Pruthak- mining and analyzing graph substructures
Swapnil Shrivastava, Kriti Kulshrestha, Pratibha Singh and Supriya N. Pal
Centre for Development of Advanced Computing (CDAC)
68, Electronic City, Hosur Road,
Bangalore, India, 560100
+91 80 2852 3300
{swapnil,kriti,pratibha,supriya}@cdacbangalore.in

ABSTRACT
In many scientific and commercial domains, graph as a data structure has become increasingly important for modeling of sophisticated structures. In the past few years, there has been a sharp increase in research on mining graph data. We had proposed a unified framework for graph mining and analysis of extracted substructures, which was then an unattended task. Pruthak, a graph mining tool is developed based on this proposed framework. The tool provides preprocessing, frequent substructure discovery, dense substructure extraction and visualization techniques for graph representation of data. In this paper we discuss the approach taken in design and implementation of Pruthak. We then talk about our study on the Digital Bibliography & Library Project (DBLP) dataset for mining and analyzing substructures using this tool. The study results have demonstrated the intended correctness and usability of the tool.

Categories and Subject Descriptors
D.2.8 [Database Management]: Database Applications– data mining

General Terms
Design

Keywords
Graph mining, Graph visualization, Frequent substructure discovery, Dense substructure extraction, Graph preprocessing.

1. INTRODUCTION
Pruthak, a graph mining tool is developed in Java using open source technologies. It is developed for mining and analysis of substructures in graph representation of data. Pruthak in the Sanskrit language means to decompose and extract. The tool design and development is based on our previous work of graph mining framework [23], maximum clique detection [22] and informative graph visualization [24]. It is developed with an intention to resolve the following issues:

• In several scientific and commercial domains, interesting knowledge can be mined from relationships between entities. Existing data mining tools like Weka [38] perform mining tasks on data available in the form of single flat file or relation. Their use for relational knowledge extraction results in lossy knowledge discovery.

• Researchers have designed several efficient algorithms for mining various substructures (subgraphs) within the graph. These algorithms can be useful for mining relational data in several domains. But they are still not available as tools or libraries for usage.

• Several graph visualization tools and techniques exist which mainly provide graph drawing functionalities. Some of the tools provide structural measures and/or statistical measures for analysis. However they do not support informative visualization which according to our past experience is also important for in-depth analysis of the graph data.

In the absence of a much needed unified framework for mining and analyzing data from applications in unconventional domains, the graph mining framework [23] was proposed. Pruthak is developed based on this proposed framework. The tool currently consists of five components viz. graph preprocessing, graph database, dense substructure extraction, frequent substructure discovery and graph visualization.

• The graph preprocessing module integrate the data from heterogeneous sources, transform them to an appropriate graph format for analysis and cleans these graphs if required before storing them to the graph database.

• The graph database stores the graph representation of data in the underlying model. The graph database provides input to the frequent substructure discovery and dense substructure extraction modules.

• The frequent substructure discovery module finds frequent substructures using efficient algorithms like FSG [17], gSpan [27] and SPIN [13]. For a given labeled graph transaction dataset D, a graph g is a frequent substructure, if its support (number of times g is subgraph in D) is greater than the specified minimum support threshold minsup. By discovering frequent substructures we find commonalities or associations among related entities in the graph transaction dataset. Using frequent substructure module we can find the result of the queries like “Which author’s paper gets frequently published in a conference?” in DBLP database or “Which director/actor pair works frequently?” in IMDb database.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MLG’10, July 24-25, 2010 Washington, DC, USA.
Copyright © 2010 ACM I978-1-4503-0214-2/10/07... $10.00
Dense substructures are extracted using efficient dense substructure extraction algorithms like Trawling [16], Shingling [11] and Connection Subgraphs [10] in the dense substructure extraction module. Dense substructure is a collection of vertices such that, many or sometimes all has edge between them. A few common dense substructures are identified by name like bipartite core and maximum clique. Dense substructures arise frequently in real world analysis of graphs and represent clusters within the graph. In social network, dense substructure extraction module of the proposed framework can answer queries like “Who are the friends or co-workers of Mr A?” or in DBLP database “Which authors share co-authoring relationship?”

