries such as DBLP, CiteSeer, IEEE, ACM and Springer LINK, collecting a large number of publication records, named as "citations" in this paper, and providing a bibliography search service for academic community. To have a consistent, accurate and up-to-date citation dataset is a very important task for all digital libraries. However, due to the incomplete information of citations and different authors sharing the same name, digital libraries cannot always correctly map citations to real authors. The name disambiguation problem [1] can be formalized as: given a list of citations with all sharing an identical author name but might actually referring to different persons, the task is to assign the citations to several different clusters, each of which contains resembling citations written by the same author.

Meng et al. pointed that there exists three major Entity-Name problems [2]: name sharing problem, name variant problem and name mixing problem. The above three situations are more common for citation field with several data sources. Through

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 198–209, 2010.

crawling on the different data sources, we can obtain tremendous, heterogeneous and redundant citation records. Undoubtedly, it makes more difficult to solve the name disambiguation problem. For example, there are 16 different authors named "Wei Wang" in DBLP, but only 2 in CiteSeer. Furthermore, we have observed that name disambiguation for non-canonical authors doesn't do well in DBLP, e.g. DBLP considers that there exists one author named "Qin Zhang" with 61 publications but actually seven authors. We should deal with the name disambiguation not only in every single data source, but also among the different data sources.

We encountered the following three challenges during our study. (1) The information we can obtain from the citation records is limited, mainly co-author names, title and publication venue, which have a little contribution for name disambiguation. (2) Both the change of authors' affiliation and their research interests may lead to the change of co-authors correspondingly. That makes difficult to integrate citations before and after changes. (3) Focusing on name disambiguation problem in citations over Deep Web, we must deal with the tremendous citation records from heterogeneous data sources, such as data clean and information integration.

In the previous work, most of methods achieve the name disambiguation goal by calculating the similarity between citation records according to the usual features, such as title, co-authors, publication venue, etc. In order to improve the performance, we should acquire the special personal information of an author, such as author's affiliation, e-mail address, telephone number, etc. We have observed that the author's affiliation and e-mail address are extremely helpful to name disambiguation problem in our study, especially to non-canonical author names.

Our contributions in this paper include: (1) Present a complementary study to the author name disambiguation problem from another point of view, namely, using the personal information by crawling on the Deep Web. (2) Identify and deal with the redundant citation records mainly according to the Digital Object Identifier. (3) Take advantage of a few but useful features, i.e. author's affiliation, co-author names and citation's title, and employ Affinity Propagation clustering algorithm to disambiguate.

The remainder of this paper is organized as follows. Section 2 reviews the related work, and Section 3 describes the proposed disambiguation approach. The experiment results are presented in Section 4. Finally, we conclude the paper in Section 5.

2 Related Work

A great deal of research has focused on the name disambiguation problem in different domains. Broadly speaking, there are four different directions: supervised learning, unsupervised learning, semi-supervised learning and topic-based modeling [3].

[4] proposed two supervised methods, Naïve Bayes probability model and Support Vector Machines (SVM). This method can be quite accurate and reliable, however, it relies heavily on the quality of the training data which is difficult to obtain. In addition, the premise of using the Naïve Bayes model is that the features are independent of each other, which do not meet the actual situation.

For the unsupervised approaches, training data and tagging are not required. [5] proposed an unsupervised learning approach using K-way spectral clustering method to solve the name disambiguation problem. The method is fast and can get a global

optimal solution, but it requires the clusters number given and depends heavily on the initial partition and the order of processing each data point. Furthermore, this clustering method may not work well for large datasets.

Semi-supervised learning method can be seen as a compromise of the two above methods. [6] proposed a constraint-based probabilistic model for semi-supervised name disambiguation. They formalize the problem in a constraint based probabilistic framework using Hidden Markov Random Fields (HMRF), and then define six types of constraints and employ EM algorithm to learn different distance metric for different authors. The drawback is that the clusters number is required and the result is locally optimal solution.

Recently, researchers prefer to use topic-based model to solve the name disambiguation problem. "Topic" can be considered as global information or expert knowledge. Song et al. introduced two kinds of topic-based model: Probabilistic Latent Semantic Analysis (PLSA) and Latent Dirichlet Allocation (LDA) in [7]. To enhance the LDA model, [2] proposed a generative latent topic model (LDA-dual model) witch involves both author names and words. By comparing the topic similarity, it achieved better experimental results over the other approaches. Unfortunately, this approach is difficult to implement because it requires too much labor-intensive preprocessing (e.g. manually label the training data) and relies on some special assumptions (e.g. the cluster number is given).

3 Proposed Approach

3.1 Overview

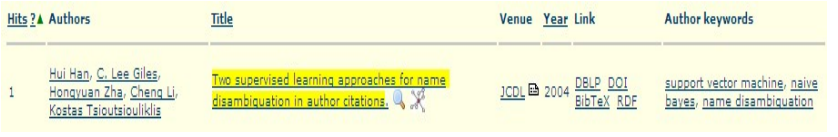
Given a large number of citation records with the same author name (including variant names) from several different data sources, the procedure of name disambiguation can be summarized as the following algorithm.

Author Name Disambiguation Algorithm on the Deep Web
Input: a list of citation records sharing the same author name from different data sources **Output:** a list of clusters with distinct author. **Process:** 1. Duplicate identification and mergence according to DOI 2. Building wrappers for each data source and extracting author personal information, and then calculate the total similarity of every two citation records according to the pairs-wise similarity metrics. 3. Cluster citation records using AP algorithm. 4. Deal with the remainder records without DOI according to the CoauSet and TitleSet.

where, the CoauSet and TitleSet represent the Sets of all co-authors and citation titles in each cluster, respectively.

3.2 Duplicate Identification and Mergence

In different data sources, the format of citation records is different. We take the paper "Two Supervised Learning Approaches for Name Disambiguation in Author Citations" for example, shown in Fig. 1 (a) (b) (c). We can retrieval the same citation



Hits ?▲	Authors	Title	Venue	Year	Link	Author keywords
1	Hui Han, C. Lee Giles, Hongyuan Zha, Cheng Li, Kostas Tsioutsiouliklis	Two supervised learning approaches for name disambiguation in author citations	JCDL	2004	DBLP DOI BibTeX RDF	support vector machine, naive bayes, name disambiguation

(a) DBLP

Two supervised learning approaches for name disambiguation in author citations

Full text PDF (144 KB)

Source International Conference on Digital Libraries archive
Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries table of contents
Tucson, AZ, USA
SESSION: Mining and disambiguating names table of contents
Pages: 296 - 305
Year of Publication: 2004
ISBN:1-58113-832-6

Authors Hui Han The Pennsylvania State University, University Park, PA
Lee Giles The Pennsylvania State University, University Park, PA
Hongyuan Zha The Pennsylvania State University, University Park, PA
Cheng Li Harvard School of Public Health, Boston, MA
Kostas Tsioutsiouliklis NEC Laboratories America, Princeton, NJ

Sponsors ACM: Association for Computing Machinery
SIGIR: ACM Special Interest Group on Information Retrieval
SIGWEB: ACM Special Interest Group on Hypertext, Hypermedia, and Web

Publisher ACM New York, NY, USA

(b) ACM

Two supervised learning approaches for name disambiguation in author citations

Han, H. Giles, L. Zha, H. Li, C. Tsioutsiouliklis, K.
Dept. of Comput. Sci. & Eng., Pennsylvania State Univ., University Park, PA, USA
This paper appears in: Digital Libraries, 2004. Proceedings of the 2004 Joint ACM/IEEE Conference on
Publication Date: 7-11 June 2004
On page(s): 296 - 305
ISSN:
ISBN: 1-58113-832-6
INSPEC Accession Number: 8179682
Digital Object Identifier: 10.1109/JCDL.2004.1336139
Current Version Published: 2004-09-27

(c) IEEE

Fig. 1. The citation record formats in DBLP, ACM, and IEEE

record with different formats from DBLP, ACM, and IEEE, not CiteSeer and Springer LINK, which prove not only the redundancy and heterogeneity of citation records, but also the necessity of our name disambiguation work.

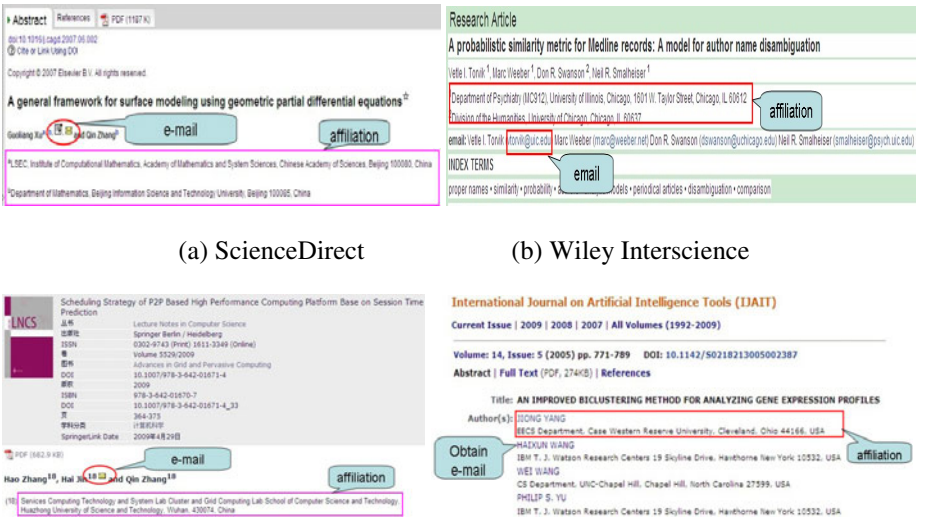
DOI, short for Digital Object Identifier, is a unique identifier to a published work [8]. It differs from commonly used Internet pointers to material such as the URL. DOI identifies an object as a first-class entity, not simply the place where the object is located. That is to say, two instances of the same object would have the same DOI name. According to the uniqueness of DOI, we can identify whether the citation records are duplicate. When a citation's DOI is missed, we have to compare its title and authors with others.

3.3 Information Extraction

To distinguish the same name authors, we need to obtain the authors' specific personal information. Concluding observations and tests, we can extract the author information through the following three approaches.

Author's Homepage. We can find an author's homepage by using Google API and heuristic rules or extracting from DBLP. In DBLP, homepage information was done manually with Google and stored in the url-field of person records, e.g. http://dblp.uni-trier.de/rec/bibtex/homepages/l/MichaelLey.xml. However, only a few canonical authors have homepages, which doesn't work for ordinary authors.

Full Contents of Publication. CiteSeer provides the full contents of most publications (in PDF format). By parsing the full contents, we can obtain the author names, affiliations, and e-mails. However, with the time and space complexity analysis, it's not a good idea to achieve the information.



(a) ScienceDirect (b) Wiley Interscience

(c) Springer LINK (d) WorldSciNet

Fig. 2. The citation records' Html pages in different Digital Library

Web HTML Pages provided by digital libraries. We can obtain author's personal information by parsing the records' Html pages provided in digital libraries, such as ACM, Springer LINK, ScienceDirect, WorldSciNet, Wiley Interscience, etc. Fig. 2 shows the examples

We define a uniform global schema, i.e. a union set of attributes among data sources, including author names, author's affiliation, author's e-mail address, DOI and title. Then a wrapper is built for each data source and used for extracting information from HTML pages. Compared to the extraction from the Web, the wrappers can obtain more accurate and targeted information. Experiments have proved that the method work out fine with much lower cost.

3.4 Pairs-Wise Similarity Metrics

In this paper, we describe a citation using four types of features: author's affiliation, co-author name, e-mail address and title. All of the feature information has been obtained by wrappers. We calculate the similarity scores between the corresponding features of every two citations by using different types of similarity metrics.

Author's Affiliation Feature. The author's affiliation can be described with various formats in web, including abbreviations, brief write, or key words with different order. For instance, "Computer Science Department" and "Department of Computer-Science" are the same in semantic but different in literal string. This situation is bad-for the accuracy. We adopt a new string matching algorithm named as Adjusted-Edit distance [9]. It is an improvement based on edit distance.

Co-author Feature. We use Dice's coefficient to compute co-author name similarity for two citations. Considering the diversity of author names, we adopt a small improvement proposed in [2] when applying Dice's coefficient, in which the name distance was denoted by δ and the Dice's coefficient is defined as Formula 1.

$$s = \frac{2|X \cap Y| + \delta}{|X| + |Y|} \tag{1}$$

Where, X and Y are two co-author names sets. For example, X = {Weiyi Meng, Clement T. Yu} and Y = {Hongkun Zhao, Weiyi Meng, Clement Yu}. Considering similarity of Clement T. Yu and Clement Yu, we compute Dice's coefficient as: (2*1.5) / (2+3) = 0.6; While standard computation is (2*1) / (2+3) = 0.4.

Title Feature. The title similarity of every two citations is defined by using LCS algorithm.

3.5 Similarity Calculation

In order to calculate the total similarity between every two citations more accurately and efficiently, we get the similarity by the following steps.

Constraint Rules Building. Given two authors a1, a2 sharing the same name, we build two kinds of constraint rules, named as Hard Rule and Soft Rule, according to heuristics and sequential covering algorithm.

Hard Rule-1: If the two authors a1, a2 have an identical email address, we consider that the names a1 and a2 must correspond to the same author entity.

$$(a1.email = a2.email) \Rightarrow a1 = a2$$

Hard Rule-2: If all the features of the two authors a1, a2 are different, we consider that the two names must not correspond to the same author entity.

$$(a1.email \neq a2.email) \wedge (a1.coau \cap a2.coau = \varnothing) \wedge (a1.affi \neq a2.affi) \Rightarrow a1 \neq a2$$

Soft Rule: If the two authors have the several same co-authors, or their affiliations are exactly similar, we can consider the two authors may be the identical entity.

$$(a1.coau \cap a2.coau \neq \varnothing) \vee (a1.affi = a2.affi) \rightarrow a1 = a2$$

Comparing the features of every two citations c1, c2, if they meet Hard rule-1, then the total similarity is defined as Sim (c1, c2) = 1; if they meet Hard rule-2, then Sim (c1, c2) = 0; if they meet the Soft rules, then the total similarity is defined as the sum of three feature similarity, shown as Formula 2.

Weights of Features. In this paper, we adopt the entropy method to determine the weights of features while computing the total similarity.

Entropy method is used for measuring the information of data and determining weight according to the degree of divergence among the evaluation features. The degree of divergence is more widely, the entropy value is smaller, and the weights should be larger. When the values of a feature are the same, the entropy will be the largest which means the feature is helpless and should be removed. For each pair of citations, we compute similarity values of five features: author's affiliation, co-author name, title, e-mail address and publication venue. Experiments shows the first three

features make contributions to get a more accurate similarity sum and their weight values are 0.52, 0.36 and 0.12 separately.

$$S_{total}(c1,c2)=\begin{cases}1, & if\ (c1,c2)\in HardRule-1;\\ \omega_1 S_{affi}(c1,c2)+\omega_2 S_{coau}(c1,c2)+\omega_3 S_{title}(c1,c2), & if\ (c1,c2)\in SoftRule;\\ 0, & if\ (c1,c2)\in HardRule-2\end{cases} \tag{2}$$

3.6 Name Disambiguation

Recall that our goal is to group a set of resembling citations from different data sources into clusters so that we can get a mapping between the citations and the real authors. The procedure of name disambiguation will be discussed in this section.

Clustering Strategy. We adopt a powerful new clustering algorithm named Affinity Propagation [10] (AP for short) in this paper.

AP clustering algorithm is an efficient and fast clustering algorithm, especially in the case of large number of clusters, and has three advantages: speed, general applicability and good performance. AP works based on similarities between pairs of data points, and simultaneously considers all the data points as potential cluster centers, called exemplars. There are two kinds of message: responsibility and availability, exchanged between data points, and each takes into account a different kind of competition. Messages can be combined at any stage to decide which points are exemplars and for every other point, which exemplar it belongs to. Based on evidence accumulation, AP searches for clusters through an iterative process until a high-quality set of exemplars and corresponding clusters emerges.

Name Disambiguation for Citations. According to our Disambiguation Algorithm, the citations with DOI link can be attributed to the proper authors in step 3. Now we obtain clusters and remaining citation records without DOI link. The clusters can be seen as standard sets. For the remaining records, we try to attribute them to proper clusters according to co-author names and titles. First, we sum up the variant names of author, all of his/her co-authors (represented with a Set named CoauSet), and citation titles (represented with TitleSet) in each cluster. Second, we compare the every remaining citation record with the clusters according to the author name, including the variant names. If co-author name of the compared record can be found in the CoauSet, we can assign the record to the cluster; if the citation's author is just one, we have to compare its title with TitleSet by using title similarity. In the end, we obtain the citation record clusters attributed to distinct author, that's to say, we achieve the goal of name disambiguation in citation field.

Note that there are two special situations, in which it is difficult to identify the authors with the same name even for human beings. First, for the citations with a single author, name disambiguation work mainly depends on the similarity of author's email and affiliation; Second, for two different author entities with the same name, the same affiliation and sharing common co-authors, we make a judgement according to their emails. However, if the email information is missed, our method is helpless. We will focus on the above two situations in the future work.

