GraphMiner: A Structural Pattern-Mining System for Large Disk-based Graph Databases and Its Applications*

Wei Wang† Chen Wang† Yongtai Zhu† Baile Shi† Jian Pei‡ Xifeng Yan¶ Jiawei Han¶

† Fudan University, China, {weiwang1, chenwang, bshi}@fudan.edu.cn

‡ Simon Fraser University, Canada, jpei@cs.sfu.ca

¶ University of Illinois at Urbana-Champaign, USA, {xyan, hanj}@cs.uiuc.edu

ABSTRACT

Mining frequent structural patterns from graph databases is an important research problem with broad applications. Recently, we developed an effective index structure, *ADI*, and efficient algorithms for mining frequent patterns from large, disk-based graph databases [5], as well as constraint-based mining techniques. The techniques have been integrated into a research prototype system—*GraphMiner*. In this paper, we describe a demo of *GraphMiner* which showcases the technical details of the index structure and the mining algorithms including their efficient implementation, the mining performance and the comparison with some state-of-the-art methods, the constraint-based graph-pattern mining techniques and the procedure of constrained graph mining, as well as mining real data sets in novel applications.

1. BACKGROUND

Mining frequent graph patterns is an interesting research problem with broad applications, including mining structural patterns from chemical compound databases, plan databases, XML documents, web logs, citation networks, and social networks. Several efficient algorithms were proposed in the previous studies [1, 2, 3, 4, 7, 6], ranging from mining graph patterns, with and without constraints, to mining closed graph patterns.

Most of the existing methods assume implicitly or explicitly that *the graph databases to be mined are not very large so that they can be held in (main) memory* and *the graphs in the databases are relatively simple*. That is, either the databases or the major part of them can fit in memory, and the number of possible labels in the graphs is small. Under the main memory-based assumption, the mining is CPU-bounded instead of I/O-bounded. Then, those algorithms focus on effective heuristics to prune the search space.

However, when mining a large graph database that cannot fit in memory, an algorithm may have to scan the database and navigate the graphs repeatedly. The mining becomes I/O-bounded and thus a main memory-based algorithm may face serious difficulties. Efficient and scalable mining of massive disk-based graph databases remains challenging for large applications, such as mining large collections of XML documents and large databases of chemical structures.

Recently, we developed an effective index structure, *ADI* (for ADjacency Index), as well as efficient algorithms using *ADI* for mining frequent structural patterns from large, disk-based graph databases [5]. The general idea is as follows. First, instead of proposing new heuristics for search, we developed the simple and effective *ADI* index structure for large graph databases. Once an *ADI* index is built, efficient graph-pattern mining uses the index only and does not need to access the original database. Furthermore, the major data access operations in frequent graph-pattern mining are identified and the information requirements are categorized according to their frequency and cost. The related information is organized into three layers of the *ADI* index structure, namely edge table, linked list of graph-ids, and adjacency information. Adaptive to the available main memory, the layers of the *ADI* structure can be accommodated in main memory or on disk. The three-layer design makes the *ADI* structure highly flexible and scalable for graph-pattern mining. The extensive performance study in [5] verifies the efficiency and the scalability of the *ADI* index structure and the mining algorithms. *ADI-Mine*, the adapted *gSpan* [7] using the *ADI* structure, can be substantially faster than *gSpan* when mining graph databases that can be held in memory; and can mine graph databases containing millions of graphs which are far beyond the capability of the original main memory-based *gSpan*.

In addition to the *ADI* index structure and the *ADI-Mine* algorithm published in [5], we also developed efficient algorithms for mining graph patterns with various constraints. Constrained graph-pattern mining makes the mining more focused and thus significantly improves the effectiveness and the efficiency.

The above techniques have been integrated into one research prototype system—*GraphMiner*. We also applied *GraphMiner* to mine real data sets in several novel appli-

*The first four authors are supported in part by National Science Foundation of China (NSFC) grants No. 60303008 and 69933010. The fifth author is supported in part by NSF Grant IIS-0308001, a President's Research Grant and a startup grant in Simon Fraser University. The last two authors are supported in part by NSF Grant IIS-0209199/0308215. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

SIGMOD'05, June 14-16, 2005, Baltimore, MD, USA.

cations, including mining topic-oriented webpage graphs in web mining and mining chemical structure databases for computational chemistry. The mining results are interesting in applications and indicate the high potential of graph mining in those applications.

The remainder of the paper is organized as follows. In Section 2, the technical features and the architecture of *GraphMiner* are discussed. We outline the demo in Section 3.

2. GRAPHMINER

GraphMiner is based on a series of techniques we are developing for graph mining. Some preliminary results were published in [5]. The technical novelty of *GraphMiner* in the current version can be summarized in the following four aspects.

