BVG Trust

J. Sci. Trans. Environ. Technov. 2015, 8(4) : 207-211

Scientific Transactions in Environment and Technovation

A Study on Frequent Subgraph Mining Algorithms and Techniques for Software Bug Localization

A. Adhiselvam[1], E. Kirubakaran[2] and R. Sukumar[3]

[1]PG Department of Computer Applications, S.T.E.T Women's College, Mannargudi, Tamilnadu
[2]STTP (Systems), Bharat Heavy Electrical Limited, Trichirappalli, Tamilnadu
[3]Department of Computer Science and Engineering, Sethu Institute of Technology, Kariapatti, Tamilnadu

Abstract

With increasing demand on the study of large amounts of structured data, graph mining has become an active and important subfield in data mining domain. Frequent subgraph mining, graph classification, graph clustering and graph matching are the common types of graph mining algorithms for graph data. Mining graph data has recently emerged as a promising area in the current research. Software Engineering (SE) graph data is particularly important for the graph data. SE graph data includes static or dynamic call-graphs. Call-graphs are formed from control flow of programs. Software bug localization is one important application of frequent subgraph mining algorithm in which the structure of the call-graph is studied in order to determine and isolate bugs in the program. This paper deals with survey of generic frequent subgraph mining algorithms and a survey of specific graph mining techniques for software bug localization (or bug detection) application and graph mining methodology. A compareson two generic algorithms such as gSpan and CloseGraph with respect to six factors namely graph representation, subgraph generation, algorithmic approach, frequency evaluation, input, and output has also been made. These two algorithms are most relevant for software bug localization.

Keywords: Graph Mining, Software bug localization, Frequent Subgraph mining, call-graphs

INTRODUCTION

Many scientific and commercial applications need patterns that are more complicated than frequent itemsets and sequential patterns and require extra effort to discover (Han and Kamber, 2000) . Traditional data mining algorithms such as frequent pattern mining, classification, clustering, and indexing have now been extended to graph scenario. As in the case of other data types such as multi-dimensional or text data, we can design mining problems for structure data. The structured data include chemical data, biological data, web data and software engineering data. These structured data can be easily represented as graph. In addition, many new kinds of data such as semi-structured data and XML (Aggarwal *et al*., 2007) can typically be represented as graphs. Graphs become increasingly complicated structures in modeling such as circuits, chemical compounds, protein structures, biological networks, social networks (Han and Kamber, 2000). The control flow of programs can also be modeled in the form of graph called call-graph (Cham *et al*., 2010). Graph mining extracts useful information from structured data that can be represented as graph. To analysis graph data, graph mining plays important role in data mining. Graph mining has been a popular research area in recent years because of numerous applications in bioinformatics, cheminformatics, software engineering, network analysis, image processing (Cham *et al*., 2010)

Corresponding Author :
adhiselvam@yahoo.com

A number of algorithms have been developed for graph mining problems such as frequent subgraph mining, graph clustering and graph classification. The frequent subgraph mining problem is particularly important for the graph domain, because the end-results of the algorithms provide an overview of the important structures in the underlying data set, which may be used for other applications such as indexing (Yan *et al*., 2003). Frequent subgraphs are useful at characterizing graph sets, discriminating different groups of graphs, classifying and clustering graphs, and building graph indices (Cham *et al*., 2010). The problem can be defined in two different ways depending upon the application domain. In the first case, we have a group of graphs, and we wish to determine all patterns which support a fraction of the corresponding graphs (Inokuchi *et al*., 2000; Kuramachi and Karpis, 2001; Venetik *et al*., 2002). In the second case, we have a single large graph, and we wish to determine all patterns which are supported at least a certain number of times in this large graph (Kuramochi and Karypis, 2001; Bringmann and Nijssen, 2008; Fielder and Borgelt, 2007).

The frequent subgraph mining problem can be defined as the process of finding those subgraphs from a given graph or a set of graphs which have frequent or multiple instances within the given graph or the set of graphs. There are two basic approaches to the frequent subgraph mining problem: the Apriori-based approach and the Pattern-growth approach (Han and Kamber, 2000). Hence, there exists many graph mining algorithms based on these approaches. In the first approach, the search for frequent graphs starts with graphs of small size, and

proceeds in a bottom-up manner by generating candidates having an extra vertex, edge, or path. Apriori-based algorithms have considerable overhead when joining two size-k frequent subgraphs to generate size-(k+1) graph candidates. In order to avoid such overhead, non-apriori-based algorithms have been developed, most of which adopt the pattern-growth methodology (Han and Kamber, 2000). Typical Apriori-based algorithms include AGM, FSG, and edge-disjoint path-join algorithm. Pattern-growth graph mining algorithms include gSpan, MoFa, FFSM, SPIN, and Gaston (Cham *et al.*, 2010).