4 Experiment Design and Results

4.1 Datasets

In our experiments, we use five data sources to simulate the citation field, i.e. ACM, DBLP, CiteSeer, IEEE and Springer LINK, which contains about 2.5M papers. In order to compare with other methods, we select ten common names shown in Table 1, together with the number of authors and number of citations. For each name, we have checked manually according to authors' homepages or affiliations shown on the papers. Note that authors with no more than 2 papers have been removed.

Table 1. Statistics of the data set

Name	Authors	Citations	Name	Authors	Citations
Jim Smith	5	58	Qin Zhang	7	50
Wei Wang	42	278	Tao Wang	18	112
Jun Zhang	22	286	Michael Wagner	11	87
Ying Liu	24	143	Lei Wang	50	211
Hui Fang	6	37	Rakesh Kumar	2	36

4.2 Pre-processing

In this paper, we think that the author name should not be abbreviated beyond recognition and spelling errors. With regard to the format of citation records, it is against to our method that the author's name is usually abbreviated. Therefore, we need to do pre-processing work against to the above two situations.

For every citation record, we can obtain author full names by wrappers. During pre-processing work, we pay attention to the format of author full names. For instance, author name is represented as "first name + last name" in most of data sources, but "last name + first name" in ACM. In our method, we consider the two formats represent the same one, e.g. both "Qin Zhang" and "Zhang Qin" represents one author name, namely "Qin Zhang".

4.3 Measuring Quality

Traditionally, the performance quality is measured using precision, recall, and F1-measure. Although precision and recall capture the quality of results well most of the time, it is now often argued that such measures are not fully adequate for the problem of entity resolution, because they are representation-centric measures, whereas entity-centric measures are often preferred for entity resolution [11]. Thus, we will also use another well-known measure: purity, inverse purity and their harmonic mean Fp-measure. Purity measures the homogeneity of the clusters when evaluated against pre-classification, while inverse purity measures the stability of the classes when split into clusters. Fp-measure is the harmonic mean of the purity and inverse purity.

Definition 1. We first define two sets: Sa= {(i, j)| citation i and j are merged by the algorithm, and i≠j}, and Sr= {(i, j)| citation i and j are merged in reality, and i≠j}. Then we define precision and recall as follows**:**

$$precision = \frac{|S_a \cap S_r|}{|S_a|}, \quad recall = \frac{|S_a \cap S_r|}{|S_r|}, \quad F1-measure = \frac{2 \times precision \times recall}{precision + recall} \tag{3}$$

Definition 2. The purity measure is based on the precision. The precision of a cluster p∈P for a given category l∈L is given by

$$precision = \frac{|p \cap l|}{|p|}, \tag{4}$$

The overall value for purity is computed by taking the weighted average of maximal precision values:

$$purity(P,L) = \sum_{p \in P} \frac{|p|}{|D|} \max_{l \in L} precision(p,l),$$

$$inversePurity(P,L) = \sum_{l \in L} \frac{|l|}{|D|} \max_{p \in P} precision(l,p) \tag{5}$$

$$Fp = \frac{2 \times purity(P,L) \times inversePurity(P,L)}{purity(P,L) + inversePurity(P,L)}$$

4.4 Experiments in Citation Field

We deal with each dataset using the Author Name Disambiguation Algorithm and obtain the author numbers and entries numbers in each dataset. We measure the performance quality by using both F1 and Fp measures. Table 2 shows the performance of our method for each name. In general, our method successfully group citations with high accuracy. In some cases, citations to one author are divided into multiple groups if both the intersection sets of affiliation and coauthor names are empty. For example, 18 citations to "Ying Liu" whose affiliation is "IBM China Research Laboratory" are divided into two groups, which lead to low recall. Table 3 shows two examples of the clustering results with our method, and proves that not only name sharing problem but also name variant problem can be solved by our method. From Table 3(a), we can see that both "Jim Q. Smith" and "Jim E. Smith" are the variants of the author name "Jim Smith", and they can be identified well. Table 3(b) shows part of clustering results of author name "Wei Wang", which prove that our method can perform well even if the affiliation or co-authors have changed.

In this paper, we also use the first author's own name "Rui Zhang" to test. With our name disambiguation algorithm, we can distinguish 12 different author entities sharing this common name (precision: 0.931, recall: 0.862).

Table 2. Accuracy for distinguishing citations

Name	Precision	recall	F1	Fp	Name	Precision	recall	F1	Fp
Jim Smith	0.953	0.932	0.946	0.968	Qin Zhang	0.927	0.912	0.923	0.959
Wei Wang	0.972	0.938	0.966	0.979	Tao Wang	0.892	0.885	0.888	0.925
Jun Zhang	0.987	0.963	0.975	0.988	Michael Wagner	0.988	0.912	0.956	0.973
Ying Liu	0.934	0.687	0.867	0.902	Lei Wang	0.939	0.921	0.935	0.966
Hui Fang	1.0	1.0	1.0	1.0	Rakesh Kumar	1.0	1.0	1.0	1.0

Table 3. Detailed statistics of two author names

(a) Jim Smith

Author Names	Affiliations	Citations
Jim Smith	University of Newcastle, UK	24
Jim Smith	University of Melbourne, Australia	3
Jim Smith/Jim Q. Smith	University of Warwick Coventry, UK	8
Jim Smith/Jim E. Smith	University of the West of England, Bristol, UK	19
Jim Highsmith	Cutter Consortium	4

(b) Wei Wang

Author Names	Affiliations	Citations
Wei Wang	The Australian National University; Nanyang Technological University	24
Wei Wang	University of Electro-Communications	11
Wei Wang	National University of Defense Technology	6
Wei Wang	Chinese Academy of Sciences	9
…	…	…

Single-feature VS Multi-features. In this part, we do two sets of experiments to find the contribution of author's personal information in our method.

In one of the experiments, we only take coauthor feature into account when we compute the similarity of every two citations. In the other, we use three features: coauthor, author's affiliation and e-mail address. As shown in Fig. 3, the performance of author name disambiguation is better by using three features. The author's personal information is helpful to name disambiguation.

Hierarchal Agglomerative Clustering VS AP Clustering. Hierarchal Agglomerative Clustering (HAC for short) is a state-of-the-art clustering algorithm. We use it as a baseline method for comparison. As shown in Fig. 4, the performance of AP clustering is better than HAC from three aspects: F1, Fp and Time Cost. On the average, the F1 is improved by about 4.7% and the Fp is improved by about 5.4% with AP clustering. The time cost of AP clustering is almost one percent of that of HAC.

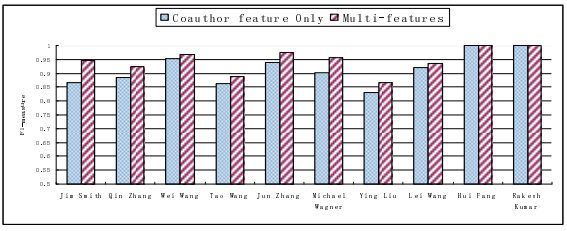


Fig. 3. The comparison of F1-measure between Single-feature and Multi-feature

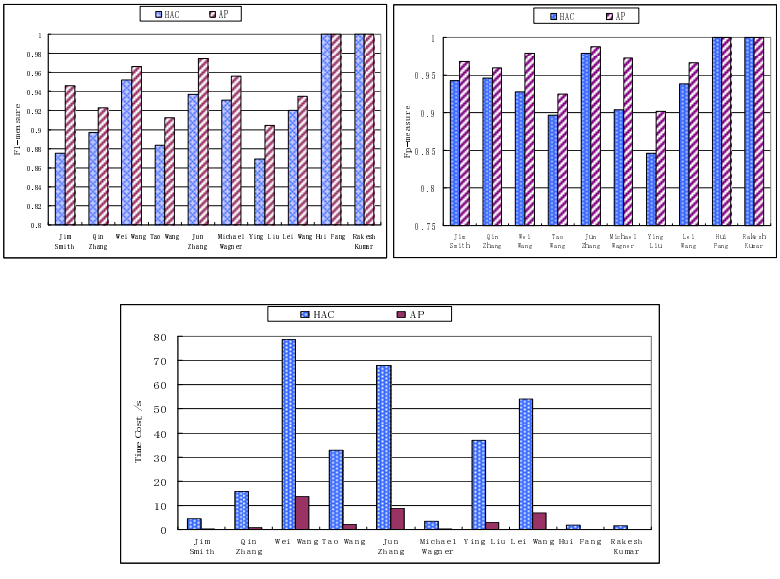


Fig. 4. The comparison of F1, Fp and Time Cost between HAC and AP

5 Conclusions

In this paper we extended the range of the author name disambiguation problem to the whole citation field. Considering the data's heterogeneity and redundancy among different data sources, we proposed an approach of automatic access to authors' personal information over Deep Web, calculated pairs-wise similarity and adopted Affinity Propagation clustering algorithm. Experiments show that significant improvements can be obtained by using the proposed approach. We can not only effectively solve the problem of author's affiliation or co-author change, but also the name sharing and name variant problem.

Acknowledgments. This work is supported by the National Science Foundation (60673139, 60973021), the National High-Tech Development Program (2008AA01Z146).

References

1. Wang, F., Li, J., Tang, J., Zhang, J., Wang, K.: Name Disambiguation Using Atomic Clusters. In: WAIM, pp. 357–364. IEEE, New York (2008)
2. Shu, L., Long, B., Meng, W.: A Latent Topic Model for Complete Entity Resolution. In: ICDE, pp. 880–891. IEEE, New York (2009)
3. Zhu, J., Zhou, X., Fung, G.P.C.: A Term-Based Driven Clustering Approach for Name Disambiguation. In: Li, Q., Feng, L., Pei, J., Wang, S.X., Zhou, X., Zhu, Q.-M. (eds.) APWeb/WAIM 2009. LNCS, vol. 5446, pp. 320–331. Springer, Heidelberg (2009)
4. Han, H., Giles, C.L., Zha, H., Li, C., Tsioutsiouliklis, K.: Two supervised learning approaches for name disambiguation in author citations. In: JCDL, pp. 296–305. ACM, New York (2004)
5. Han, H., Zha, H., Giles, C.L.: Name disambiguation in author citations using a K-way spectral clustering method. In: JCDL, pp. 334–343. ACM, New York (2005)
6. Zhang, D., Tang, J., Li, J., Wang, K.: A constraint-based probabilistic framework for name disambiguation. In: CIKM, pp. 1019–1022. ACM, New York (2007)
7. Song, Y., Huang, J., Councill, I.G., Li, J., Giles, C.L.: Generative models for name disambiguation. In: WWW, pp. 1163–1164. ACM, New York (2007)
8. `http://en.wikipedia.org/wiki/Wikipedia:Digital_Object_Identifier`
9. Zhu, M.D., Shen, D.R., Kou, Y., Nie, T.Z., Yu, G.: A Model of Identifying Duplicate Records for Deep Web Environment. Journal of Computer Research and Development 46I(suppl.), 14–21 (2009)
10. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science 315(16), 972–976 (2007), `http://www.psi.toronto.edu/affinitypropagation`
11. Chen, Z., Kalashnikov, D.V., Mehrotra, S.: Adaptive graphical approach to entity resolution. In: JCDL, pp. 204–213. ACM, New York (2007)
12. Masada, T., Takasu, A., Adachi, J.: Citation data clustering for author name disambiguation. In: Infoscale, p. 62. ACM, New York (2007)
13. Tang, J., Zhang, J., Zhang, D., Li, J.: A unified framework for name disambiguation. In: WWW, pp. 1205–1206. ACM, New York (2008)
14. Fan, X., Wang, J., Lv, B., Zhou, L., Hu, W.: GHOST: an effective graph-based framework for name distinction. In: CIKM, pp. 1449–1450. ACM, New York (2008)

Approximate Content Summary for Database Selection in Deep Web Data Integration

Fangjiao Jiang[1], Yukun Li[2], Jiping Zhao[1], and Nan Yang[2]

[1] Institute of Intelligent Information Processing, Xuzhou Normal University, Jiangsu, China
jiangfj@gmail.com, zhaojp@xznu.edu.cn
[2] School of Information, Renmin Universityof china
{liyukun,yangnan}@ruc.edu.cn

Abstract. In Deep Web data integration, the metaquerier provides a unified interface for each domain, which can dispatch the user query to the most relevant Web databases. Traditional database selection algorithms are often based on content summaries. However, many web-accessible databases are uncooperative. The only way of accessing the contents of these databases is via querying. In this paper, we propose an approximate content summary approach for database selection. Furthermore, the real-life databases are not always static and, accordingly, the statistical content summary needs to be updated periodically to reflect database content changes. Therefore, we also propose a survival function approach to give appropriate schedule to regenerate approximate content summary. We conduct extensive experiments to illustrate the accuracy and efficiency of our techniques.

Keywords: Database Selection, Approximate Content Summary, Survival Function.

1 Introduction

A large portion of data available on the web is "hidden" behind search forms, which is so called "Deep Web"[1]. It is observed that the topics of the Deep Web cover all the domains existing in the real world. Specifically, it is estimated that the size of the Deep Web is more than 500 times of *Surface Web*. However, the information in the Deep Web cannot be effectively found by the traditional search engines, such as, Google, Yahoo, etc. Therefore, it is essential to construct a new system to help the user find their interesting information. The metaquerier is the exact system which is one way to provide one-stop access to the information in the Deep Web. The metaquerier provides a unified interface for each domain. When a user query is submitted to the unified interface, it will dispatch the query to the most relevant Web databases. When the results retrieved are returned back from the Web databases, it will merge them together and present in a legible style to the user.

Due to the features of huge-scale, heterogeneity and autonomy, there are several serious challenges to implement an effective and efficient Metaquerier, e.g., database selection, entity identification, result merging, etc. In this paper, we address the problem of

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 210–221, 2010.

database selection, i.e., how to automatically select the most relevant ones from hundreds or thousands Web databases in one domain to reduce the communication cost?

For the cooperative text databases, because the statistics of content summaries that characterize each database's contents can be accessed easily, database selection algorithms are often based on theses content summaries. However, many web-accessible databases are uncooperative. They do not report the metadata about their databases. The only way of accessing the contents of these databases is via querying. A naïve approach obtaining content summaries is to crawl all the data of these databases. Unfortunately, most of databases forbid all the data to be crawled. In fact, even if the crawling is allowed, the communication cost is incredible. Therefore, accessing random sample by submitting queries and then analyzing approximate content summaries will be a perspective approach. As is well known, the hidden Web databases are classified into Web text databases and Web structured databases. For the Web text databases, the queries only contain some keywords. In contrast, for the Web structured databases, the queries have to contain several attributes and their values, which are more complex than those of Web text databases. Therefore, the more difficulty is how to issue the appropriate queries for accessing a random sample for Web structured database.

Furthermore, it's worth noticing that real-life databases are not always static and, accordingly, the statistical summaries that describe their contents need to be updated periodically to reflect database content change. Defining appropriate schedules for making the database approximate content summaries up to date and avoiding unnecessary summaries regeneration at the same time is a challenging task.

So there are two main objectives that our algorithm seeks to achieve:

- ***How to access high-quality sample for approximate content summaries?*** Due to the restricted nature of the interface, it is challenging to produce samples that are truly uniform. Consequently, the task is to produce samples that have small skew, i.e., samples that deviate as little as possible from uniform distribution.
- ***When to resample to make approximate content summaries up to date?*** Because the rate of change of the database contents might vary drastically from database to database, it is a challenging task to give appropriate schedule to regenerate approximate content summaries.

In this paper, we exploit an attribute-correlation-based sampling approach for constructing approximate content summary. We then resort to the field of statistics named "survival function" to develop summary update strategies. Our approaches attempt to contact the Web databases and update the content summaries only when needed, thus minimizing the communication with the Web databases. In brief, the main contributions are as follows:

-- We propose an attribute-correlation-based sampling approach for constructing approximate content summary.
-- We propose a survival function for predicting the time to regenerate new approximate content summary.
-- We present an extensive experimental result for our techniques.

The rest of this paper is organized as follows. In section 2 we present an approximate content summary for Web structure database. Section 3 is devoted to exploiting attribute-correlation-based sampling algorithm to access the approximate random attribute-level sample. In section 4 we present survival function to predict the schedule of regenerating database content summary. Section 5 reports the detailed experimental results. Section 6 is the related work. We conclude in Section 7.

2 Preliminaries

Content summary is widely used in database selection for cooperative text database. In this section, we define the notion *Content Summary* and *Approximate Content summary* for Web structure database. And, we also give our observation about database content's changes.

2.1 Approximate Content Summary

Definition 1. *The **Content Summary C(D)** of a database D consists of:*
—The actual number of tuples in D, |D|, and
—For each attribute A_i, the distribution of its values a_{ij} $d(a_{ij},A_i)$.