Graph visualization module will enable the user to analyze by direct, interactive and informative visual exploration of data from the graph database as well as discovered frequent substructures and dense substructures.

The rest of the paper is structured as follows. In section 2 we shall first discuss the background and motivation behind the graph mining tool-Pruthak. Section 3 describes the approach taken in design and implementation of core functionalities of the tool. The details of our study conducted on DBLP dataset using tool is mentioned in section 4. Section 5 summarizes the paper and in section 6 we briefly talk about the future work.

2. BACKGROUND AND MOTIVATION

Recent research in frequent substructure discovery has resulted in several efficient algorithms for mining frequent substructures. These algorithms are mostly either extension of Apriori based itemset mining algorithm [1] or they adopt pattern growth methodology [12]. AGM [14] is Apriori based approach to mine the frequent substructures and association rules from the given graph dataset. The discovery of frequent substructures consists of two steps. In first step it generates frequent substructure candidates while the frequency of each candidate is checked in the second step. The frequent candidate generation method in AGM is vertex based that increases the substructure size by one vertex in each iteration. AGM had practical efficiency for real world problem but needed further investigation to be done on its computational efficiency. A computationally efficient Apriori based algorithm named FSG [17] was defined for finding all frequent subgraphs in large databases. Like AGM, in FSG too the frequent substructure discovery consists of two steps but here the candidate generation is edge based that increases the substructure size by one edge in each iteration. gSpan [27] is a novel algorithm which discovers frequent substructures without generation of candidates. It is based on pattern growth methodology, which intends to extend the patterns from a single pattern directly. SPIN[13] discovers maximal frequent subgraphs i.e. subgraphs that are not contained in any other frequent subgraphs. This reduces computational resources. The pattern growth approach algorithms provide better memory utilization and scalability as compared to Apriori based algorithms.

There are several dense substructure extraction algorithms as a result of recent research in web link structure mining. Trawling algorithm [16] is a procedure for automatically identifying fine grained communities within the graph. The algorithm identifies small complete bipartite graphs as signature of communities, and then expands those signatures to find the full community. Shingling Algorithm [11] extracts dense substructure in large graphs. The algorithm starts with finding small patterns with common out neighbors. The algorithm extracts dense subgraphs by repeated application of this procedure that makes dense subgraphs denser and sparse subgraphs sparser. These algorithms process the entire graph to extract all the dense subgraphs. An interactive algorithm is defined [10] which takes query consisting of two vertices from user and returns a connection subgraph or dense subgraph that best captures the relationship between the two vertices. There are several algorithms and applications available for maximum clique detection [5]. These algorithms can be useful in several domains for analysis. Presently, for frequent substructure discovery or dense substructure extraction from a graph, individual algorithms need to be implemented by the user. It is cumbersome to implement separate algorithms for each individual analysis.