To improve both software productivity and quality, software engineers are increasingly applying data mining algorithms in various software engineering tasks (Foxie *et al.*, 2009). Frequent subgraph mining algorithm is one such example. Some of the interesting software engineering tasks include how to invoke Application Programming Interface (API) methods provided by a complex library, bug detection, debugging and testing. SE data mainly concerns with various software development phases. SE data can be broadly categorized into sequences such as execution traces and static traces, graphs such as dynamic call-graphs and static call-graphs, and text such as bug reports, code comments, and documentation (Foxie *et al.*, 2009). Mining SE graph data poses several challenges. Example of SE graph data includes call-graphs as well as program-dependence graphs (Chang *et al.*, 2007). Call-graphs are formed from control flow of programs in which all methods, procedures and functions are defined as nodes, and relationship between these methods are defined as edges. Call-graphs are of two types namely (i) dynamic call-graph collected at run-time, and (ii) static call-graph extracted from source code. Software bug localization is one of the important applications of frequent subgraph mining algorithms from the perspective of software quality and productivity.

II. SURVEY OF ALGORITHMS AND TECHNIQUES

A wide range of algorithms on frequent subgraph mining were developed by many researchers. Frequent subgraph mining algorithms are usually generic in nature. These algorithms can be applied across various domains by tuning the core algorithm to the application requirement. The problem of frequent subgraph discovery has its roots in the early nineties with the formulation of the algorithm for market-based analysis (Agarwal and Srikant, 1994). Some of them are reviewed in this section.

Holder *et al.* (1994) developed the SUBDUE system, which uses minimum description length principle to discover substructure that compress the database and represent structural concepts in the data. This system uses pattern-growth algorithmic approach in frequent subgraph mining. Yan and Han (2002) designed new pattern-growth based frequent subgraph mining in graph datasets and proposed a novel algorithm called gSpan. Haun *et al.* (2003) proposed a novel pattern-growth subgraph mining algorithm called FFSM, which employs a vertical search scheme within an algebraic graph framework. FFSM uses adjacency matrix representation to store graph data. Nijssen and Kok (2004) proposed a pattern-growth based algorithm called GASTON, which uses quickstart principle. It uses hash table representation to store graph data. In the search for sub-structures, first paths are considered, then paths are transformed into trees and finally trees are transformed to graphs. Saigo *et al.* (2008) proposed an iterative mining method based on partial least squares regression. To apply PLS to graph data, a sparse version of PLS is developed first and then it is combined with a weighted pattern mining. Yan *et al.* (2008) proposed an efficient algorithm which mines the most significant subgraph pattern with respect to the objective function. Ranu and Singh (2009) proposed GraphSig, a scalable method to mine significant subgraph based on a feature vector representation of graphs.

Besides the frequent subgraph mining algorithms, constraint-based subgraph mining algorithms have also been proposed. Feida Zhu *et al.* (2007) proposed a new general framework called gPrune to incorporate all the constraints in such a way that they recursively reinforce each other through entire mining process. A frequent graph pattern with n edges can have 2^n frequent subgraphs, which is an exponential number (Cham *et al.*, 2010). To overcome this problem, closed subgraph mining and maximal subgraph mining algorithms were proposed. Yan and Han (2003) proposed a new algorithm called Close Graph which mines closed frequent sub patterns instead of mining the entire subgraph. Haun *et al.* (2004) proposed a new algorithm that mines only maximal frequent subgraphs. Maximal frequent subgraph is a subgraph that is not part of any other frequent subgraphs.

The current researches also focus on directed frequent subgraph mining. Termier *et al.* (2007) developed an algorithm called DIGDAG which aimed at discovering directed acyclic subgraphs. Li et al. (2009) proposed an algorithm to mine all the frequent labeled directed subgraphs.

There has been a rise in the development of application-centric algorithms which are most suited for certain applications and efficient in solving the problem associated with that domain (Varunkrishna *et al.*, 2011). Software bug localization is a natural application in frequent subgraph mining algorithms. The generic algorithms such as gSpan and Close Graph are well suited for software bug localization. Several algorithms have been developed in the context of software bug localization. There are two sorts of approaches namely

structural and frequency approaches. Structural approaches for bug localization can locate structure affecting bugs. Frequency approaches are used to locate frequency affecting bugs.

Liu *et al.* (2005) developed a novel method to classify the structured traces of program executions using software behavior graphs. By analyzing the correct and incorrect executions, they have made good progress at the isolation of program regions that may lead to the faulty executions. Dallmeier *et al.* (2005) proposed common method to localize defects. Di Fatta *et al.* (2006) presented a method to enhance fault localization for software systems based on a frequent pattern mining algorithm. This method is based on a large set of test cases for a given set of programs in which faults can be detected. Eichinger *et al.* (2008) investigated the utilization of call graphs of program executions and graph mining algorithms to approach this problem.