Table 1. The fragment of content summaries of database D_1 and D_2

D_1	$\|D_1\|$=230,900		
	A_i	a_{ij}	$d(a_{ij},A_i)$
	Title	algorithm	29,500
		mining	1,250
	Year	2009	6,000
		1995	1,500

D_2	$\|D_2\|$=20,500		
	A_i	a_{ij}	$d(a_{ij},A_i)$
	Title	algorithm	2,100
		mining	1,260
	Year	2009	370
		1995	2,700

From Table 1, we can know that there are much more tuples contain "algorithm" on attribute *Title* in database D_1 than those in database D_2. And we can also know that the published *Year* in D_1 should be later than that in D_2. However, content summary is hardly used for uncooperative structured database. So we define another notion *approximate content summary* which is based on samples as follows.

Definition 2. *An **Approximate Content Summary** $\hat{C}(D)$ of a database D consists of:*

—An estimate $|\hat{D}|$ of the number of tuples in D, and

—For each attribute A_i, an estimate $\hat{d}(a_{ij},A_i)$ of the distribution $d(a_{ij},A_i)$.

Example 1. *Consider the query [Title contains algorithm] and two databases D_1 and D_2. Based on the approximate content summaries of these databases (Table 2), a database selection algorithm may infer that D_1 is a promising database for the query, since more tuples satisfy the condition of the query in database D_1. In contrast, D_2 will probably be less relevant to this query.*

Table 2. The fragment of approximate content summaries of database D_1 and D_2

D_1	$\lvert \hat{D}_1 \rvert = 231{,}750$		
	A_i	a_{ij}	$\hat{d}(a_{ij}, A_i)$
	Title	algorithm	29,000
		mining	1,200
	Year	2009	5,000
		1995	1,000

D_2	$\lvert \hat{D}_2 \rvert = 20{,}830$		
	A_i	a_{ij}	$\hat{d}(a_{ij}, A_i)$
	Title	algorithm	2,000
		mining	1,220
	Year	2009	390
		1995	1,500

Example 2. *Consider another query [Title contains mining] and two databases D_1 and D_2. The number of tuples satisfying this query condition in D_1 is nearly equal to that in D_2 (Table 2). Given that the size of database D_2 is much smaller than that of D_1, D_2 is probably more focused on mining than D_1. Therefore, a database selection algorithm may infer that D_2 is a promising database for the query.*

So the database selection component of a metaquerier can make the selection decisions using these approximate content summaries, i.e., issue a query only to a relatively small number of databases that are relevant to the query for efficiency.

2.2 Database Content Changes

Given an extensive study on the contents of some real web databases, we discover that they are not always static but changeable over a period of time. For example, from 2000 to 2008, the number of tuples on mining, i.e., *Title* contains *mining*, (Microsoft Academic Search) increases with years. But, in 2009, the number of tuples satisfying the same condition decreases greatly. In another website (amazon.com), the changes of different attribute values, e.g., *XP, windows 7, vista, UNIX* of attribute *Title*, are much larger year by year. Specially, they hardly have any regularity.

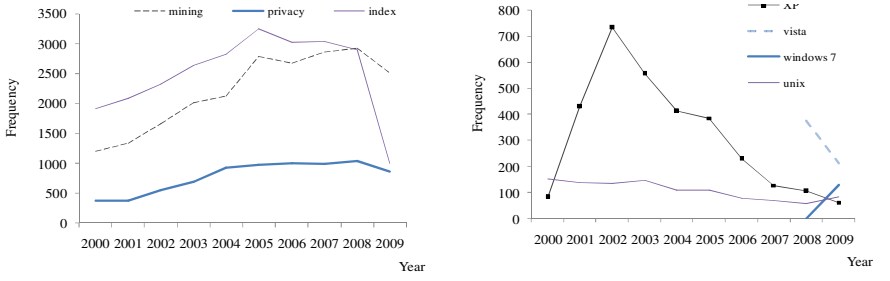


Fig. 1. The change of database content
(http://academic.research.microsoft.com/ and http://www.amazon.com/)

The static content summaries cannot reflect database content changes. Therefore, defining schedules for updating the database content summaries is an indispensable task. In section 4, we analyze the changes of database content and give the strategy to resample for approximate content summary update.

3 Sampling for Approximate Content Summary

As we discussed above, an important task is to access random sample for approximate content summary via querying.

3.1 Attribute-Level Sample

As we know, commercial DBMS commonly uses one-dimensional histograms which do not provide correlation information between attributes. It is generally not feasible to compute the multi-dimensional histograms for all attribute combinations, because the number of combinations is exponential in the number of attributes. Similarly, it is also not feasible to do complete query probes, which is the only approach to obtain data from Web database, due to the huge number of possible combinations of values on all the attributes. Therefore, an alternative approach is to obtain the random attribute-level sample. Based on these samples, we can estimate the approximate content summary as follows.

We probe the reality Web database by a few attribute values and gain the reality $d(a_{ij},$Ai). Then, the size of the database can be estimated as follows.

$$|\hat{D}| = \frac{d(a_{ij},A_i)_{reality}}{d(a_{ij},A_i)_{sample}} |D_{sample}| \quad (1)$$

Consequently, the estimate of the distribution $d(a_{ij},A_i)$ can be calculated.

$$\hat{d}(a_{ij},A_i) = d(a_{ij},A_i)\frac{|D_{sample}|}{|\hat{D}|} \quad (2)$$

Therefore, the key issue is how to obtain a high quality attribute-level sample. A naive approach is to collect the sample through submitting query probes on other attributes. However, it is difficult to guarantee that the sample is random due to the various correlations among the attributes. We propose an attribute-correlation-based sampling approach to make the sample as random as possible.

3.2 Attribute Correlation

In its most general sense, attribute correlation denotes the interdependence between quantitative or qualitative data of the different attributes. In this paper, we use Attribute Word Distribution of different attributes to define the concept of attribute correlation.

Definition 3 ***Attribute Word Distribution (AWD)****. Given all the words w_1, w_2, … , w_m of the values of attribute A in a database D, the Attribute Word Distribution for A is a vector $\vec{w}$, each component of which $w_i(i=1,2,\ldots,m)$ is the frequency of the word w_i. Under the assumption that no word appears more than once in an attribute value, the frequency of the word w_i is the number of tuples returned by the query $\pi_{A=w_i} D$.*

Definition 4 ***Attribute Correlation*** *is the dependence between any attribute pair ($Attr_u$, $Attr_v$) and is measured by the difference of the Attribute Word Distributions of the returned results on an attribute ($Attr_u$).Those returned results are obtained through submitting different values on another attribute ($Attr_v$). The more different the Attribute*

Word Distributions are, the more dependent the $Attr_u$ is on the $Attr_v$. The more dependent the $Attr_u$ is on the $Attr_v$, the stronger the Attribute Correlation of the attribute pair is.

Example 3. *In Academic Papers domain, because the topics that every Author focuses on are different, the Attribute Word Distributions of the corresponding results on attribute Title by submitting the query probes on the attribute Author are usually quite different. So Attribute Correlation of Title and Author is relatively strong. However, if we submit different values on attribute Year, the Attribute Word Distributions of the different returned results on attribute Title are very similar because the topics usually change very slowly each year. Therefore, the Attribute Correlation of Title and Year is relatively weak.*

Note that the differences of *Attribute Word Distribution*s refer to the direction differences of the vectors, i.e., if the probability of each component w_i in $\vec{w}_x$ is equal to that of the same component w_i in $\vec{w}_y$, these two *Attribute Word Distribution*s is identical.

Example 4. $\vec{w}_o = (w_1, w_2, w_3) = (100,60,20)$, $\vec{w}_p = (w_1, w_2, w_3) = (20,60,100)$, and $\vec{w}_q = (w_1, w_2, w_3) = (300,181,160)$, the difference of Attribute Word Distributions *between* $\vec{w}_o$ *and* $\vec{w}_p$ is much larger than that between $\vec{w}_o$ *and* $\vec{w}_p$.

A measure of the distribution difference is Kullback-Leibler(KL) divergence[2]. If we submit different queries Q_1, Q_2, …, Q_s on $Attr_v$, we will gain the corresponding result sets S_1, S_2,…, S_s on $Attr_u$. Suppose that S is the union of S_1, S_2, …, S_s and S consists of a set of words w_1, w_2, …, w_k. Then the KL-divergence of $Attr_u$ from S to S_j [3] is:

$$DK(S \parallel S_j) = \sum_{l=1}^{k} prob(Attr_u = w_l \mid S) \log \frac{prob(Attr_u = w_l \mid S)}{prob(Attr_u = w_l \mid S_j)} \tag{3}$$

where $prob(Attr_u = w_l|S)$ refers to the probability that $Attr_u = w_l$ in S and $prob(Attr_u = w_l|S_j)$ refers to the probability that $Attr_u = w_l$ in S_j; if w_l is not in S_j, it will be ignored. In reality, the divergence of the word distribution can be reflected by other w_l contained in S_j.

Attribute correlation is the average of the KL divergence of $Attr_u$ from S to S_j:

$$Correlation(Attr_u, Attr_v) = \frac{1}{s} \sum_{j=1}^{s} D_{KL}(S \parallel S_j) \tag{4}$$

The smaller the divergence is, the more independent the $Attr_u$ is of the $Attr_v$. Thus, the less correlative, the $Attr_v$ is with the $Attr_u$.

3.3 Correlation-Based Sampling

In this section, we give a correlation-based sampling approach to make the attribute-level sample as random as possible. Given a Web database, its interface contains several attributes. Assume that we want to discover an attribute $Attr_i$ that is the least correlative with attribute $Attr_u$ (e.g., *Title*). First, we select any attribute $Attr_v$ (e.g., *Subject*) except for $Attr_u$ from the Web database interface and submit some query probes on $Attr_v$ to the Web database. Then we collect and extract each returned result set on $Attr_u$ and store them in a table (line 3-7). Second, analyze appearance probability of each word in every returned result set and compute the attribute correlation of

$Attr_u$ with $Attr_v$ (line 8-12). The iteration continues until each attribute except $Attr_u$ is selected (line 2-12). Finally, select the minimal correlation value and the corresponding attribute $Attr_i$ is what we want to discover (line 13-14).

```
Algorithm 1: Attribute-Correlation-based Sampling
1  Begin
2    for each v=1,2,…,m and v<>u do
3       for each j=1,2,…,k do
4          Submit Q_j on Attr_v
5          Extract and store the result set S_j
6       End for
7       Merge S_{j=1,2,…,k} to S
8       for each j=1,2,…,k do
9          DKL(S||S_j)
10      End for
11      Correlation(Attr_u, Attr_v)
12   End for
13 Correlation(Attr_1,…,Attr_i)
                  =Min(Correlation(Attr_u, Attr_v))
14 return Attr_i
15 End
```

Example 5. *Table 3 shows the Attribute Correlation in Academic Papers Domain. We can see that attribute Year is the least correlative with attribute Title, which means the values of Year have the weakest affect on the values of Title.*

Table 3. An example of Attribute Correlation

$Attr_v$	Correlation(Title, $Attr_v$)
Title	/
Author	0.8539
Conference	0.09066
Journal	0.08751
Year	0.03112

After discovering the least correlative attribute $Attr_i$, we submit some query probes on $Attr_i$ to the Web databases and collect the returned results on attribute $Attr_u$ as the attribute-level sample of $Attr_u$, which is the approximate random sample. Then we order the words of the sample by their frequencies and the word rank can be viewed as the actual one due to the randomness of the sample.

4 Survival Function for Sample Schedule

As discussed in Section 1 and 2, when the underlying Web database change, the approximate content summary based on sample need to resample accordingly. Unfortunately, updating a content summary is an exhausting task, because the content summaries of Web databases are constructed via querying the databases.

Therefore, it is important to schedule resampling carefully. In this section, we present our "survival function" approach for deciding *when* to resample for constructing content summaries.

4.1 Survival Analysis

Survival analysis is a statistical method initially used to predict the time of survival for patients under different treatments. For this reason, it is also called *survival time.* It can help predict the time when an event occurs.

Before using any survival analysis technique for our problem, we need to define "change." A straightforward definition is that two content summaries $C(D)$ and $C(D, t)$ for the database D are not identical. However, a small change in database D is unlikely to be of importance for database selection. Therefore, we definite survival time in this paper as follows:

Definition 5 ***Survival Time.*** *of a content summary C(D) for database D is the minimum number of days t such that C(D, t) becomes "sufficiently different" from the current content summary C(D).* Therefore we define the survival time as follows.

$$C(D, t) - C(D) > \tau \tag{5}$$

where, we give 0.1 as the threshold τ. And $C(D, t)$ becomes "sufficiently different" can be reflected by the distribution changes of the attribute values. The distribution change of Attribute value a_{ij} of A_i can be calculated as follows.

$$\frac{d(a_{ij}, A_i, t) - d(a_{ij}, A_i)}{d(a_{ij}, A_i)} \tag{6}$$

Meanwhile, it is worth noting that the importance of each attribute is not the same which will affect the result of "sufficiently different". A reasonable assumption is that the more important is the attribute, the more greatly the distribution changes of its values affect the database content. Usually, we should give the more important attribute a larger weight; in contrast, the less important attribute a relative smaller weight. This issue will be illustrated in the next section.

4.2 Attribute Importance

The users usually submit their queries on the important attributes. In the other words, the importance of attributes is directly related to the frequency of their occurrence in query strings in the workload. Consider the database D_1 discussed in Section 2, if there are more queries requesting on attribute *Title* than attribute *Year*, the frequency of attribute *Title* appearing in the workload will be more than that of attribute *Year*. It tells us a piece of information that the importance of Attribute *Title* will be more important than attribute *Year*.

A feasible idea that takes advantage of this observation is to record the frequencies of attributes appearing in the workload, and then let importance coefficients depend on these frequencies. Let $QF(A_i)$ be the frequency of occurrence of Attribute A_i in the query strings of the workload. We define the importance coefficient β_i of attribute A_i as follows.

$$\beta_i = QF(A_i) / \sum_{k=1}^{n} QF(A_k) \tag{7}$$

4.3 Selection of Attribute Values

The distribution change of attribute value may be detected for *categorical* attribute because it is convenient to enumerate the values of such an attribute and we can submit each value as a query to obtain the corresponding number of results (i.e., the hit number which is provided by most of Web databases). However, it is not suitable for *Text* attribute (e.g., *Title* attribute). This is because the values of this type of attribute are infinite and the cost of submitting these values to the Web database one by one will be unacceptable. In this case, we should select the finite values to reflect the changes of all values.

As we observed, only a small portion of *Text* attribute values occurs more frequently as illustrated in Fig.2. These frequent words represent the content of the database and are indeed what the database concern on. (Note that, the stop words have been deleted from these values.) Thus, the changes of these "hot" words are sufficient for estimate the changes of the database on according attribute.

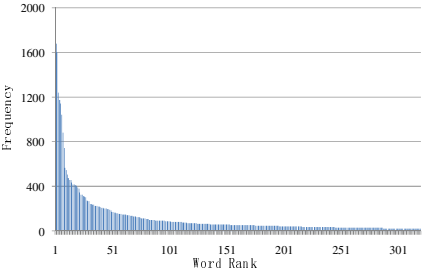


Fig. 2. Frequencies of Text Attribute values

4.4 Estimation of Survival Function

We describe the survival time model through the following survival function.

Definition 6 Survival function $S(t)$ *predicts the time point t, which represents the number of days between two updates of a database, when t' is greater than* t *, the change of database content will greater than the threshold* τ..

As discussed above, given the distribution changes of a part of attribute values and the differences of attribute weights, we formally define *Survival function* as follows.

$$S(t) = C(D,t) - C(D) = \sum_i \beta_i \left(\frac{1}{n}\sum_{j=1}^{n} \frac{d(a_{ij}, A_i, t) - d(a_{ij}, A_i)}{d(a_{ij}, A_i)}\right) > \tau \tag{8}$$

where $t=t_0$ means that the metaquerier should update the approximate content summary of the database every t_0 days. And n equal to the number of the top words selected from all the values when A_i is a *Text* attribute. Otherwise, n is the number of values when A_i is a *Categorical* attribute or a *Numeric* attribute.

We compute the survival times for all the databases and all the time points in our data set and select the *useful* database features (the topic, domain and size of the

database) for predicting the survival time. Finally, we run a Cox regression analysis with the survival times as dependent variables and the useful database features as independent variables.

$$\ln S(t) = \exp(\sum_j \lambda_j x_j) H_0(t) \tag{9}$$

Where $H_0(t)$is the Baseline Cumulative Hazard function, λ_j is regression Coefficient.

5 Experiments

We now report the results of the experiments. All the proposed algorithms are implemented in Java, SPSS and MySQL.

5.1 The Data Set

We have selected 50 Web databases as our data sets which cover three domains: Papers, Movies and Books. Their contents were downloaded every 10 days for 9 months.

5.2 Evaluation Method

We evaluate our approach with the accuracy measure which is defined as follows.

$$Accuracy = \frac{1}{N} \sum_i \sum_j \frac{\hat{d}(a_{ij}, A_i) - d(a_{ij}, A_i)}{d(a_{ij}, A_i)} \tag{10}$$

So *Accuracy* is the average percentage of the estimation deviation and actual distribution, where the estimation deviation is the difference between the estimation distribution and the actual distribution. Note that, we can submit some attribute values to Web database to obtain corresponding actual distribution of attribute value.