A large body of graph visualization literature has been published in the last two decades. Several graph drawing algorithms [7] [21] [26], book chapters [9] and surveys [25] are available. There are currently a plethora of open source and commercial libraries, frameworks and tools available for graph visualization and analysis. Graphviz [39] provides several graph drawing layouts, web and interactive graphical interfaces, auxiliary tools, libraries, and language bindings. Tulip [2] is designed to enable manipulation and visualization of huge graphs. Java Universal Network/Graph Framework (JUNG) [19][32] is an open-source library, which provides a common framework for graph/network analysis and visualization. JGraph [31] is an open source graph visualization library written in Java Swing. But it provides only basic visualization functionalities like graph drawing. JGraph Layout Pro Package [41] and mxGraph [42] are commercially available products of JGraph. Open Graph Drawing Framework (OGDF) [8] is an open source, feature rich and self-contained C++ class library for the automatic layout of diagrams. The Graph Visualization Framework (GVF) [33] is a set of design patterns and approaches that can serve manipulation and visualization of graph structure. aSee [36] reads a textual graph specification in GDL format and automatically calculates a customizable graph layout. Tom Sawyer Software [45] provides high performance visualization, layout, and analysis softwares for graph structure. These softwares are commercially available in Java, ActiveX, .NET and C++. It also provides perspective software to develop strategic corporate applications and layout assistant product to seamlessly integrate automatic layout into the Microsoft Visio environment. Graph Magics [40] is a tool purely for graph theory. It also provides graph generator tool and rich collection of implementation of different algorithms from graph theory. There are some visualization tools, libraries and techniques which are application specific. The goVisual [44] software libraries provide a complete collection of layout styles primarily for visualizing UML class diagrams. yFiles [46] is an extensive Java class library that provides algorithms and components for analyzing, viewing, and drawing graphs, diagrams, and networks. DBdraw [4] software tool generates graphical representation of database schema. Visone [6] is a tool that integrates analysis and visualization of social networks. Pajek [3] is another tool for analysis and visualization of large networks. SONIVIS [34] is eclipse-based open source software for analysing and visualizing virtual information spaces such as Wiki or Social Network Sites. BabelGiraph [37] is a social network analysis, visualization and simulation tool. We studied the functionalities provided by above.
mentioned visualization tools and techniques. They take the input data in a specified format for example, GML and GraphML. Also all the tools provide different sets of visualization techniques. In order to use these tools for visualization, the output of dense substructure and frequent substructure discovery algorithms should be converted to respective input format. Since input format and visualization techniques vary across tools, this becomes a tedious task. Some of the tools provide structural measures and/or statistical measures for analysis. However they do not support informative visualization which according to our experience is also important for in depth analysis of graph data.

gMine\[20\] is a scalable, interactive graph visualization and mining system. It allows multiresolution visualization by partitioning large graph into hierarchies and an algorithm to extract substructures based on an initial set of target nodes. For DBLP dataset, the system finds “Which authors share co-authoring relationship?” However the system cannot mine frequent substructures like “Which author’s paper frequently gets published in a conference?”

DBLP (Digital Bibliography & Library Project) \[28\] is a computer science bibliography website consisting of more than one million records on computer science from various journals like VLDB, IEEE and ACM transactions. Recently DBLP launched DBLPVis \[29\], a new web-based access tool to the DBLP data set. DBLPVis extracts different relations between authors, conferences/journals and keywords from the DBLP dataset and also visualize them. But the tool is DBLP project specific and cannot be used for other similar applications.

With the growing number of applications in several unconventional domains there is a need for analysis of relational data. As discussed earlier, graph mining and visualization techniques can be useful in this. But these techniques aren’t available under one umbrella. Hence there is a need for tool which integrates functionalities required for mining and analysis of graph representation of data.

3. Approach
For current version of Pruthak, our focus is on the design and implementation of core functionalities of graph mining framework. The module wise design of these functionalities is discussed in this section.

3.1 Graph Preprocessing
The graph preprocessing functionalities can be categorized into four sub modules viz. data integration, data selection, data transformation and data cleaning.

3.1.1 Data Integration
The data to be analyzed can come from multiple heterogeneous data sources. For some real world applications like DBLP, dataset is available in the form of XML files. The operational system data to be analyzed mostly exist in relational database. The tool currently supports input data from relational database and XML file. XML parser extracts metadata (DTD and XMLSchema) of the input XML file. Database parser connects to specified input database and extracts its metadata viz table names, attribute names and referential integrity constraints. The extracted metadata is visualized using ERViewer for data selection.

3.1.2 Data Selection
The data selection technique of ER Viewer is as shown in figure 1. The ER Viewer visualizes extracted metadata in the left pane.

![Figure 1: ER Viewer data selection technique.](image-url)
For input data from relational database, relation is visualized as box and attributes as each stripe in the box. The connecting line represents referential integrity between attributes. The user can select entities/attributes and relationships by clicking the boxes and connecting lines respectively. The selected metadata is visualized in the right pane of ERViewer. The selected entities/attributes are displayed as nodes and referential integrity as edge between the related entities. The data transformation and cleaning operations are applied on the selected data.

### 3.1.3 Data Transformation

The selected data should be transformed to an appropriate format in order to facilitate subsequent graph mining tasks. Some algorithms like gMine substructure extraction algorithm require data in the form of correlation graph. The frequent substructure discovery algorithms like FSG requires data to be stored as labeled graph transaction datasets. The data selected using ER Viewer can be converted to graph transaction or correlation graph based on the type of analysis to be performed. Before applying frequent substructure discovery technique like FSG the data should be transformed to graph transaction dataset. For dense substructure extraction technique like maximum clique detection, correlation graph should be generated.