Eichinger *et al.* (2008) proposed a novel reduction technique for call graphs which introduces edge weights. They presented an analysis technique for such weighted call graphs based on graph mining. They mined weighted graphs with a combination of structural and numerical techniques. Ray-Yaung Chang and Andy Podgurski (200) presented a novel approach for revealing neglected conditions that integrates static program analysis and advanced data mining techniques to discover implicit conditional rules in a code base and to discover rule violations that indicate neglected conditions. David Lo *et al.* (2009) addressed software reliability issues by proposing a novel method to classify software behaviors based on past history or runs. With the technique, it is possible to generalize past known errors and mistakes to capture failures and anomalies. This technique first mines a set of discriminative features capturing repetitive series of events from program execution traces. It then performs feature selection to select the best features for classification. These features are then used to train a classifier to detect failures. Liu *et al.* (Year) proposed a new statistical model-based approach called SOBER which localizes software bugs without any prior knowledge of program semantics. Nguyen *et al.* (2009) developed Grou Miner algorithm which is applicable to object-oriented programming. It is used to discovering usage patterns of one or more objects or classes where objects or classes are represented in the form of labeled directed acyclic graphs. Parsa *et al.* (2010) presented a paper to extract dynamic behavioral graphs from different executions of a program and analyze them to find bug relevant sub-graphs. They have proposed a new formula to rank the edges based on their suspiciousness to the failure.

MINING METHODOLOGY

The methodology to mine SE graph data is very important. Tao Xie et al. (2008) presented mining methodology to mine SE data. This consists of five steps:

Step 1: Collect/investigate SE data to mine

Step 2: Determine the Software Engineering task to assist

Step 3: Preprocessing data

Step 4: Adopting/developing a mining algorithm

Step 5: Postprocessing or applying mining results

Software engineers can start with either a problem-driven approach or a data-driven approach, but in practice they commonly adopt a mixture of the first two steps. Third step involves extracting relevant data from the raw SE data. For example, call-graph is obtained from source code. The data are further processed by cleaning. The next step is to adopt or develop a graph mining algorithm. The final step transforms the mining algorithm results into an appropriate format required to assist the SE task.

In the applications perspective, several approaches have been proposed to localize bugs by means of call-graph mining. The approaches consist of three steps [1]:

Step 1: Deduction of call-graphs: They can be obtained by tracing program executions. Further, a classification of program executions as correct or failing is needed to find discriminative patterns.

Step 2: Reduction of call-graphs: It is necessary to overcome the huge sizes of call-graphs

Step 3: This step includes frequent subgraph mining and the analysis of resulting frequent subgraphs.

DISCUSSION

In this section, we first compare two generic algorithms such as gSpan and CloseGraph. Both algorithms use adjacency matrix representation, rightmost extension subgraph generation and pattern growth algorithmic approach. These three characteristics are important for execution of these algorithms. Frequency evaluation is another important characteristic in the frequent subgraph mining algorithms. It is the process of counting the number of occurrences of a subgraph in a large graph or set of graphs. gSpan and CloseGraph use Depth First Search lexicographic ordering for frequency evaluation. Both algorithms take a set of small graphs as input. The only difference between gSpan and CloseGraph is that first algorithm mines complete set of frequent subgraphs as output and second algorithm finds partial set of frequent subgraphs as output. The above details are summarized in Table I.

Secondly, we consider some specific algorithms for software bug localization problem. In (Difalla *et al.*, 2006), data mining techniques are used in proposed method to analysis data generated during program test. A frequent pattern mining algorithm is also used to identify frequent subtrees in successful and failing test executions.

Table I. Comparison of Generic Algorithms gSpan and CloseGraph with respect to six factors

Factors	Frequent subgraph mining algorithms	
	gSpan	CloseGraph
Graph Representation	Adjacency list	Adjacency list
Subgraph generation	Rightmost extension subgraph	Rightmost extension subgraph
Algorithm-based	Pattern-growth	Pattern-growth
Frequency evaluation	DFS lexico graphic order	DFS lexico graphic order
Input	A set of small graphs	A set of small graphs
Output	Complete set of frequent subgraphs	Partial set of frequent subgraphs

In (Frank Eichinger et al., 2008), a dynamic control flow approach is used for localization of noncrashing bugs. This novel technique used edge weights to represent call frequencies. In (Parsa *et al.,* 2010), a new ranking method is developed to analyze program dynamic graph. Both weighted and un-weighted graphs were analyzed with ranking method.