5.3 Experimental Results

The static approximate content summary is not fit for database selection because its accuracy decreases obviously after a period time. In contrast, the accuracy of changeable approximate content summary is relatively stable, as Fig. 3 shows. In theory, the shorter is the time interval, the more accuracy is the approximate content summary. In order to avoiding unnecessary cost, we calculate the time interval to resample for regenerating new approximate content summary. As Fig. 3 shows, when t is less than update time, the change of accuracy is not obvious.

We also do experiments on the relationship between accuracy of approximate content summary and different n in formula 8. Let n equal to 50, 100, 200, 400 to predict the update time individually. Experiments show, when n is greater than 100, the accuracy increases very slowly.

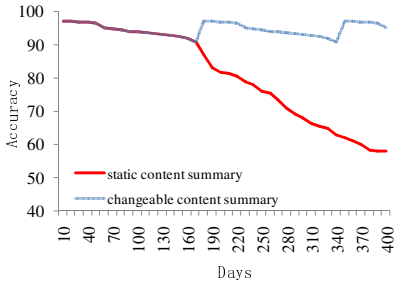


Fig. 3. Comparison between static content summary and changeable content summary

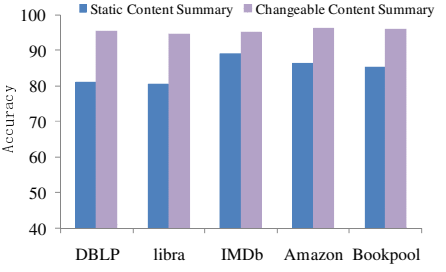


Fig. 4. The accuracy of approximate content summary

Overall, as we can see from Fig.4, the accuracy of our approach is generally good. The average accuracy of approximate content summary is 95.2%, which is much higher than that of static content summary.

6 Related Work

In recent years, there has been significant interest in the study of database selection.

For the text databases, based on the frequencies of the words in a database, reference [4] proposes a statistical approach for estimating the number of potentially useful documents in a database. Reference [5] captures the dependencies of the words in the database representative for text database selection. However, those methods are not applicable when data sources are "uncooperative".

For the hidden Web text databases, some work generates the description of the content of the text databases manually. Such an approach would not scale to the thousands of text databases on the web. Callan et al. [6] probe text databases with random queries to determine an approximation of their vocabulary and associated statistics. Ipeirotis et al. [7] present an automatic "focused-probing" sampling algorithm that detects the topics cover in a database, adaptively extracts documents, and constructs the content summary. However, these techniques do not apply to a scenario where there is complex interface to the Web structured database.

For the hidden web structured databases, reference [8] employs simple random sampling and stratified random sampling to decide the probing queries and addresses the database selection problem of learning the statistics for sources with respect to user queries. And reference [9] proposes a random walk approach to sampling the hidden databases in Deep Web scenario. However, it is more complex compared with our attribute-level sampling method.

7 Conclusion

Content summary is usually used for text database selection algorithm. However, in Web structured database integration, it is very difficult to access all the data for constructing

content summary. Furthermore, static content summary can not accurately reflect the database content which changes all the time.

In this paper, we propose an approximate, sample-base content summary approach for database selection. And we present a survival function to predict the time to re-generate the approximate content summary. Experimental results on several large-scale Web databases indicate that our approach can achieve high accuracy on approximate content summary for database selection.

Acknowledgments. This research was supported by the grants from the Natural Science Foundation of China (No: 60773216).

References

1. The Deep Web: Surfacing Hidden Value, http://www.completeplanet.com/Tutorials/DeepWeb/
2. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley and Sons, Chichester (2001)
3. Jiang, F., Meng, W., Meng, X.: Selectivity Estimation for Exclusive Query Translation in Deep Web Data Integration. In: Chen, L., Liu, C., Liu, Q., Deng, K. (eds.) DASFAA 2009. LNCS, vol. 5667, pp. 595–600. Springer, Heidelberg (2009)
4. Meng, W., Liu, K., Yu, C., Wang, X., Chang, Y.: Determining Text Databases to Search in the Internet. In: VLDB 1998, New York, pp.14–25 (1998)
5. Wu, W., Yu, C., Meng, W.: Database Selection for Longer Queries. In: The 2004 Meeting of the International Federation of Classification Societies, Chicago, pp. 575–584 (2004)
6. Callan, J.P., Connell, M.E.: Query-based sampling of text databases. J. ACM Transactions on Information Systems (TOIS) 19(2), 97–130 (2001)
7. Ipeirotis, P., Gravano, L.: Classification-Aware Hidden-Web Text Database Selection. J. ACM Transactions on Information Systems (TOIS) article 6 26(2) (2008)
8. Nie, Z., Kambhampati, S.: A Frequency-based Approach for Mining Coverage Statistics in Data Integration. In: ICDE 2004, Boston, pp. 387–398 (2004)
9. Dasgupta, A., Das, G., Mannila, H.: A random walk approach to sampling hidden databases. In: SIGMOD 2007, Beijing, pp. 629–640 (2007)

Detecting Comment Spam through Content Analysis*

Congrui Huang, Qiancheng Jiang, and Yan Zhang**

Key Laboratory of Machine Perception, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University Beijing
hcr@pku.edu.cn, {jiangqiancheng,zhy}@cis.pku.edu.cn

Abstract. In the Web 2.0 eras, the individual Internet users can also act as information providers, releasing information or making comments conveniently. However, some participants may spread irresponsible remarks or express irrelevant comments for commercial interests. This kind of so-called comment spam severely hurts the information quality. This paper tries to automatically detect comment spam through content analysis, using some previously-undescribed features. Experiments on a real data set show that our combined heuristics can correctly identify comment spam with high precision(90.4%) and recall(84.5%).

1 Introduction

From its original intention as a platform for sharing data, the web has grown to be a central part of cultural, educational and commercial life. On the internet, people could purchase goods, keep an eye on people around and exchange ideas at any time. The principle of human centeredness helped bring about the idea of social network service, which was first raised by Starr [1]. Social network service provides a platform for people to dig topics and activities they concerned. The usage of social network opens up the virtual and real life: improving communication, presenting business opportunities and advancing technology. So, improving the quality of social networking is well worth considering. Blog, as a kind of social networking, has been widely used in recent years. In this paper, we will propose a method to perfect the quality of the blog.

As blog sites rely on user-generated content, making them both incredibly dynamic and tempting targets for spam [2]. Spam, a cause of low-quality social networking, not only brings bad impression on users experience, but also blinds the search engine. Comments, the main part of a blog, is a good indication for the significance of the weblog [3], most bloggers also identify comment feedback as an important motivation for their writing [4, 5]. In this scenario, comment spam hurts both bloggers and users. Blog publishers provide comments so that their thoughts can be easily shared, but once you allow readers to add comments,

* Supported by NSFC under Grant No.60673129 and 60773162,863 Program under Grant No.2007AA01Z154.

** Corresponding author.

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 222–233, 2010.

you also provide a shortcut for spammers to abuse the comment system. To solve the problem raised by being open, we cannot turn off comment system radically. Although filtering comment spam manually may be effective, it is an exhausting and unrealistic method. Practitioners and researchers have tried to find an efficient and automatic way to combat comment spam. In the last few years, research in this respect has led to good results. In this paper, we proposed a heuristic method on the basis of predecessors' works.

We noticed that Trevino [5] and Mishne et al. [6] mainly focused on link spam common in blog comments. Mishne et al. has followed a language modelling approach for detecting link spam in blogs and similar pages. KL-divergence, the main method to solve their problem, is barely one part of our work. What's more, link spam is one of the manifestations of comment spam. Sculley [7] summarized that academic researchers advocated the use of SVMs, while practitioners prefer Bayesian methods for content-based filtering. Sculley also showed his Relaxed Online SVM to reduce computational cost.

Creating an efficient way to detect comment spam is a challenging problem. We must ensure that we do not mistakenly consider legitimate comments to be spam. In this paper, we propose a fresh approach to filter comment spam. We consider extracting several features of comments. By analyzing the combined features, we get relative rules to filter blog comments. In detail, we propose several features, according to the distribution of features, we apply heuristic methods to obtain judgment rules of blog comments for each web site, with which we check comments step by step. At last, we get comment spam set and the normal comments set.

Compared with traditional method, we do not apply classification algorithms directly, what we are faced with is the real and complicated network environment. With features we proposed, we built our own heuristic model, not the same as the classical ones. When we apply our model in popular blog sites, it removes comment spam effectively, certifying that our method is feasible indeed.

The remainder of our paper is structured as follows: In section 2, we discuss some related work. In section 3, we define comment spam. In section 4 and 5 we analyze comment features and describe our heuristic methods. In section 6 we show our experimental framework and the real-world data set that we used.

2 Related Work

Academic research directed towards comment spam is relative rare, however, we could have a look at some existing solutions. Spammers will reply to their own comments, while normal users will not [8], so comment spam can be reduced by disallowing multiple consecutive submissions. Google is advocating the use of rel="nofollow" [9] tag in order to reduce the effect of heavy inter-blog linking on page ranking. Special software products such as Math Comment Spam Protection Plugin for Wordpress [10] and Movable Type [11] have their own ways to prevent comment spam.

A survey [2] showed that the three main anti-spam strategies commonly used in practice are: Identification-based, Rank-based and Interface or Limit-based. It

mentioned that the third method has been used to prevent comment spam. Adam Thomason [12] has reviewed the current state of spam in the blogosphere, concluding that anti-spam solutions developed for emails are effective in detecting blog.

Also, Gordon et al. [13] has considered the problem of content-based spam filtering that arises in three contexts, blog comments involved. Their experiments are conducted to evaluate the effectiveness of state-of-the-art email spam filters, without modification. Further experiments are conducted to investigate the effect of lexical feature expansion. Detection of Harassment on Web2.0 [14] employs content features, sentiment features and contextual features of documents, using a supervised learning approach to identify online harassment, including comment spam.

We believe that comment features we propose in this paper will do favor for other researchers of this field. We have supplied a simple, but more practical method to filter comment spam. Our work will throw new light on comment spam studies.

3 Comment Spam

Normally, comments cannot exist independently, as should be attached to the body of the blog. Some related concepts should be declared in order to define comment spam.

Definition 1. *Spam Info. If comment s contains publicity information or advertisement, generally refers to hyperlink, e-mail, phone number, MSN and so on. We call s Spam Info, and use Spam Info(s) to measure it.*

Definition 2. *Correlation. If comment c discusses something about page p, then the Correlation between c and p will be marked as Rel(C, P). Generally speaking, Rel(C, P) can be indicated by similarity, that is Rel(C, P) = Sim(C, P).*

Usually we use two methods below to compute the similarity between c and p. Cosine Similarity, we get term-frequency vectors C and P of comment c and page p, then the cosine similarity, Sim(C,P), is represented using a dot product and magnitude:

$$\boldsymbol{Rel(C,P)} = \frac{\boldsymbol{C \cdot P}}{\boldsymbol{|C| \cdot |P|}} \tag{1}$$

In information theory, KL-Divergence has been used to indicate the divergence between two distributions. For probability distributions C and P of a discrete random variable, the KL-Divergence of P from C is defined to be:

$$\boldsymbol{D_{KL}(C||P)} = \sum_{\boldsymbol{i}} \boldsymbol{C(i)} \log \frac{\boldsymbol{C(i)}}{\boldsymbol{P(i)} + \boldsymbol{\varepsilon}} \tag{2}$$

Rel(C, P) will be calculated as:

$$\boldsymbol{Rel(C,P)} = \mathbf{1} - \boldsymbol{D_{KL}(C||P)} \tag{3}$$

Unlike cosine similarity, KL-Divergence does not satisfy symmetry and triangle inequality, C and P are two different distribution densities of random variable χ. KL-Divergence measures the difference between C and P. As for our problem, we would like to know how far each comment is from the blog text.χ indicates words in each comment. In (2), χ is referred to i, C(i) and P(i) represent the number of times the word i appears in each corresponding text. We have to claim that C and P in (3) do not represent term-frequency we used for (1). C indicates the distribution of each comment, the probability distribution of the blog body corresponds to P. As not every word in the comment will exist in the blog body, we need to add a small constant to deal with a zero denominator, in (2), the constant is ε,in our experiment, ε is greater than 0 and less than 0.5.Facts show that KL-Divergence is a good method to compute relevance in our model. Thus, with definition 1 and 2, we have the definition below.

Definition 3. *Comment Spam. If comment c satisfies Spam Info(c)¿α . Rel(C, P) ¡β, then c is considered to be comment spam.*

4 Content-Based Spam Detection

Ntoulas et al. [15] have analyzed several features of spam web pages. Based on their work, we selected 2,646 blogs randomly as the training set and labeled the data manually, as a result, we got 277 spams. According to the feature distribution of the set, we proposed some features to distinguish the normal comment from comment spam. Features used in our model will be discussed here.

4.1 The Length of the Comment

Length is an apparent feature of a comment, so we investigated whether length is a good indicator of spam. To this end, we plotted length distribution for comments, the result is showed in Fig. 1. This figure consists of a bar graph and a line graph. The bar graph depicts the distribution of a certain interval of length of comments in our training set. The horizontal axis depicts a set of value ranges. The left scale of the vertical axis applies to the bar graph, and depicts the number of comments in training set that fell into a particular range. The right scale of the vertical axis applies to the line graph, and depicts the percentage of sampled comments in each range that have been judged to be spam.

As can be observed in Fig. 1, short comments hold a high percentage. In general, the number of comments declines, while the possibility of appearance of spam rises, as the comment length increases. This supports the intuition that in the real world, normal comments are always short and to the point, while spammers will repeat its propaganda or discuss a topic not related to the text in detail. Obviously, such a feature cannot indicate comment spam alone. However, we will be able to identify a spam by combining other features, for example, a long comment with low relevance to the blog body is more likely to be a spam compared with a short one.

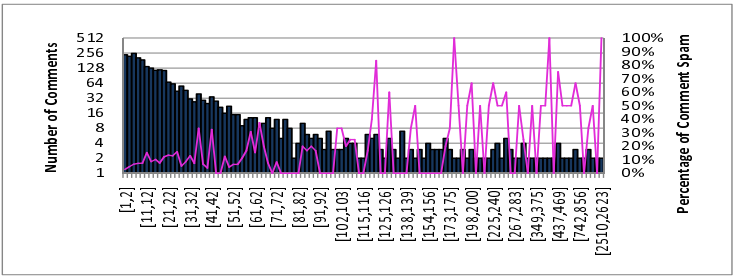


Fig. 1. Relationship between comment spam and comment length

4.2 Similarity

A regular comment is always relevant to the body of the blog, except some short ones. Thus, text similarity will do favor to filter comment spam. However, short normal comments may have low cosine similarity and long comment spam may have high cosine similarity, which is also the major drawback of measuring comment spam by computing text similarity. Like Fig. 1, Fig. 2 also has a bar graph and a line graph. Comments collect in area with low similarity. A commentator will not write long comments, also it will not comment on the blog with the original text of the blog content. Everyone may use its own words to represent its opinion. As text similarity is a statistical method, similarity between comment and the blog text is certainly not very high. When similarity is greater than 0.28, comments start to shrink and comment spam do vanish.

4.3 KL-Divergence

KL-Divergence has a good effect on exploring the text difference. Short comments may have low similarity but low divergence comparatively speaking. Long comment spam may have high similarity but high divergence relatively speaking. Thats why we use this feature as a complement to the method of cosine similarity. Compared with Fig. 2, comments collect in area with low divergence,

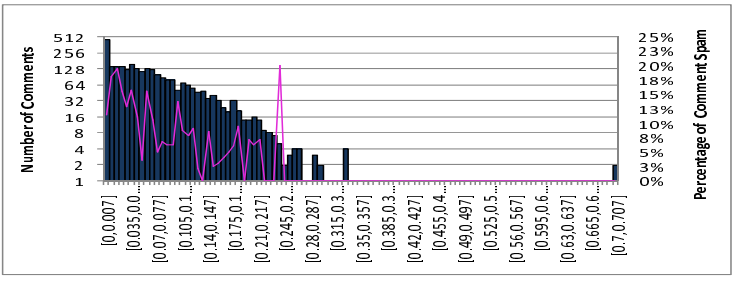


Fig. 2. Relationship between comment spam and similarity

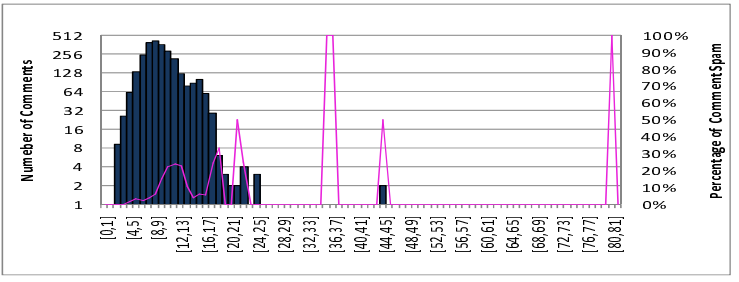


Fig. 3. Relationship between comment spam and KL-Divergence

indicating that comments related to the blog content account for the majority. When divergence becomes high, the possibility of being comment spam goes up. Still we cannot identify comment spam by divergence alone.