### 3.1.4 Data Cleaning

Data cleaning is important for both accurate analysis and cost effectiveness. The tool provides deduplication, attribute conversion and filling in the missing values techniques. The tool is designed to provide three techniques for deduplication viz. exact string matching, string similarity and relational similarity. But as of now only exact string matching technique is implemented. The data aggregation and summarization is supported by attribute conversion functionality of the tool. Conversion of numeric attribute to ordinal attribute and adding prefix/suffix to string are some of the techniques implemented for attribute conversion. The missing values are filled based on the related existing values in the dataset.

### 3.2 Graph Database

Input data after applying graph preprocessing techniques is stored in the graph database for subsequent mining tasks. It stores graph transaction dataset and correlation graph as labeled undirected graphs. The graph elements (nodes and edges) are defined to be of any of the two types: individual or aggregate. The aggregate element stores the element name. The individual element stores an element value. For ‘Author’ entity in DBLP dataset, ‘Author’ name will be a node of aggregate type and “Jiawei Han” value will be node of individual type. Element types are defined for flexible, informative and interactive analysis of the graph data. This is illustrated in section 4 of the paper. Element type is incorporated while storing graph transaction dataset and correlation graph in the graph database module.

A graph transaction dataset, \( D = \{ G_1, G_2, \ldots, G_t \} \) where \( t \) is the number of graph transactions. Figure 2 shows the data model for storing graph transaction dataset in graph database. The ‘graphTransaction’ relation stores the ‘graphId’, ‘node1Id’, ‘node1LabelId’, ‘node2Id’, ‘node2LabelId’ and ‘edgeLabelId’. The ‘graphId’ is unique for each graph. The ‘node1’ and ‘node2’ are endpoints of the edge ‘edge’ in a graph. ‘node1Id’ and ‘node2Id’ stores the entity/attribute value id of ‘node1’ and ‘node2’ respectively and references ‘nodeDetails’ relation. ‘node1LabelId’ and ‘node2LabelId’ stores the entity/attribute name id for ‘node1’ and ‘node2’ respectively and references ‘nodeLabelDetails’ relation. ‘edgeLabelId’ stores the relationship name id and references ‘edgeLabelDetails’ relation. Detailed information of entity/attribute represented by node can be extracted with the help of generalization relationship between ‘nodeDetails’ and input specific entity tables ‘Entity1’, ‘Entity2’… ‘Entityn’. For example, ‘Author’ is one of the entities selected from DBLP dataset for graph transaction generation. At the time of creation of graph transaction dataset ‘Author’ relation will be created which will store details of all the authors. It will be referenced by ‘graphTransaction’ relation. Same holds true for all the other selected entities/attributes.

![Figure 2: Graph transaction dataset model](image)

A correlation graph in our tool is an undirected weighted graph in which set of nodes belongs to same entity type and set of edges represents relationship amongst them. The weight of the edge between any two nodes is its degree of correlation. Degree of correlation \( \hat{\sigma} \) is the number of time the two nodes have correlated. The strength of correlation is measured in terms of \( \hat{\sigma} \). For any pair of nodes, the higher the value of \( \hat{\sigma} \), the stronger is the co-relationship. The correlation graph data model is as shown in figure 3. The ‘correlationGraph’ relation stores the endpoints of edge in node1Id and node2Id attributes. The degree of correlation \( \hat{\sigma} \) is also stored in this relation. The detailed information of attribute/entity represented by node is referenced from ‘nodeDetails’ relation. The data required for statistical analysis is stored in ‘trendDetails’ relation.

The graph database not only store data in a format suitable for subsequent analysis. But it also stores the data required for informative visualization and statistical analysis.

![Figure 3: Correlation graph model](image)
3.3 Frequent Substructure Discovery

In frequent substructure discovery module of the tool FSG [17] algorithm is implemented. FSG is an Apriori based approach to mine the frequent substructure from the given graph transaction dataset for specified minimum support threshold \( \minsup \). The input to this algorithm is a graph transaction dataset \( D \) stored in graph database and \( \minsup \). And the output will be frequent substructures discovered in \( D \) for the specified value of \( \minsup \).