2.1 ADI: An Effective Index

As reported in [5], *ADI* is an effective index structure for large disk-based graph databases. An example of an *ADI* index is shown in Figure 1.

Information about graphs is organized into three layers in an *ADI* index, namely the edge table, the linked list of graph-ids and the adjacency information. The edge table is stored as a sorted list in main memory if it fits, or otherwise a B^+-tree on disk. The linked list of graph-ids and the adjacency information are stored on successive disk pages and blocks. As analyzed in [5], an *ADI* index facilitates the major and costly operations in frequent graph-pattern mining.

The construction and the search of projected graph databases can also be implemented efficiently using the *ADI* index. Since the information is stored in three layers, the layers can be accommodated either in main memory or on disk adaptive to the size of available main memory. This design introduces the great flexibility and scalability of mining both small and large graph databases, in main memory or on disk.

2.2 Efficient Graph-Mining Algorithms Based on ADI

The *ADI* index is a general index structure for graph-pattern mining. Many previous graph mining algorithms can be adapted using the *ADI* index. For example, in [5], we showed how algorithm *gSpan*, a state-of-the-art graph-pattern mining method, can be adapted easily using *ADI* index. Recently, we also adapted some other graph-pattern mining algorithms, such as *Closegraph* [6], for mining closed frequent graphs.

2.3 Constraint-based Graph Mining

As indicated by many previous studies, frequent pattern mining may return a large number of patterns. To make the mining focusing and effective, constraints are often used to confine the pattern search.

We have developed a systematic approach to pushing various useful constraints into the graph-pattern mining. The categories of the constraints that we have implemented so far are as follows.

1. *Element constraints.* A user can specify the constraints on the vertices and/or the edges in the patterns. For example, a user can request certain vertices/edges must appear in the patterns and exclude some others.
2. *Size constraints.* A user can specify the range of the number of vertices and/or the number of edges in the patterns.
3. *Model constraints.* A user can specify that the patterns must be super-patterns or sub-patterns of some given graphs.
4. *Aggregate constraints.* The vertices and edges of the graphs may carry some attributes, such as the length for an edge and the weight of a vertex. A user can specify constraints on the aggregates on the patterns. The aggregates can be defined based on the attributes of vertices and edges in the patterns.

GraphMiner provides a graphical user interface for users to easily specify the constraints, and employs efficient algorithms to push the constraints into the mining.

2.4 Mining Procedure Management and Graph-Pattern Analysis

GraphMiner provides a user with the freedom to control the mining procedure, such as monitoring and managing the available main memory for *GraphMiner*, and specifying and revising the constraints. To help analyze and understand the patterns mined, *GraphMiner* provides a user interface to browse and analyze the graph patterns. A user can also query the graph patterns to select the interesting ones.

2.5 The Architecture of *GraphMiner*

The architecture of *GraphMiner* is shown in Figure 2. The core of the system is the *GraphMiner mining engine*. It has two major functions: building an *ADI* index for graph databases and mining frequent graph patterns using an *ADI* index with respect to a specification (i.e., the minimum support threshold and/or some constraints).

The mining engine provides a standard interface for graph mining algorithms. The interface is basically the related function calls for graph data access. The mining engine uses the *API* index to handle the data access requests efficiently. Following the interface, a graph mining method such as an implementation of *gSpan* and *Closegraph* using the standard function calls can be plug in the mining engine directly. By this standard interface, the previously developed graph-pattern mining algorithms can be adapted effectively.

A *constraint composer* with a user interface is provided for specifying constraints for the mining. The constraints are compiled and then transferred to the mining engine.

The mining procedure is managed by the *control panel*. From the panel (a screen snapshot in Figure 3), a user can specify the size of main memory that should be used by *GraphMiner* and the *ADI* index to be used, as well as select the mining algorithm.

The patterns mined will be stored in a *pattern database*. A user can query the database and browse the patterns using the *pattern browser*. Several tools are provided in the pattern browser. For example, a user can browse patterns containing some specific vertices and edges. A user can also query the other patterns that are similar to a given pattern.

3. ABOUT THE DEMO

GraphMiner V1.0 has been implemented, and can be showcased in four aspects.

First, in the demo, *we will explain the technical details of the* ADI *index structure and the mining algorithms, includ-*

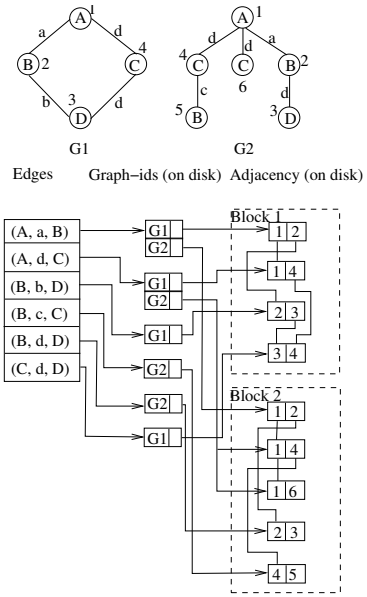

Figure 1: An *ADI* index structure.