4.4 Popular Words Ratio and Propaganda

Generally speaking, comment spam always contains propagandistic information to propagate the commentator's website or its business. So whether a comment contains URL, E-mail address or whatever, will be a good cue for a spam. Fig. 4 shows the distribution of comments on Propaganda.Some short comments, such as "bravo", "marvelous" and other common comment terms, though meaningless, should not be classified as comment spam actually. Hence, we collect a common glossary to check popular words ratio of a comment to identify such short and low correlation comments. Popular words ratio is used to measure the proportion of those words we mentioned above in a comment. Fig. 5 shows the distribution of comments on popular words ratio.

Clearly, when a comment contains a lot of propagandas, it must be comment spam. Normal comments will not contain propagandas or contain little. Fig. 4 indicates that propaganda is a feature with high discriminability. Through this distribution, we notice that when a comment only contains popular words it will not be considered to be a spam. Spams stress on area where the ratio of popular words is low. Here we may discuss a kind of comment spam, from Fig. 5, we find that there is still comment spam even when the ratio is greater than

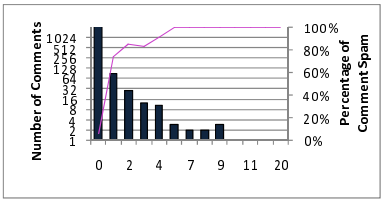


Fig. 4. Distribution of Propagandas

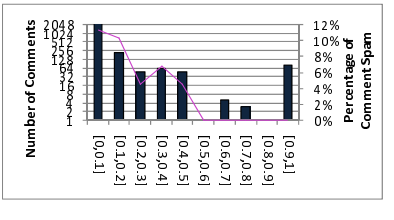


Fig. 5. Distribution of Popular words Ratio

0.4. Some spammers may use some popular words to disguise their comments. Behind these words, they always put the propagandistic information. This very fact underscores that we should combine features mentioned above to identify comment spam.

5 Using Classifiers to Combine Heuristics

How can we identify comment spam with features mentioned above? The combination of these features would access to our purpose. Our purpose is to classify comments, so we should take some classification algorithms into account. We build 3 simple models on the training set to see which method is better. Results are showed in Table 1.

Table 1. Classification Model Comparision

Models	Precision	Recall
SVM	67.1%	30.7%
Naive Bayes	70.9%	46.0%
$DecisionTree_{(c4.5)}$	84.7%	44.9%

Due to the unadjusted parameters, these models do not exhibit their best performance, especially the values of recall. However, we can see that decision tree model performs better and thus it is selected and enhanced as our final solution. Actually, identifying comment spam is a decision process. Each feature can decide whether a comment is a spam, with the decision of one feature we get new classification results of comments, finally all features will make their decisions to obtain the ultimate results whether a comment can be ranked as a spam.

We are encouraged to employ a heuristic decision method to build a tree-model. In simple terms, after we get features of each comment, we apply statistical method to measure a single feature's ability to classify comments. The best feature will be chose as the root of a tree. Then every possible decision of the root will be treated as its son. The corresponding comments will be put under the proper branch. Repeat the process; choose the current best feature from the comment set embedded in branch nodes.

The best feature should have high resolution. For mass behavior is credible, when a comment's value of a feature closes to the integral level, it will be ranked as a normal one, else further decisions are needed. We examine skewness coefficient[1] of the distribution on each feature of all comments to select the best feature. Symmetrical distribution always implies lower discriminability compared with long-tailed distribution. Thus, the bigger the absolute value of skewness coefficient is, the better the feature is.

We may take the mean value or mode of the distribution as a threshold. Note that there is big difference in the style and participants of a blog site in real network.

[1] Skewness coefficient: $\mathrm{Skew(x)}=\mu^2/\sigma^2=1/\mathrm{n}\sum_{i=1}^{n}(x_i-\tilde{x})^3/(1/n\sum_{i=1}^{n}(x_i-\tilde{x})^2)^{2/3}$.

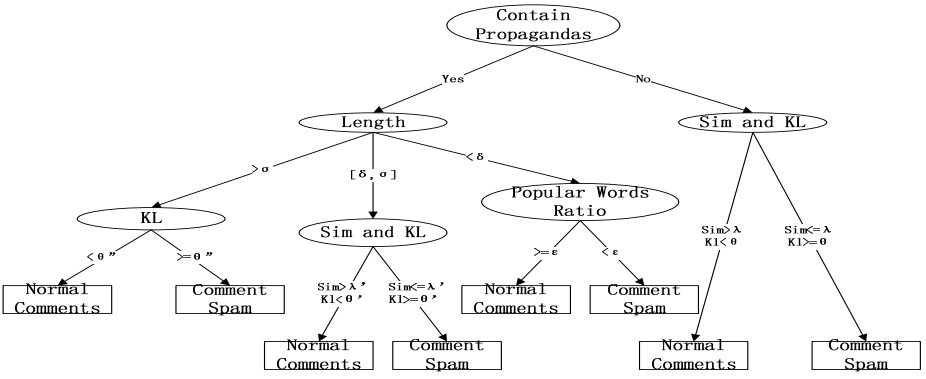


Fig. 6. Core Part of the heuristic model

So strictly speaking, neither best features nor threshold is the same for different blog sites, in other words, a uniform model cannot receive precise results.

Fig. 6 is a typical shape of the core part of our model. In Fig. 6 the best feature is propaganda. Continually, similarity and divergence of propagandistic comments are computed to minimize misjudgment. Comments without propagandistic information will be divided into three classes. Short comments with low popular words ratio should be considered to be comment spam. For long comments, we only use divergence to decide whether they are comment spam to avoid the noise of the long text.

6 Experiments and Results

6.1 Data Collection

Large benchmark data sets of labeled blog comment spam do not yet exist [8].Thus, we have to run our experiment on the only publicly available blog sites. Fig. 7 displays how we get our dataset. Our data mainly comes from popular blog websites, from which we get many representative comments. We select some seed sites[2], and we pick out such data that the length of the blog body is greater than 500 and the quantity of the comment surpass 25 from the crawling pages. The description of the crawling data will be found in Table 2. The crawling data is our testing set, which is different from our training set mentioned in section 4 and 5. As lack of hand-crafted training data, we have to label the raw corpus manually to form our relatively small scale training set.

6.2 Distribution of Comment Feature

Fig. 8 shows the distribution of comment length on each site. Assuming that most users will release valid comment, we come to this conclusion that the possibility

[2] http://blog.sina.com.cn; http://blog.sohu.com; http://hi.baidu.com/hi/cms/artivle/content[1-6]/index.html

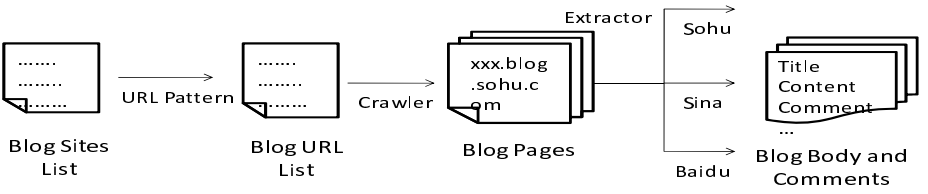


Fig. 7. Data Collection

Table 2. Statistical Data of Comments

Blogsites	Sumofarticles	Avgofartilcleslength	Sumofcomments	Avgofcomments
Baidu	298	545.3	11,563	38.8
Sina	359	845.3	10,518	29.3
Sohu	583	784.2	31,908	54.7
Total	1240	741.3	53,989	43.5

of a comment being a spam increased by its distance from the center which is represented by threshold. Our threshold will exceed the mean value and the mode within a certain range.

From Fig. 9, we can see that the distribution of propagandas is a discrete long-tail distribution. Popular words ratio gives focus on some short comment. Fig. 10 indicates the distribution of this feature. A number of comments contain popular words. Thus this feature can help us differentiate the meaningless comment from comment spam well. Finally, we analyze the similarity and divergence between the comment and the blog body. These two aspects can depict the relationship between comments and the body accurately. Fig. 11 and Fig. 12 are these two distributions. The divergence values distribute uniformly, mainly concentrate in the area of [-1,0].

6.3 Results and Evaluation

After analyzing the distribution of each feature, we build our heuristic model successfully. We get results in Table 3 by processing blog comments with our

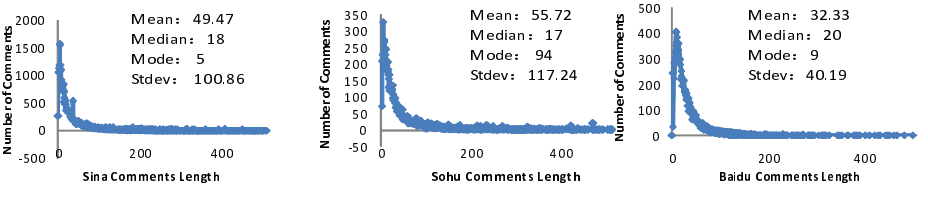


Fig. 8. Distribution of Comments Length on each site

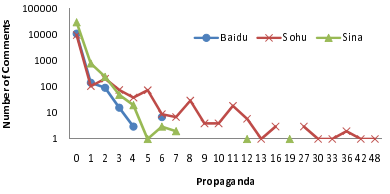


Fig. 9. Distribution of Propagandas

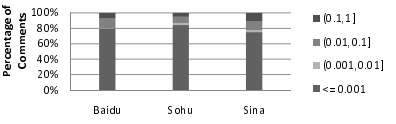


Fig. 10. Distribution of Popular words

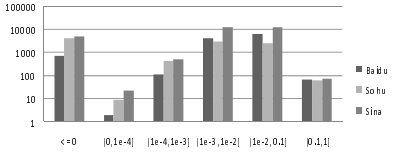


Fig. 11. Distribution of Similarity

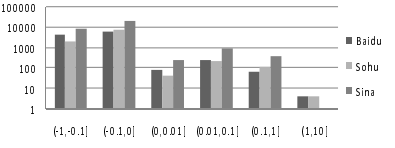


Fig. 12. Distribution of KL_Divergence

model. In general, comment spam makes up about 20% of the total comments. In our model, Sohu blogs are mostly current affairs which would gain more attentions, as a result, the Sohu site has a high proportion of comment spam.

To evaluate our model, we have to compute precision and recall of the results. Precision indicates the effectiveness of our method and the credibility of our model. Precision P will be calculated as:

$$\boldsymbol{P = \frac{C_{actual}}{C_{find}}} \tag{4}$$

C_{actual} is the amount of actual comment spam in our result. C_{find} is the amount of comment spam that our model has found.

Recall evaluates the identification ability of our model. Recall R will be calculated as:

$$\boldsymbol{R = \frac{C_{actual}}{C_{total}}} \tag{5}$$

C_{total} is the amount of comment spam in our corpus.

We apply sample annotation to reduce the workload. For precision, we select 500 comments from the results randomly, and mark comment spam manually

Table 3. Statistical Data of Comments

Blogsites	Sumofarticles	Sumofcomments	Sumofcomment_spam	Proportion
Baidu	298	11,563	1,108	9.58%
Sina	359	10,518	3,354	31.88%
Sohu	583	31,908	5,852	18.34%
Total	1240	53,989	10,314	19.1%

Table 4. Statistical Data of Comments

BlogSites	Precision	Recall
Baidu	92.6%(463/500)	86.4%(432/500)
Sina	91.4%(457/500)	83.0%(415/500)
Sohu	87.2%(436/500)	84.2%(421/500)
Total	90.4%(1,356/1500)	84.5%(1,268/1500)

by some measures[3]. With the number of the marked comments, we will get the precision. To get the recall, we extract comment spam with our heuristic model from a total of 500 spams that been marked in the whole comment set. With the above methods, we will get the precision and recall of the three different sites. From Table 4 we can see that the precision of our model is fundamentally satisfied. However, various comment spam have different forms of expression, which causes the low recall of our model.

7 Conclusions

This research deals with comment spam. Initially we defined comment spam, simultaneously, we proposed some features and analyzed these features of comments with statistical methods. Statistics show that comment spam would be filtered out by combining these features. Experiments show that our heuristic model can find comment spam effectively with high precision and recall. As far as we concerned, research on comment spam is still a novel topic at present. As a significant attempt, our method has acquired satisfied effects. In future, we should try to build determination model more reasonably to enhance recall. On the other hand, we could apply more knowledge and technology of other fields, such as Nature Language Processing, to combat comment spam more aggressively.

References

1. Starr, R.H., Turoff, M.: The Network Nation. Addison-Wesley, Reading (1978/1993)
2. Heymann, P., Koutrika, G., Garcia-Molina, H.: Fighting Spam on Social Web Sites: A Survey of Approaches and Future Challenges. IEEE Internet Computing 11(6), 36–45 (2007)
3. Mishne, G., Glance, N.: Leave a reply: An analysis of weblog comments. In: Proc. of WWW 2006 Workshop on the Weblogging Ecosystem (2006)
4. Gumbrecht, M.: Blogs as "protected space". In: WWW 2004 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics (2004)
5. Trevino, E.M.: Blogger motivations: Power, pull, and positive feedback. In: Internet Research 6.0 (2005)

[3] 1. Not relevant to the text. 2. Relevant to the text with obvious advertising.

6. Mishne, G., Carmel, D., Lempel, R.: Blocking blog spam with language model disagreement. In: Proceedings of AIRWeb (2005)
7. Sculley, D., Wachman, G.M.: Relaxed online support vector machines for spam filtering. In: 30th ACM SIGIR Conference on Research and Development on Information Retrieval, Amsterdam (2007)
8. Spam in blogs, `http://en.wikipedia.org/wiki/Spam_in_blogs`
9. Massa, P., Hayes, C.: Page-reRank: Using Trusted Links to re-Rank Authority. In: Proceeding of Web Intelligence Conference, France (2005)
10. Jayagopal, R.: No Business Like E-Business: The Spectacularly Simple Secrets Behind How You Can Create A Web Site And Make Money With It, pp. 85–87 (2007)
11. Lindahl, C., Blount, E.: Weblogs: Simplifying Web publishing. IEEE Computer 36(11), 114–116 (2003)
12. Thomason, A.: Blog Spam: A Review. In: Proceedings of Conference on Email and Anti-Spam (2007)
13. Cormack, G.V., Hidalgo, J.M.G., Snz, E.P.: Spam filtering for short messages. In: Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management (2007)
14. Yin, D., Xue, Z., Hong, L., Davison, B.D., Kontostathis, A., Edwards, L.: Detection of harassment on Web 2.0. In: Proceedings of the Content Analysis in the Web 2.0 (CAW2.0) Workshop at WWW2009 (2009)
15. Ntoulas, A., Najork, M., Manasse, M., Fetterly, D.: Detecting spam web pages through content analysis. In: Proceedings of the 15th International Conference on World Wide Web (2006)

Enriching the Contents of Enterprises' Wiki Systems with Web Information*

Li Zhao, Yexin Wang, Congrui Huang, and Yan Zhang**

Department of Machine Intelligence, Peking University, Beijing 100871, China
Key Laboratory on Machine Perception, Ministry of Education, Beijing 100871, China
{lizhao,yexin,hcr}@pku.edu.cn, zhy@cis.pku.edu.cn

Abstract. Wikis are currently used in providing knowledge management systems for individual enterprises. The initial explanations of word entries (entities) in such a system can be generated from the pages on the Intranet of an enterprise. However, the information on such internal pages cannot cover all aspects of the entities. To solve this problem, this paper tries to enrich the explanations of entities by exploiting Web pages on the Internet. This task consists of three steps. First, it obtains pages from the Internet for each entity as an initial page set with the help of search engines. Secondly, it locates the pages which have a high correlation with the entity from the page set. At last, it produces new snippets from such pages and chooses those which can enhance the explanation and throw away the redundant ones. Each candidate snippet is evaluated by two aspects: the correlation between it and the entity, and its ability to enhance the existing explanation. The experimental results based on a real data set show that our proposed method works effectively in supplementing the existing explanation by exploiting web pages from outside the enterprise.

1 Introduction

Wikis are getting much popular in the web users' life. They provide clearer and neater information to users than original web pages do. Many enterprises try to construct wikis about themselves as their knowledge management systems. Most of enterprises have their Intranet on which there is confident information about enterprises. Enterprises' Wiki system can be built basing on information from the Intranet. Moreover, much confident information related to an enterprise on the Internet is not on its Intranet. These information should be added into the Enterprises' Wiki.

In our previous work [12], we proposes the MagicCube to automatically construct an Enterprises' Wiki system. However, all these tasks are based on the Intranet environment of one enterprise. Obviously information in the Intranet is not

* Supported by NSFC under Grant No.60673129 and 60773162,863 Program under Grant No.2007AA01Z154, and the 2008/2009 HP Labs Innovation Research Program.

** Corresponding author.

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 234–245, 2010.