3.4 Dense Substructure Extraction

Maximum clique detection algorithm [22] is a combination of breadth first search traversal and best-in heuristics. The algorithm also has functionality of maximum clique refinement process. The algorithm finds all the maximum cliques from a single large graph. We extended the algorithm to detect maximum cliques as well as star substructures from the correlation graph. The input to this algorithm is a single large correlation graph \( D \) and degree of correlation \( \partial \). The output of this module will be maximum cliques and start substructures detected in \( D \) for the specified value of \( \partial \).

3.5 Graph Visualization

The informative graph visualization [24] design is based on the visualization requirements of graph mining framework [23] and decision support automated refactoring [15] project. The visualization techniques for Pruthak are as shown in figure 4. Multi level visualization, view information, statistical analysis and structural analysis are main functionalities of this module. In this module we call JUNG and JFreeChart[30] APIs for graph drawing and statistical analysis respectively.

![Graph Visualization Diagram](image)

Figure 4: Informative graph visualization components in Pruthak

3.5.1 Multilevel visualization

The multi level visualization algorithm implemented by us currently provides two levels of visualization viz summary and detailed. It visualizes dense substructures extracted from large graph. This is done using expand-collapse mechanism. Each dense substructure is collapsed into a single node at summary level. The user can view corresponding dense substructure or cluster at detailed level by clicking on the summary node for expansion.

3.5.2 View Information

As discussed in section 3.2, the graph database module not only stores graph representation of input data but also details of graph elements. This helps the user to view additional information of graph elements when clicked. When a graph is selected for visualization, the corresponding detailed information also gets loaded. For example co-author graph is selected for visualization. If the user clicks the edge between two authors then their association period, list of publications will be displayed in the bottom pane.

3.5.3 Statistical Analysis

This provides traditional visualization in the form of bar chart, pie chart or numeric value for the given set of data. In current version of tool, when the user clicks an aggregate element the statistical information is shown in the form of bar chart using JFreeChart. Say an aggregate edge represents co-authoring relationship between two authors who have association of 10 years during which they have published 50 papers. When the user clicks the edge, it can view the number of publications of two authors over a period of 10 years in the form of bar chart.

3.5.4 Structural Analysis

We have designed our tool to provide structural measures at three levels viz element, substructure and overall graph. The tool currently provides algorithms to measure density, diameter, radius and clustering coefficient for overall graph. Also element measures available in the tool are betweenness centrality, degree, closeness centrality, average path length and characteristic path length.

The module also supports GML and GraphML format for portability. The graph in these formats can be read for visualization. Also the visualized graph can be saved in these formats. The mining results can be exported to JPEG format for report generation.

4. DBLP CASE STUDY

In order to illustrate the intended usage of tool, we conducted our study on DBLP dataset. In this study we used the tool for mining and analyzing substructures from DBLP dataset. DBLP is a computer science bibliography website consisting of more than one million records on computer science from various journals like VLDB, IEEE and ACM transactions. Every record belongs to a category viz. book, proceeding, article, paper, journal, www or thesis. We have conducted our study on paper-proceeding and article-journals categories. DBLP dataset is available for download in XML format. Ley [18] has given detailed description of convention and syntax rules of dblp.xml file. Based on it we have defined data model for DBLP database. We wrote scripts to transform DBLP data in XML format to the data model as shown in figure 5.