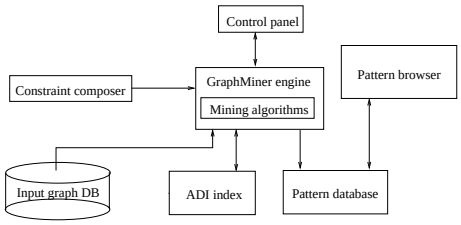

Figure 2: The architecture of *GraphMiner*.

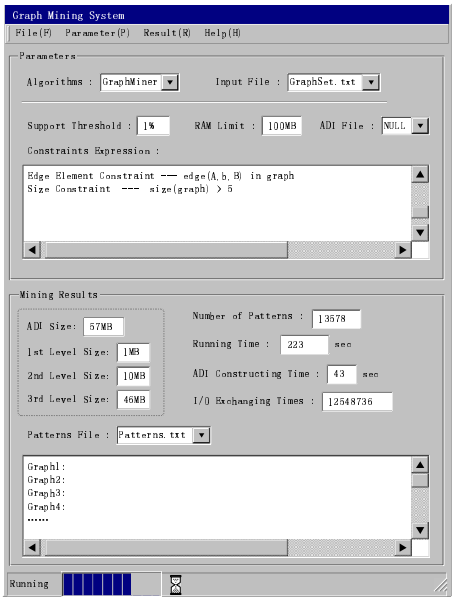

Figure 3: The control panel of *GraphMiner*.

ing their efficient implementation. We will share with the audience the lessons and experience that we have learned from the research and development of graph mining.

We will demonstrate the standard interface of *GraphMiner* to graph mining algorithms, and show how such an interface can support future development of new graph-pattern mining algorithms and new index structures. In addition to algorithm *ADI-Mine* that was published in [5], we will also illustrate how the *ADI* index structure can be used in mining other kinds of graph patterns, such as frequent closed graph patterns.

Second, *we will analyze the performance of mining various graph database, in main memory or on disk, using* ADI *index.* Particularly, we will break down the cost of mining into small pieces so that the efficiency of the index structure for graph mining as well as the challenges and potential opportunities can be understood. We will also compare the performance of the algorithms using and not using *ADI* index.

As another interesting issue, we will discuss how the layers of an *ADI* index can be accommodated in main memory and on disk, and their effect on the performance. Moreover, we will demonstrate the control of the tradeoff between efficiency and main memory usage, as well as the scalability on large disk-based graph databases.

Third, *we will explain the categories of constraints that* GraphMiner *can handle in the current version.* The constraint-pushing techniques will be disclosed and discussed. For selected constraints, we will demonstrate how those constraints can improve the effectiveness and efficiency of the mining.

The audience will be encouraged to try the constraint-based graph-pattern mining by writing their own constraints and browsing and analyzing the patterns mined. We hope that the experience of *GraphMiner* will help to attract more attention and interest for future research and development of graph mining and index for data mining in general.

Last, *several real data sets will be brought to the demo, including a topic-oriented webpage graph database and a chemical structure database.* A series of selected patterns mined from the real databases will be illustrated. The performance of *GraphMiner* on mining real data sets and in some example applications will be discussed.

By demonstrating the mining of real data sets and its applications, we hope to help the audience understand the values, the challenges and the potential of mining graph databases.

4. REFERENCES

[1] C. Borgelt and M.R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *ICDM'02*, Maebashi TERRSA, Maebashi City, Japan, Dec. 2002.

[2] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *PKDD'00*, pages 13–23, Lyon, France, Sept. 2000.

[3] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *ICDM'01*, pages 313–320, San Jose, CA, Nov. 2001.

[4] N. Vanetik, E. Gudes, and S.E. Shimony. Computing frequent graph patterns from semistructured data. In *ICDM'02*, Maebashi TERRSA, Maebashi City, Japan, Dec. 2002.

[5] C. Wang, W. Wang, J. Pei, Y. Zhu, and B. Shi. Scalable mining of large disk-base graph databases. In *KDD'04*, pages 316–325. ACM Press, 2004.

[6] X. Yan and J. Han. Closegraph: Mining closed frequent graph patterns. In *KDD'03*, Washington, D.C, 2003. ACM Press.

[7] Y. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *ICDM'02*, Maebashi, Japan, December 2002.