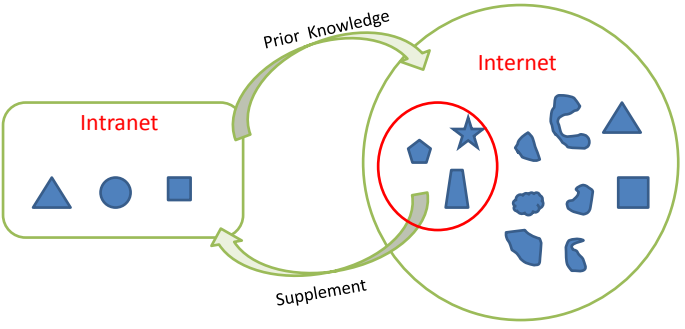


Fig. 1. Explanation enrichment of an entity

sufficient to describe a person, a product or even an event. In this paper, we consider enriching the contents of Enterprises' Wiki system by using the Internet information. This task begins from Intranet. We use the internal information to construct prior knowledge of an entity. Then we exploit the prior knowledge to choose proper segments of pages from Internet to supplement the description we have had.

Figure 1 shows the explanation enrichment of an entity. The rectangle stands for the Intranet environment. The triangle, circle and the square are snippets of an entity. The big circle on the right stands for the Internet environment. The patterns in it are snippets related to the entity. The ones in the red little circle are chosen to reinforce the explanation from Intranet.

Our method has three steps as follows.

Retrieving external web pages: First, we need to get information from Internet which is related to an entity. In this step, we use a search engine. Queries are provided to the search engine. These queries are all built from the internal web pages. We use a simple but effective way to get these queries. The top pages the engine returns are downloaded as the initial page set.

Computing correlation: We want to use the information from outside the enterprise to enhance the description of entities, so we should first compute the correlation between the entities and external pages which we collect in the first step. The simple way is computing cosine value between them. However, the cosine value is not good at expressing the correlation when entities are very short. [12] extracts segments from internal pages as snippets to describe entities. In this step, we compute the correlation value between internal snippets and external pages of an entity to represent mutuality of external pages and the entity.

Choosing snippets: Sometimes there are much overlap between the existing snippets and the new ones extracted from the external web pages. Except computing the correlation between them, we compute the divergence value to choose novelty snippets and throw away redundant ones.

We will demonstrate the effectiveness of our approach by experiments running on a real data set which was from HP Labs China. The results indicate that our method is valuable and meaningful.

Our contributions are summarized as follows:

We propose an idea to exploit web pages on the Internet to enrich the contents of Enterprises' Wiki system. In this task, we present some simple but effective method to deal with problems like correlation computing between an entity and its related external web pages and picking novelty snippets from the external snippet set of an entity.

The rest of the paper is organized as follow. Section 2 shows related work. In Section 3 we present our approach. Section 3.1 gives an outline of the method. Section 3.2 presents our prior work. Section 3.3 describes our simple method to retrieve external web pages. Section 3.4 gives a model of computing correlation value. Section 3.5 introduces the algorithm of choosing novelty snippets. Section 4 demonstrates the experiments. Section 5 presents our evaluation method. And in the last part of this paper, section 6, we will conclude our work and give the future work.

2 Related Work

Search engines provide users with pages which are involved with the query. In our work, we extract segments from external web pages to supplement the description of entities we have in the Intranet. The early work, document abstraction, is meaning in the area of Natural Language Processing. And it is an important subject in the Internet information processing. The main idea of it is information mining and abstract. Document abstraction extracts the segments automatically from the original pages which are comprehensive and then organizes these pieces into a coherent passage. Columbia Newsblaster system summarizes all the news on the web every day. Dragomir R. Radevyz et al.[6] present a system called WebInEssence to search for information effectively based on query and automatically summarize selected documents based on the users' personal profiles. Their work is related to researches on clustering and summarizing of web search results. The works in [10] [11] [9] are around this problem. There are some other studies on summarization, such as [4] [1] [3]. Our work differs from these summarization work. We endeavor to select segments from external web pages based on the prior knowledge which is extracted from the internal expression of entities.

Josef and Karel[8] propose a method to update summary of a set of topic-related documents. The prior summary is determined by a set of older documents of the same topic. The approach in this paper is based on Iterative Residual Rescaling(IRR) which generates a latent semantic space of documents in consideration. It consists of two step, sentence extraction and selection. In sentence-extractive summarization step, compute the redundancy, novelty and significance of each topic. These results are used in the sentence selection process. Josef and Karel's work focuses on novelty summary and it is based on a set of topic-related documents. Our work is running on external web pages returned by a search engine for our queries.

3 Enriching the Explanation of Entities with Web Pages from Outside an Enterprise

3.1 Outline

Given one entity named N, the explanation of it consists of two parts. One part is from the Intranet, and another is from the Internet. Taking use of the method mentioned in [12],we get the internal part automatically. Now the problem is how to get the external part explanation by exploiting the internal part. It can be viewed as a three-stage process.

First, we extract snippets from internal web pages of each entity. The approach in our prior work [12] is used. After we have the internal explanation of an entity, we pick up some key words in it. These key words are related to the entity and they have high expression to the entity. We mix these key words with the entity name as queries. A web search engine is employed to retrieve various contexts in which the query occurs. We then get these contexts as an initial Internet source set.

Second, we need to filter the initial source set which we get in the first step. Though the pages are returned by giving the entity name with its related key words as queries to the search engine, the relations between pages and the entity are all not very high. So we filter the initial source set to pick the pages which have high relationship with the entity. Similarity value is computed in this step.

Third, we get a page set whose cells are all high associated with the entity in the second step. However, not all external pages represent different meaning with the internal ones. A single semantic meaning can be expressed in several ways. In this paper, we present an efficient similarity measuring algorithm to identify the different contexts that denote a particular semantic meaning. We choose the novelty external snippets to supplement the internal snippets.

3.2 Generating Explanation for Entities

In [12], we proposed a methodology to automatically select the best snippets for entities as their initial explanations. This method consists of two steps. First, it focuses on extracting snippets from a given set of internal web pages for each entity. This work starts from a seed sentence, a snippet grows up by adding the most relevant neighbor sentences into it. The relevant sentences are chosen by the Snippet Growth Model. In this model, a distance function and an influence function are used to make decision. Secondly, select the best snippets for each aspect of an entity. In this process, we use Latent Dirichlet Allocation method and K-means approach with data preprocessing to cluster snippets and choose the center one as the final snippet in the explanation. The combination of all the selected snippets serves as the primary explanation of the entity.

3.3 Retrieving External Pages

As we described in Section 3.2, [12] proposed an approach to select the best snippets for entities as their initial explanations. All these snippets are extracted from the internal web pages of an enterprise. This work should be done before the tasks mentioned in this paper as a prepared task.

A web search engine is used to get relational external web pages of an entity. Though quantity of web pages are growing larger and larger, the existed web search engine can provide most of relational information of your queries on the web. It is a convenient and effective tool to search Internet information.

The most important problem we need to consider is what the queries are. The pages we require are related to the entity we prepare to describe. The simplest query is the entity name itself. However one word always has different meanings. Even though the query is a famous person's name, a web search engine will return some pages which have this name but have no relationship with the famous person. If we just use the entity name as the query, we will do much more work on filtering initial information we get. In another side, different queries returns different results. Web search engine returns much more abundant pages if it is provided more than one query.

To retrieve pages for an entity, we use the following type of queries: A, A B. A is the entity name and B is the words we extract from the initial explanation of A. We use Google to search Internet information. In the search rule of Google, A B, this type means in the result pages, they must have both A and B.

We use a simple method to extract key words from an initial explanation. The Stanford POS tagger[1] is used to tag the part of speech of all the words in this explanation. Stanford POS tagger is a log linear tagger. It processes passages in a fast speed. We get the words which has the sign "/NN" or "/NNP". Words with tag "/NN" are all none in the explanation while the ones with "/NNP" are the names of a person or an enterprise. For example, consider a segment of an entity explanation shown in Figure 2, the black words are key words we extract to be B in the query.

> **Mark Hurd** exercised $1.37 million in options less than two weeks before **hp** disclosed its spying tactics in a **SEC filing** and congressional investigators want to know why.

Fig. 2. A segment of explanation of the entity "Mark Hurd" (the words in bold are the key words that will be put into the queries)

We download the whole result pages which the web search engine return to us rather than the snippets provided by the search engine along with the search results. The reason is, typically, a snippet contains a window of text selected from a document that includes the queried words, but is not comprehensive to provide enough contexts to do the next process in our work. So, though the download task is time consuming, we need to download the whole page from the web. While we collect URLS for queries using the procedure described above, we remove duplicate ones. The download task is collateral. It can save much time.

[1] http://nlp.stanford.edu/software/tagger.shtml

3.4 Computing Correlation Value

In Section 3.3, we get an initial set of pages. These pages are the result which Google returns with our queries consisting of the entity name and key words. Though all pages in this set contain our queries, there must be some ones are not really associated with the entity. So we need to filter this page set further. The work we should do is identifying whether one page from Internet describes the internal entity which as a part of queries. Because the entity name is very short and usually it just has one word, it is hard to compute correlation value between entity name and one page. In the prepared processing, we get entities, explanation and snippets of each entity. So we transform word-passage correlation into snippets-passage correlation. The reason we do not use the explanation is the explanation is always short, but we have thousands pages in the initial set, while the number of snippets is large.

Computing correlation of passages is a task in various problems such as clustering passages. Cosine similarity value is a measure of similarity between two vectors of n dimensions by finding the cosine of the angle between them. It is often used to compare documents in text mining and approximate correlation of two passages. But it should be noticed that we have thousands of snippets and thousands of external pages. It must take a long time to compute correlation between each snippet and each page.

LSA [2] is a theory and method for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of text. The LSA produces measures of word-word, word-passage and passage-passage relations as a practical method for the characterization of word meaning. These measures are well correlated with human cognitive phenomena along with association or semantic similarity. So, LSA can closely approximate human judgments of meaning similarity between words to objectively predict the consequences of overall word-based similarity between passages. In LSA, representing a text as a matrix in which each row stands for a unique word and each column stands for a text passage or other context. Each cell contains the frequency of the word in its row which appears in the passage denoted by its column. LSA has the ability to model human conceptual knowledge. It is assessed as a simulation of word-word and passage-word relations found in lexical priming experiments. Passages are all based on words. And word collects can represent one passage. However, LSA needs multiple passages to compute correlation value of passage-word. And it is a time-consuming process.

The TF-IDF [7] weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. This weight is often used in information retrieval and text mining. According to the TF-IDF method, the importance of a word to a document increases proportionally to time it occurs in this document while decreases by the frequency of the word in the collection. Variations of the TF-IDF value scheme are used by search engines as an important tool in scoring and ranking a document's relevance with a given query.

In this paper, we use a statistical method which can be seen to be a variant of TF-IDF weight to extract key words from internal snippets of an entity we have already had through the prepared processing. These key words generate a key word list. This list expresses main idea or main latent semantic meaning of snippets for an entity. In other words, this word list describes the entity.

The computing correlation between entity and external web pages algorithm consists of following three steps.

Step1: Given an entity $\boldsymbol{N}$, its primary explanation and all of its snippets, retrieve for a page set $\boldsymbol{C}$ according to the procedure described in Section 3.3.

Step2: For each entity, we gather its snippets as a collection. And then use TF-IDF to compute word values in this corpus. The common TF-IDF scheme is that all snippets of all entities from Intranet are viewed as the whole passage collection. And snippets of one entity are viewed as one document. TF value is computed on the set of one entity snippet corpus while IDF weight is on the whole entities snippet collection. However, TF value is considered only in this method. We compute this weight separately by entity. Every snippet corpus of one entity is independence. A snippet corpus is formalized as a vector of words. We compute the TF value of none word in this vector except for the high frequency words like week, day and so on.

After we have TF-IDF value, we choose words with the value above β. These words form a word list. And at the same time, we have a vector $\boldsymbol{v}$. It has n dimensions. Each cell of it means the TF-IDF value of a word in the word list. We normalize this vector by amount 1.

Step3: We use the vector we get after step 2 to compute correlation value with external pages. We go through each page, count the frequency of words in the vector. Each page will get a point s.

$$s = \frac{\sum_{i=1}^{n} f_i * v_i}{\sum_{i=1}^{n} f_i} \tag{1}$$

In the formula 1, f_i is the frequency of word i in a page and v_i is the value of this word in vector $\boldsymbol{v}$. If the point s is above θ, we will put this page into the set $\boldsymbol{C'}$. $\boldsymbol{C'}$ is a new set whose cell has high relevance with the entity.

Our correlation computing algorithm has two parameters, β and θ. While in Step 2, we have two choices to process. These problems are solved experimentally, as explained later in Section 4. It is noteworthy that the proposed approach in this section turns the problem between entity-page into word-page. And the words-page correlation can effectively approximate the correlation of entity and pages.

3.5 Enriching Snippets

Pages in set $\boldsymbol{C'}$ have high relevance with the entity. There must be redundancy in this corpus. Our purpose is to exploit external web pages to supplement internal explanation. The final explanation is neat and has no duplicate. So we need to throw snippets that are duplicates of internal explanation from the collect $\boldsymbol{C'}$.

Divergence between internal explanation and elements in corpus ***C'*** is estimated. KL-Divergence, first introduced by Kullback and Leibler in 1951 [5]. In information theory, it has been used to indicate the divergence between two distributions. While it also emphasizes the distance between two passages, has a good effect on exploring the text difference. Unlike cosine similarity, KL-Divergence does not satisfy symmetry and triangle inequality.

However it should be noticed that the cell in our work is one snippet not one document. Therefore we consider a better way to choose novelty snippets to supplement the old explanation. Our method consists of two parts:

First, we compute the cosine similarity value between the internal explanation and the external pages in set ***C'***. We choose the ones with the cosine value above γ. These pages are put into a set I. Because we have limited the number of the collect ***C'***, so this process will not take a long time. In this step, we choose the pages which are definitely related to the entity we need to describe.

Second, we use the method in [12] to cut pages into snippets. Then we compute the cosine value between the snippet and the internal explanation. When the value is greater than δ, we believe it may be too close to the explanation. The snippets with value below δ will be added into a new set ***F***.

Finally, we get a snippet set. These snippets have high correlation with the entity while the low similarity with the internal explanation.

4 Experiments and Results

4.1 The Data

Our experiments are based on a real data set which is from HP Labs China. The data set consists of more than 560,000 internal web pages. These pages are all from the Intranet of HP. We collect information of employees which occur in the data set from the HP Labs China. From these employee names we pick out the top 500 frequent appearance ones as entities to do the internal snippet extracting and explanation generating. And then we choose 16 entities out of 500 to do experiments in this paper. These entities stand for all 500 entities in accordance with the following rules: the degree of famous, ranking in 500 entities, the length of the explanation, the quality of the explanation.

In the first step mentioned in Section 3.3, we get initial page set from Internet for each entity. We download 75,041 pages from urls Google returns to our queries of 16 entities. We select 5 entities from 16 to be a small data set which is used to determine the parameters in our approach.

4.2 Computing Correlation Values

In Section 3.4, we proposed an algorithm to compute a correlation value between external web pages and internal entity in order to filter external pages which has no relevance with the internal entity. We experiment it on the small data set which consists of 5 entities to set down the threshold β and θ. The threshold β is set to 1 at last. β is a threshold to confine number of key words of one entity.

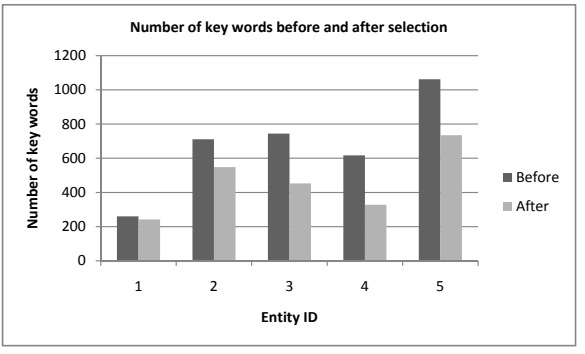


Fig. 3. Number of key words before and after selection, $\beta = 1$

When the count is above β, word is viewed as enough ability to join in the set to stand for an entity. β equals 1 means words with none speech occur more than one time in the snippet corpus of an entity has certain description ability to an entity. Figure 3 shows the word number in the key word list. It indicates there are enough words to do the next step after filter.

θ is a threshold to limit the number of external pages. When the value s is greater than θ, pages are believed to have high correlation with the entity. We set the threshold θ at 0.001 at last. It is a small number. That is because key words are chosen from all internal snippets. This key word list is large to one document from web. So the relevant value must not be big. However, this small value makes sure that the pages have high relationship with the entity and the number of pages is not too large. Figure 4 shows the number of external web pages on the small data set after this step.