The graph preprocessing module of the tool converts the DBLP database to a format suitable for subsequent graph mining tasks. The transformation of data from DBLP database to graph transaction dataset using ERViewer is shown in figure 1. In order to find dense substructures, the graph preprocessing module of the tool is used to generate a single large co-author (correlation) graph from input DBLP database. A co-author graph is an undirected weighted graph in which set of vertices represents the authors and set of edges represent co-authoring relationship between the authors. Both co-author graph and graph transaction dataset are stored in graph database.
Next we applied FSG algorithm on graph transaction dataset by specifying minimum support threshold $\text{minsup}$ as 2. The input data was selected from graph transaction dataset in order to find “Which author’s paper gets frequently published in a conference in a year?” As a result to these queries we discovered several frequent substructures. One of the many output frequent substructures is shown in figure 6. The figure also demonstrates the need for element type in graph transaction dataset which was discussed in section 3.2. The element type gives the flexibility for finding frequent substructures from various perspectives. For same dataset we can find “Which author frequently writes long paper in a conference?” Without element type we will not get any frequent substructures for $\text{minsup}>1$ as all the graphs in graph transaction dataset will be unique. By adding element type we can find frequent substructure and also provide user with flexibility to query from different perspectives. To get the query result shown in figure 6, ‘Paper’ name is selected i.e. it is an aggregate node. Whereas for rest all the nodes values are selected i.e. they are all individual nodes. When the user clicks on aggregate node ‘Paper’ the statistical information is shown in the form of bar chart and detailed information is displayed in the bottom pane. The tool helps the user to perform comprehensive analysis on graph data.

**Figure 5:** Data model for DBLP database to store details for paper-proceeding and article-journal categories.

**Figure 6:** Frequent substructure (association) in DBLP dataset for $\text{minsup}=2$. 
We also executed maximum clique detection algorithm by specifying co-author graph and degree of correlation $\partial = 50$ as input, to find "Which authors share co-authoring relationship?" The output of the algorithm is as shown in figure 7. The maximum cliques along with star substructures represent the clusters of authors who usually write together. In these substructures the edges are of aggregate type. When the user clicks on a node the additional information related to author such as homepage and list of publications are displayed in bottom pane. When the user clicks the edge, the list of publications co-authored by authors represented by endpoints (vertices) of edge is displayed in the bottom pane. Also the bar chart is displayed in right pane to show the number of co-authored papers with respect to time. The user can also find structural measures for the substructures. We have verified the correctness of output results of both the mining techniques and found them to be valid.

5. SUMMARY
With the growing number of applications in unconventional domains there is a need to discover, visualize and analyze interesting patterns (substructures) from the relational data. The existing data mining tools and techniques cannot be used effectively for mining knowledge in such domains. Several graph mining algorithms and visualization tools exist for analysis of relational data. In the absence of much needed unified framework for mining and analyzing substructures from graph data, the graph mining framework was proposed. Pruthak is based on this framework and provides preprocessing, frequent substructure discovery, dense substructure extraction and visualization techniques for graph representation of data. The graph preprocessing module extracts data from XML file or relational database and transforms it into graph structure suitable for subsequent analysis. The graph database module stores the transformed data. The technique like FSG algorithm provided in frequent substructures discovery module discovers valid frequent substructures or associations. The dense substructure extraction module finds all the clusters in correlation graph. The graph visualization module visualizes the input as well as output results in order to facilitate analysis. The informative graph visualization techniques provided by the tool supports comprehensive analysis of data. Our study on DBLP dataset using the tool demonstrates that the availability of these techniques in single unified platform makes graph analysis simple, informative, and interactive. The tool can be used in several commercial and scientific domains such as operational system databases and research citations for analysis of relational data.
6. FUTURE WORK
In the current implementation of the tool, our focus was on implementation of core functionalities of the graph mining framework in order to illustrate its intended usage. The future work involves implementation of remaining functionalities of the framework, resolve performance issues and improve the look and feel of the existing tool. Subsequently it will be either hosted as independent tool on SourceForge.net or customized as graph mining plugin for Weka. Till then the tool executable will be available for download from our project webpage on http://www.cdacmumbai.in website. We are also exploring the possibilities of extending the tool for mining software repositories.

7. ACKNOWLEDGMENTS
The name Pruthak was suggested by Ms Urjaswala Vora, Research Scientist, CDAC Mumbai. The first author is grateful to her for suggesting the appropriate name and also for her timely valuable feedbacks.

8. REFERENCES


[28] http://dblp.uni-trier.de/xml/
[29] http://dblpVis.uni-trier.de
[34] http://sourceforge.net/projects/sonivis/

[37] http://wwwbabelgraph.org/
[38] http://www.cs.waikato.ac.nz/~ml/weka/
[40] http://www.graph-magics.com/