From Figure 4, we can find that entity with ID 5 has 0 page after filtering. This entity name is "Young Iris". We randomly check 200 from 1431 pages. The

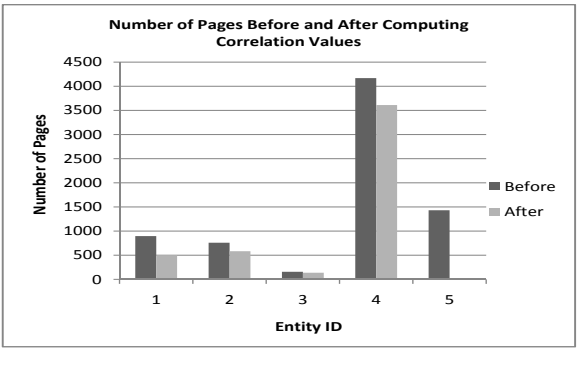


Fig. 4. Number of pages before and after computing correlation values between internal entity and external web pages. $\theta = 0.001$

result shows these pages are not relevance with the internal entity "Young Iris". It indicates our method to compute relevance between entity and pages works.

4.3 Snippet Enrichment

In this section we carry on our supplement snippet extracted method to choose snippets. In the first step mentioned in section 3.4, we compute the cosine similarity between internal explanation and external pages. Lots of variants of cosine value schema are used in estimate relevance between two documents. In our experiments, we use a sentence as a cell to compute this value not the whole document. This method guarantees the similarity between two documents is not low. So in our experiments the parameters γ is set at 0.5. These values ensure external page has high relevance with the entity. After filtering the pages further, we extract snippets from pages. And the threshold δ is set to be 0.4. When a snippet with similarity value in this step below 0.4 is chosen into the final snippet corpus.

5 Evaluation

In this section we will evaluate the results of our method. We judge the quality of explanation from external web pages by their correlation with the entity which they describe and their novelty.

11 entities along with their internal snippets, explanation, their external web pages, external snippets and external explanation are provided to a judge. The judgement result is shown in Table 1. This table has five columns. The first column is the Entity ID. The second one shows how many aspects of an entity are described in its internal explanation. The third column gives us the number of aspects in external explanation. The next shows how many relevant snippets in external explanation. The last one shows how many novelty aspects in external explanation which are not mentioned in internal explanation.

Table 1. Experiment results with our approach

Entity ID	Internal Explanation	External Explanation		
		total	relevant	novel
0	4	7	5	2
1	6	7	7	5
2	3	9	8	6
3	6	8	7	4
4	5	7	5	4
5	5	10	10	5
6	4	9	8	4
7	5	10	8	8
8	1	8	5	5
9	2	5	5	4
10	6	6	5	5

Experiments indicate the quality of external explanation depends on the internal explanation we generated. These entities are chosen as representations of all entities. However, table above shows that external explanation of any entity we generate by our method have certain description to entities. And they have ability to supplement the internal ones.

Through the evaluations above, we can see that no matter the computing correlation algorithm or the supplement snippet extracted method are promising, our methods are working well on real data set.

6 Conclusions

In this paper, we propose an idea to use external web pages to supplement initial explanation for entities we generate in our previous work [12]. In our previous work, we can automatically generate and select the appropriate snippets for entities as their initial explanations based on the information from within the enterprise. However, the snippets only from the Intranet information are not sufficient. To supplement these snippets with the information on external web pages, we focus on two problems. First, we compute the correlation between the external pages and the entities. We propose an algorithm to approximate this value and get a page set with high relevance for each entity. Second, choose the novel and valuable ones to supplement the existing snippets. We present a comprehensive method instead of computing KL-divergence only. Experiments on a real data set indicate our method is highly effective.

As our future work, we will try to conduct our proposed method on the entities other than employees, saying products of an enterprise.

References

1. Berger, A.L., Mittal, V.O.: Ocelot: A system for summarizing web pages. In: SIGIR (2000)
2. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science 41 (1990)
3. Goldstein, J., Kantrowitz, M., Mittal, V., Carbonell, J.: Summarizing text documents: Sentence selection and evaluation metrics. In: SIGIR (1999)
4. Knight, K., Marcu, D.: Statistics-based summarization - step one: Sentence compression. In: AAAI (2000)
5. Kullback, S., Leibler, R.A.: On information and sufficiency. Annals of Mathematical Statistics 22(1), 79–86 (1951)
6. Radev, D.R., Fan, W., Zhang, Z.: Webinessence: A personalized web-based multidocument summarization and recommendation system. In: NAACL Workshop on Automatic Summarization (2001)
7. Salton, G., Buckley, C.: Term weighting approaches in automatic text retrieval. Information Processing and Management 24(5), 513–523 (1988)
8. Steinberger, J., Jezek, K.: Update summarization based on novel topic distribution. In: Proceedings of the 9th ACM Symposium on Document Engineering (2009)

9. Wan, X., Yang, J., Xiao, J.: Collabsum: exploiting multiple document clustering for collaborative single document summarizations. In: SIGIR (2007)
10. Wan, X., Yang, J., Xiao, J.: Manifold-ranking based topic-focused multi-document summarization. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (2007)
11. Wan, X., Yang, J., Xiao, J.: Towards an iterative reinforcement approach for simulataneous document summarization and keyword extraction. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (2007)
12. Wang, Y., Zhao, L., Zhang, Y.: Magiccube: choosing the best snippet for each aspect of an entity. In: CIKM (2009)

Query Processing with Materialized Views in a Traceable P2P Record Exchange Framework

Fengrong Li[1] and Yoshiharu Ishikawa[2]

[1] Graduate School of Information Science, Nagoya University
[2] Information Technology Center, Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan
lifr@db.itc.nagoya-u.ac.jp, ishikawa@itc.nagoya-u.ac.jp

Abstract. Materialized views which are derived from base relations and stored in the database are often used to speed up query processing. In this paper, we leverage them in a traceable peer-to-peer (P2P) record exchange framework which was proposed to ensure reliability among the exchanged data in P2P networks where duplicates and modifications of data occur independently in autonomous peers. In our proposed framework, the provenance/lineage of the exchanged data can be available by issuing tracing queries. Processing for tracing queries was based on the "pay-as-you-go" approach. The framework can achieve low maintenance cost since each peer only maintains minimum amount of information for tracing. However, the user must pay relatively high query processing cost when he or she issues a query. We consider that the use of materialized views allows more efficient query execution plans. In this paper, we focus on how to incorporate query processing based on materialized views in our framework.

1 Introduction

With the advance of high-performance of computer and the wide spread of high-speed network, *peer-to-peer* (*P2P*) network has become a new paradigm for information sharing. However, unlike the traditional client-server architecture, a P2P network allows a peer to publish information and share data with other peers without going through a separate server computer. It brings us a critical problem; since copies and modifications of data are performed independently by autonomous peers without a specific central server control, it is difficult to determine how data is exchanged among the peers and why the data is located in a peer.

To interpret database contents and to enhance the reliability of data, the notion of *data provenance* is considered very important. Practical and theoretical methodologies for describing, querying, and maintaining provenance information have been proposed, for example in [5,6]. The importance of understanding the process by which a result was generated is fundamental to many real life applications, such as fields of bioinformatics and archaeology. Without such information, users cannot reproduce, analyze or validate processes or experiments.

H.T. Shen et al. (Eds.): WAIM 2010 Workshops, LNCS 6185, pp. 246–257, 2010.

Based on the background, to ensure the reliability of data exchanged in P2P networks, we have proposed a *traceable P2P record exchange framework* in [16,18]. In the framework, a *record* means a tuple-structured data item that obeys a predefined schema globally shared in a P2P network. An important feature of the P2P record exchange framework is that it is based on the database technologies to support the notion of *traceability*. User can trace the lineage of target record by issuing a tracing query. Processing for tracing queries was described in [19].

In this paper, we focus on the issue on how to improve query processing performance by using materialized views. The remainder of this paper is organized as follows. Section 2 describes the fundamental framework of the proposed P2P record exchange system. Section 3 shows the current strategy for query processing and analyzes its problems. Section 4 explains how materialized views are used to improve efficiency in our context for query processing and discusses the maintenance of materialized views. Section 5 reviews the related work. Finally, Section 6 concludes the paper and addresses the future work.

2 Traceable P2P Record Exchange Framework

Figure 1 shows the overview of the traceable P2P record exchange framework proposed in [16,18], but some terminologies are revised.

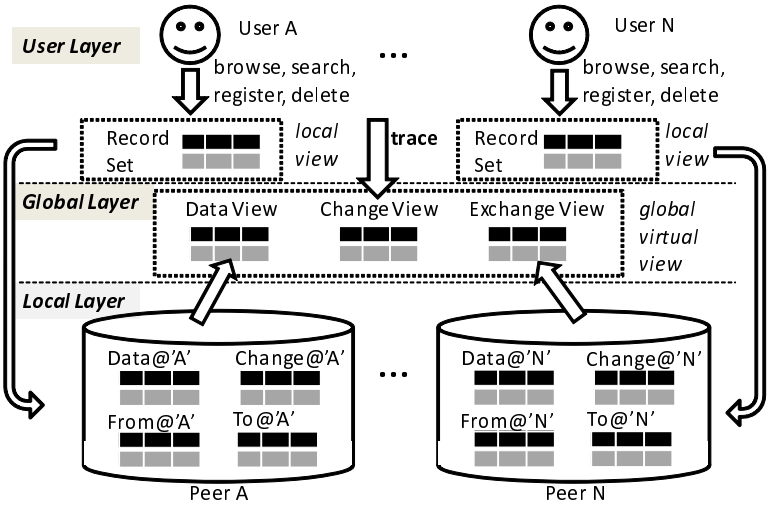


Fig. 1. Traceable P2P Record Exchange Framework

In the framework, we assume that each peer corresponds to a user and maintains the records owned by the user. Each record has the same structure, which is defined by a predefined schema that globally shared within the network. The framework has the following main features:

- In our P2P record exchange framework, every peer can act as a provider and a searcher. Records are exchanged between peers and peers can modify, store, and delete their records independently. Each peer has its own record set in the *user layer*, but their contents are not the same. Peers can behave autonomously and exchange records when required. A peer can find desired records from other peers by issuing a query.
- For reliable data sharing in a P2P network, we want to know, for example, the original creator of the given record and the path of the record in circulation before reaching the current peer. We assume that each peer maintains its own relational tables for storing record exchange and modification histories in the *local layer* and facilitates traceability. All the information required for tracing is maintained in distributed peers. When a tracing query is issued, the query is processed by coordinating related peers in a distributed and recursive manner.
- For ease of understanding and writing tracing queries, we provide an abstraction layer called the *global layer* which virtually integrates all distributed relations and a datalog-like query language [2] for writing tracing queries in an intuitive manner.

In the following, we briefly explain the three-layer model using an example.

User Layer. For the ease of presentation, we assume that each peer in a P2P network maintains a `Novel` record set that has two attributes `title` and `author`. Figure 2 shows three record sets maintained by peers A to C. Each peer maintains its own records and wishes to incorporate new records from other peers in order to enhance its own record set. For example, the record `(t1, a2)` in peer A may have been copied from peer B and registered in peer A's local record management system.

Peer A

title	author
t1	a2
t7	a6

Peer B

title	author
t1	a2
t5	a5

Peer C

title	author
t1	a1
t3	a3

Fig. 2. Record Sets in Three Peers

Local Layer. In the local layer, each peer maintains minimum amount of information that is required to represent its own record set and local tracing information. In our framework, every peer maintains the following four relations in its local record management system implemented using an RDBMS.

- `Data[Novel]`: It maintains all the records held by peer. Figure 3 shows `Data[Novel]` for peer A. Every record has its own record id for the maintenance purpose. Each record id should be unique in the entire P2P network. Note that there are additional records compared to Fig. 2; they are *deleted* records and usually hidden from the user. They are maintained for data provenance.

id	*title*	*author*
#A01	t1	a2
#A02	t6	a6
#A03	t7	a6
#A04	t3	a3

Fig. 3. `Data[Novel]`@'A'

from_id	to_id	time
–	#A02	...
#A02	–	...
#A02	#A03	...
–	#A04	...
#A04	–	...

Fig. 4. `Change[Novel]`@'A'

id	from_peer	from_id	time
#A01	B	#B02	...

Fig. 5. `From[Novel]`@'A'

id	to_peer	to_id	time
#A04	C	#C02	...

Fig. 6. `To[Novel]`@'A'

- `Change[Novel]`: It is used to hold the creation, modification, and deletion histories. Figure 4 shows an example for peer A. Attributes `from_id` and `to_id` express the record ids before/after a modification. Attribute `time` represents the timestamp of modification. When the value of the `from_id` attribute is the null value (−), it represents that the record has been created at the peer. Similarly, when the value of the `to_id` attribute is the null value, it means that the record has been deleted.
- `From[Novel]`: It records which records were copied from other peers. When a record is copied from other peer, attribute `from_peer` contains the peer name and attribute `from_id` has its record id at the original peer. Attribute `time` stores the timestamp information.
- `To[Novel]`: It plays an opposite role of `From[Novel]` and stores information which records were sent from peer A to other peers. Fig. 6 shows the `To[Novel]` relation of peer A.

Global Layer. Three virtual global views are constructed by unifying all the relations in distributed peers. Relation `Data[Novel]` in Fig. 7 expresses a view that unifies all the `Data[Novel]` relations in peers A to C shown in Fig. 2. The `peer` attribute stores peer names. Relation `Change[Novel]` shown in Fig. 8 is also a global view which unifies all `Change[Novel]` relations in a similar manner.

`Exchange[Novel]` shown in Fig. 9 unifies all the underlying `From[Novel]` and `To[Novel]` relations in the local layer. Attributes `from_peer` and `to_peer` express the origin and the destination of record exchanges, respectively. Attributes `from_id` and `to_id` contain the ids of the exchanged record in both peers.

Since recursive processing is required to collect historical information, our framework provides a modified version of *datalog* query language [2]. For example, the following query detects whether peer X copied the record `(t1, a2)` owned by peer A or not:

```
Reach(P, I1) :- Data[Novel]('A', I2, 't1', 'a2'),
                Exchange[Novel]('A', P, I2, I1, _)
Reach(P, I1) :- Reach(P, I2), Change[Novel](P, I2, I1, _), I1 != NULL
Reach(P, I1) :- Reach(P1, I2), Exchange[Novel](P1, P, I2, I1, _)
Query(I) :- Reach('X', I)
```

peer	id	*title*	*author*
A	#A01	t1	a2
A	#A02	t6	a6
A	#A03	t7	a6
A	#A04	t3	a3
B	#B01	t1	a1
B	#B02	t1	a2
B	#B03	t5	a5
C	#C01	t1	a1
C	#C02	t3	a3

Fig. 7. View `Data[Novel]`

peer	from_id	to_ id	time
A	–	#A02	...
A	#A02	–	...
A	#A02	#A03	...
A	–	#A04	...
A	#A04	–	...
B	#B01	–	...
B	#B01	#B02	...
B	–	#B03	...
C	–	#C01	...

Fig. 8. View `Change[Novel]`

from_peer	to_peer	from_id	to_id	time
C	B	#C01	#B01	...
B	A	#B02	#A01	...
A	C	#A04	#C02	...

Fig. 9. View `Exchange[Novel]`

Datalog is so flexible that we can specify various types of queries using the three global views; please refer to [16,18] for the detail.

3 Query Processing Approach and Problem Statement

In our original framework, every peer only maintains the minimum amount information for tracing in the local layer. In order to process tracing queries which are described in datalog using virtual global views, it is necessary to transform the given query to suit the organization of the local layer.

According to the mapping rules [16], the example query in Section 2 can be mapped as follows. The symbol @ is a *location specifier* which indicates the location (peer id) of relation in the local layer.

```
Reach(P, I1) :- Data[Novel]@'A'(I2, 't1', 'a2'),
                To[Novel]@'A'(I2, P, I1, _)
Reach(P, I1) :- Reach(P, I2), Change[Novel]@P(I2, I1, _), I1 != NULL
Reach(P, I1) :- Reach(P1, I2), To[Novel]@P1(I2, P, I1, _)
Query(I) :- Reach('X', I)
```

In [17], we compared two major strategies for datalog query execution, the *seminaive method* and the *magic set method*, in our context. Both of them are based on the "pay-as-you-go" approach [14] for tracing. It means that we need to aggregate the required historical information from the distributed peers when a tracing query is issued from a user; the user should pay the cost when he or she traces information.

The advantage is that this method is simple and there is no wastefulness in respect of the storage cost. However, when we perform the query processing, since it

is necessary to spread a requirement to all the related distributed peers. We should trace the path along the process that the records were exchanged. Generally, the cost for query processing is relatively large. To solve this problem, we consider to construct materialized views which are often used to speed up query processing.

4 Query Processing with Materialized Views

4.1 Definitions of Materialized Views

Materialized views play important roles in databases [12]. In our case, all of the materialized views do not store all of the information in the whole P2P network. They are only used to store the information at the peers in the target scope. A *target scope* is determined by a materialized view maintenance policy. In this paper, we assume that materialized views at each peer store the related information to k hops. Hops means the number of peers involved in a record exchange. For example, if peer A received a record from peer B and peer B received the record from peer C, peer C is in two hops from peer A in terms of the record.

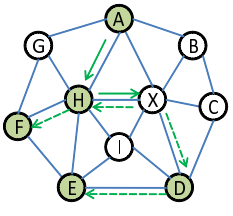


Fig. 10. Target Scope for Peer X ($k = 2$)

For instance, Fig. 10 shows the target scope of the materialized views for peer X in case of $k = 2$. A solid line arrow shows the route of the record that has been copied. A dotted line arrow shows the copy route of the offered records. Peer A, D, E, F, and H are the peers in the scope of the materialized views at peer X since there were record exchanges between them and peer X directly or indirectly in two hops. In this paper, we assume that each peer maintains four materialized views: MVData, MVChange, MVFrom, and MVTo. Each of them corresponds to Data, Change, From, and To relations in the local layer, respectively.

Like a tracing query, materialized views are expressed in datalog and using Data, Change and Exchange virtual views in the global layer. In the following, we show the representations of them in case of $k = 2$.

MVData@X: MVData is a materialized view that stores the exchanged records owned by peers which locate in the target scope. MVData stored at peer X can be described as below:

```
RData1(P, I1, T, A, H) :- Data[Novel]('X', I2, T, A),
                          Exchange[Novel](P, 'X', I1, I2, _), H=1
RData1(P, I1, T, A, H) :- RData1(P, I2, T, A, H), Change[Novel](P, I1, I2, _)
RData1(P, I1, T, A, H) :- RData1(P1, I2, T, A, H1),
                          Exchange[Novel](P, P1, I1, I2, _), H=H1+1, H<=2
RData2(P, I1, T, A, H) :- Data[Novel]('X', I2, T, A),
                          Exchange[Novel]('X', P, I2, I1, _), H=1
RData2(P, I1, T, A, H) :- RData2(P, I2, T, A, H), Change[Novel](P, I1, I2, _)
RData2(P, I1, T, A, H) :- RData2(P1, I2, T, A, H1),
                          Exchange[Novel](P1, P, I2, I1, _), H=H1+1, H<=2
RData(P, I, T, A) :- RData1(P, I, T, A, H)
RData(P, I, T, A) :- RData2(P, I, T, A, H)
MVData@X(P, I, T, A) :- RData(P, I, T, A)
```

The variable H is used to count the number of hops. The maximum value of H should be set to be equal to k. `RData1` is the collection of the records related to the copied record owned by peer X in two hops. `RData2` stores the information that which peer copied the records owned by peer X in two hops and also stores the contents of records in these peers. `RData1` and `RData2` are finally combined into `MVData@X`. Peer X executes the program and stores DataMV@X as a materialized view.

MVChange@X: The following is the definition of the materialized view for storing the change histories of the exchanged records:

```
RPeer1(P, I1, T, A, H) :- Data[Novel]('X', I2, T, A),
                          Exchange[Novel](P, 'X', I1, I2, _), H=1
RPeer1(P, I1, T, A, H) :- RPeer1(P, I2, T, A, H), Change[Novel](P, I1, I2, _)
RPeer1(P1, I1, T, A, H) :- RPeer1(P2, I2, T, A, H1),
                           Exchange[Novel](P1, P2, I1, I2, _), H=H1+1, H<=2
RChange1(P, I1, I2, _, H) :- RPeer1(P, _, _, _, H), Change[Novel](P, I1, I2, _)
RPeer2(P, I1, T, A, H) :- Data[Novel]('X', I2, T, A),
                          Exchange[Novel]('X', P, I2, I1, _), H=1
RPeer2(P, I1, T, A, H) :- RPeer2(P, I2, T, A, H), Change[Novel](P, I1, I2, _)
RPeer2(P, I1, T, A, H) :- RPeer2(P1, I2, T, A, H1),
                          Exchange[Novel](P1, P, I2, I1, _), H=H1+1, H<=2
RChange2(P, I1, I2, _, H) :- RPeer2(P, _, _, _, H), Change[Novel](P, I1, I2, _)
RChange(P, I, I1, _) :- RChange1(P, I, I1, _, H)
RChange(P, I, I1, _) :- RChange2(P, I, I1, _, H)
MVChange@X(P, I, I1, _) :- RChange(P, I, I1, _)
```

Both MVData and MVChange will increase the cost for storage and management in the operation and maintenance of materialized views. But they not only are used to improve query processing efficiency, but also can be used for the recovery of the lost data when some peer disappeared suddenly.

MVFrom@X: This materialized view stores the information of copied records from other peers in two hops.

```
FromH(I2, P, I1, H) :- Data[Novel]('X', I2, T, A),
                       Exchange[Novel](P, 'X', I1, I2, _), H=1
FromH(I2, P, I3, H) :- FromH(I1, P, I2, H), Change[Novel](P, I3, I2, _)
FromH(I2, P, I1, H) :- FromH(I1, P, I2, H1),
                       Exchange[Novel](P, P1, I1, I2, _), H=H1+1, H<=2
MVFrom@X(I, P, I1, H) :- FromH(I, P, I1, H)
```

MVFrom is effective for tracing the record retrospectively. Management cost is negligible though the storage cost is additionally needed. Path information caching and the record insertion to the materialized view can be executed one when the record is exchanged.

MVTo@X: A similar idea can be applied to the side of the To relation. This materialized view stores the information that which peer copied records from peer X in the scope of two hops .

```
ToH(I2, P, I1, H) :- Data[Novel]('X', I2, T, A),
                     Exchange[Novel]('X', P, I2, I1, _), H=1
ToH(I2, P, I3, H) :- ToH(I1, P, I2, H),
                     Change[Novel](P, I2, I3, _), I3 != NULL
ToH(I2, P, I1, H) :- ToH(I1, P1, I2, H1),
                     Exchange[Novel](P1, P, I2, I1, _), H=H1+1, H<=2
MVTo@X(I, P, I1, H) :- ToH(I, P, I1, H)
```

The management cost for the MVTo is not negligible. For example, in Fig. 10, when the record is copied from peer D to peer E, it is necessary to pass on the information of the copy event to peer X. In other words, not only peer D and E but also peer X should be involved in the transaction of the copy of the record from peer D to peer E. This becomes an additional overhead to some extent.

4.2 Query Processing with Materialized Views

Materialized views stored locally as base relations can improve query performance through query rewrites. We illustrate their use in query processing using our example.

Based on the materialized views, we can rewrite the example query in Section 2 as below:

```
Reach(P, I1) :- Data[Novel]@'A'(I2, 't1', 'a2'),
                To[Novel]@'A'(I2, P, I1, _)
Reach(P, I1) :- Reach(P1, I2), MVTo[Novel]@'A'(I2, P, I1, _)
Reach(P, I1) :- Reach(P, I2), MVChange[Novel]@P(I2, I1, _), I1 != NULL
Reach(P, I1) :- Reach(P1, I2), MVTo[Novel]@P1(I2, P, I1, _)
Query(I) :- Reach('X', I)
```

In this example, peer A wants to detect that whether peer X copied the record (t1, a2) or not. In the following, we describe the query processing referring to the Fig. 10.

In our original approach without materialized views, the query processing which is based on the seminaive method starts at peer A and the query fragments

id	to_peer	to_id	#hop
#A01	H	#H01	1
#H01	X	#X01	2

Fig. 11. Relation `MVTo` at Peer A

P	I
H	#H01
X	#X01

Fig. 12. Relation `Reach`

generated at peer A are first forwarded to peer H, and then peer H forwards them to Peer X. The query is executed in this way until it reaches the fixpoint.

With the materialized views, peer A can do the execution locally since peer X copied the record (t1, a2) in the target scope of peer A. Assuming that materialized view MVTo about the record (t1, a2) shown in Fig. 11 is stored at peer A. At peer A, notice that when the first rule is executed, the result (H, #H01) will become a tuple in `Reach` which is shown in Fig. 12. When the second rule is executed, a new tuple (X, #X01) should be inserted in `Reach`. Finally, when the last rule is executed, the #X01 as a result should return to the query. That is to say, peer X copied the record (t1, a2) from peer A.

The example indicates that the materialized views speed up the query processing. Like this case, with the materialized views, processing for queries which include recursive operation does not have to do the forward processing and queries can be answered immediately at the local peer. It is thought that the materialized views work well effectively in the query processing. Of course, processing for many tracing queries still needs forwarding operations until it reaches the fixpoint based on the seminaive method. If the query processing can not be done using materialized views, then it will be executed ordinarily as before.

4.3 Maintenance for Materialized Views

View maintenance which means the processing of updating a materialized view in response to changes to the underlying data. As we described above, materialized views can speed up query processing greatly. But they have to be kept up to date. If some of the base relations are changed, materialized views must be recomputed to ensure correctness of results to query processing. For maintaining general recursive views, [13] proposed the *DRed* (Delete and Rederive) algorithm that can handle incremental updates. However, the algorithm assumes a centralized environment, and it is quite costly to apply the algorithm in our context because the maintenance process is propagated among distributed peers. *Materialized view maintenance* problem in deductive databases was described in [12,21].

In our case, we can utilize a feature of our framework. Every update can be handled as a tuple insertion. Assume that the related peer in the first hop for peer X is peer Y, and peer Z is related peer in the second hop. Of course, related peers for peer Z in two hops are peer Y and peer X. Depending on the update types, a record is inserted in each of the following local relations and materialized views:

– record update in peer X: `Data@X`, `Change@X`, `MVData@Y`, `MVData@Z`, `MVChange@Y`, and `MVChange@Z`

- record modification in peer X: `Data@X`, `Change@X`, `MVData@Y`, `MVData@Z`, `MVChange@Y`, and `MVChange@Z`
- record deletion in peer X: `Change@X`, `MVChange@Y`, and `MVChange@Z`
- record copy from peer X to peer Y: `To@X`, `From@Y`, `Data@Y`, `MVData@X`, `MVData@Z`, `MVTo@X`, and `MVFrom@Y`

It means that there only exists *insertion* in the database updates in our framework. Materialized view maintenance for insertion can use the seminaive method for computation easily. When the fixpoint is found, the current incremental maintenance should be finished [13].

5 Related Work

Understanding provenance of documents is not a new problem. The importance of provenance goes well beyond verification. It is used in a wide range of fields, including data warehousing [7], uncertain data management [3,22], curated databases [5], and other scientific fields such as bioinformatics [4]. In this area, one of the well-known projects would be the *Trio* project at Stanford University, in which both of the uncertainty and lineage issues are considered [22]. Our research is devoted to the data provenance issue in P2P information exchange, where data provenance is important but there is few proposals for this topic.

There are a variety of research topics regarding P2P databases, such as coping with heterogeneities, query processing, and indexing methods [1]. One related project with our problem is the ORCHESTRA project [9,15], which aims at collaborative sharing of evolving data in a P2P network. In contrast to their approach, our research focuses on a simple record exchange scenario and does not consider schema heterogeneity. One of the features of our framework is to employ database technologies as the underlying foundation to support reliable P2P record exchange.

As proved in the *declarative networking* project [20], declarative recursive queries are very powerful in writing network-oriented database applications such as sensor data aggregation. In contrast to their approach, our focus is compact and understandable tracing query specifications.

Materialized views can be used to summarize, pre-compute, and replicate data. Maintenance for them is very important in database. We can find the recent survey about maintenance of materialized views in [11]. The incremental maintenance of views has received a lot of attention in database research, many incremental methods have been already proposed in the literature[8,10,21,23]. In all papers, only [12,21] described materialized view maintenance problem in deductive databases. In our research, for tracing queries, especially for the queries asking past histories, materialized views [12] are quite helpful to reduce query response time. For that purpose, we develop a query processing method which effectively uses materialized views and a view selection and maintenance method which considers the trade-off of cost and benefit.

6 Conclusions and Future Work

For the efficient query processing, data replication and caching are popular techniques. Considering practical requirements of tracing, we added some incorporate additional features and constructs to our fundamental P2P record exchange system. Although the storage and maintenance cost will increase, the query processing cost can be reduced.

In this paper, we described how to define materialized views and how to use them to improve query processing in our proposed P2P record exchange system. The maintenance of materialized views was also discussed. Nevertheless much work remains to be done. We need to consider how to take trade-off considering the total cost reduction. Several future research issues are summarized as follows:

- Full specification of complete query processing strategies: We need to enhance the strategies to handle more complex tracing queries. The effectiveness and limitation of the declarative language-based approach will become more clear.
- Fault-tolerance: In this paper, we omitted the issue of fault tolerance, but it is important for supporting P2P networks in which failure occurs frequently. We need to consider that how to use materialized views to do the data recovery for left peer in detail.
- Prototype system implementation and experiments: We are currently developing a prototype system of our P2P record exchange framework. Also, we started to construct a P2P network simulator that can be used for simulating our prototype system in a virtual P2P network. Their developments will have positive feedbacks to improve our fundamental framework.

Acknowledgments

This research was partly supported by the Grant-in-Aid for Scientific Research (#21013023, #22300034) from the Japan Society for the Promotion of Science (JSPS).

References

1. Aberer, K., Cudre-Mauroux, P.: Semantic overlay networks. In: VLDB (2005) (tutorial notes)
2. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
3. Benjelloun, O., Das Sarma, A., Halevy, A., Widom, J.: ULDBs: Databases with uncertainty and lineage. In: Proc. VLDB, pp. 953–964 (2006)
4. Bhagwat, D., Chiticariu, L., Tan, W.-C., Vijayvargiya, G.: An annotation management system for relational databases. In: Proc. VLDB, pp. 900–911 (2004)
5. Buneman, P., Cheney, J., Tan, W.-C., Vansummeren, S.: Curated databases. In: Proc. ACM PODS, pp. 1–12 (2008)

6. Buneman, P., Tan, W.-C.: Provenance in databases (tutorial). In: Proc. ACM SIGMOD, pp. 1171–1173 (2007)
7. Cui, Y., Widom, J.: Lineage tracing for general data warehouse transformations. In: Proc. VLDB, pp. 471–480 (2001)
8. Goldstein, J., Larson, P.-A.: Optimizing queries using materialized views: A practical, scalable solution. In: Proc. ACM SIGMOD, pp. 331–342 (2001)
9. Green, T.J., Karvounarakis, G., Taylor, N.E., Biton, O., Ives, Z.G., Tannen, V.: Orchestra: Facilitating collaborative data sharing. In: Proc. ACM SIGMOD, pp. 1131–1133 (2007)
10. Griffin, T., Libkin, L.: Incremental maintenance of views with duplicates. In: Proc. ACM SIGMOD, pp. 328–339 (1995)
11. Gupta, A., Mumick, I.S.: Maintenance of materialized views: Problems, techniques, and applications. IEEE Data Engineering Bulletin 18(2), 3–18 (1995)
12. Gupta, A., Mumick, I.S. (eds.): Materialized Views. MIT Press, Cambridge (1999)
13. Gupta, A., Mumick, I.S., Subrahmanian, V.S.: Maintaining views incrementally. In: Proc. ACM SIGMOD, pp. 157–166 (1993)
14. Halevy, A., Franklin, M., Maier, D.: Principles of dataspace systems. In: Proc. ACM PODS, pp. 1–9 (2006)
15. Ives, Z., Khandelwal, N., Kapur, A., Cakir, M.: Orchestra: Rapid, collaborative sharing of dynamic data. In: Proc. Conf. on Innovative Data Systems Research (CIDR 2005), pp. 107–118 (2005)
16. Li, F., Iida, T., Ishikawa, Y.: Traceable P2P record exchange: A database-oriented approach. Frontiers of Computer Science in China 2(3), 257–267 (2008)
17. Li, F., Iida, T., Ishikawa, Y.: ''Pay-as-you-go' processing for tracing queries in a P2P record exchange system. In: Zhou, X., Yokota, H., Deng, K. (eds.) DASFAA 2009. LNCS, vol. 5463, pp. 323–327. Springer, Heidelberg (2009), http://www.db.itc.nagoya-u.ac.jp/papers/2009-dasfaa-li-long.pdf
18. Li, F., Ishikawa, Y.: Traceable P2P record exchange based on database technologies. In: Zhang, Y., Yu, G., Bertino, E., Xu, G. (eds.) APWeb 2008. LNCS, vol. 4976, pp. 475–486. Springer, Heidelberg (2008)
19. Li, F., Ishikawa, Y.: Query processing in a traceable P2P record exchange framework. IEICE Transactions on Information and Systems E93-D(6) (2010) (accepted for publication)
20. Loo, B.T., Condie, T., Garofalakis, M., Gay, D.E., Hellerstein, J.M., Maniatis, P., Ramakrishnan, R., Roscoe, T., Stoica, I.: Declarative networking: Language, execution and optimization. In: Proc. ACM SIGMOD, pp. 97–108 (2006)
21. Staudt, M., Jarke, M.: Incremental maintenance of externally materialized views. In: Proc. VLDB, pp. 75–86 (1996)
22. Widom, J.: Trio: A system for integrated management of data, accuracy, and lineage. In: Proc. Conf. on Innovative Data Systems Research (CIDR 2005), pp. 262–276 (2005)
23. Zhuge, Y., García-Molina, H., Hammer, J., Widom, J.: View maintenance in a warehousing environment. In: Proc. ACM SIGMOD, pp. 316–327 (1995)

Author Index