CONCLUSION

Frequent subgraph mining is one of the most challenging problems in domain of graph mining. With the new development in software methodology, Computing technology, hardware technology, processing power and storage space of computer, more efficient algorithms need to be developed. The existing frequent subgraph mining algorithms and techniques can be extended with effective techniques such as map reduce technique, statistical techniques and machine learning techniques. This paper provides a common platform to develop a new and efficient frequent subgraph mining algorithm and technique for software bug localization.

REFERENCE

Aggarwal,C. Ta, J. Feng,N. Wang,J. Zaki,M.J. and XProj. 2007. A Framework for Projected Structural Clustering of XML Documents, KDD Conference.

Agarwal R. and Srikant,R. 1994. Fast algorithm for mining association rules in large databases, *In Proceedings of the 20th International conference on very large data bases,* San Francisco, USA, pp. 887-499.

Bringmann,B. and Nijssen,S. 2008. What is Frequent in a Single Graph?, PAKDD Conference.

Charu C. Aggarwal and Haixun Wang. 2010. *Managing and Mining Graph Data,* Springer Publishing company, Incorporated.

Chang,R. Y. Podgurski, A. and Yang, J. 2007. Finding What's Not There: A New Approach to Revealing Neglected Conditions in Software, *International Symposium of Software Testing and Analysis,* ACM Press, pp. 163-173.

Dallmeier, V. Lingig, C. and Zeller, A. 2005. Lightweight defect localization for java, *Proc. of the 19th European conf. on Object-Oriented programming.*

David Lo, Hong Cheng, Jiawei Han, SiauCheng Khoo and Chengnian Sun. 2009. Classification of Software Behaviors for Failure Detection: A Discriminative Pattern Mining Approach, *KDD'09.*

Di Fatta, G. Leue, S. and Stegantova, E. 2006. Discriminative Pattern Mining in Software Fault Detection, *Proc. of the Int. Workshop on Software Quality Assurance (SOQUA).*

Frank Eichinger, Klemens Bohm, Matthias Huber. 2008. Improved Software Fault Detection with Graph Mining, Appearing *6th International Workshop on Mining and Learning with Graphs, Helsinki, Finland.*

Frank Eichinger, Klemens Bohm, Matthias Huber. 2008. Mining Edge-Weighted Call-Graphs to localize software Bugs, *Proc. of the European conf. on machine learning and principles and practice of knowledge discovery in databases.*

Feida Zhu, Xifeng, Jiawei Han, Philip Yu,S. 2007. gPrune: A Constraints based framework for graph pattern mining, PAKDD'07 *Proc. 11th Pacific-Asia conference on Advances in knowledge discovery and data mining.*

Fiedler,M. Borgelt,C. 2007. Support Computation for Mining Frequent Subgraphs in a single graph, *Workshop on mining and Learning with Graphs.*

HanJ. and Kamber, M. 2000. *Data Mining:Concepts and Techniques,* Morgan Kaufmann.

Haun, J. Wang, W. Prins, J. 2003. Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism, *Proc. 2003 Int. Conference of Data Mining,* pp. 549-552.

Haun, J. Wang, W. Prins, J. and Yang, J. 2004. SPIN: Mining maximal frequent subgraphs from graph databases, *Proc.2004 ACM SIGKDD Int. Conf. Knowledge Discovery in databases*, pp.581-586.

Holder, L. B. Cook,D.J. Djoko,S. 1994. Substructure discovery in the SUBDUE system, *Proc. AAAI workshop knowledge discovery in database.*

Inokuchi,A. Washio,T. and Motoda,H. 2000. An Apriori-based Algorithm for Mining Frequent Substructures from Graph Data, *PKDD Conference,* pages 13-23.

Kuramochi, M. and Karypis,G. 2001. Frequent Subgraph discovery, *ICDM conference,* pp. 313-320, Nov.

Li,Y. Lin, Q. Zhong, G. Duan, D. Jin, Y. Bi, W. 2009. A directed labeled graph frequent pattern mining algorithm based on minimum code, *3rd International*

conference on Multimedia and Ubiquitous Engineering, IEEE, pp. 353-359.

Liu, C. Yan, X. Yu, H. Han, J. and Yu, P.S. 2005. Mining Behavior Graphs for Backtrace of Noncrashing Bugs, *Proc. of the Int. Conf. on Data Mining* (SDM).

Liu, C. Yan, X. Fei, L. Han, J. Midkiff, S.P. 2005. *SOBER: Statistical model-based Bug Localization,* SIGSOFT Software Engineering Notes.

Nguyen,T. Nguyen,H. Pham,H. Al-kofahi,J. *2009. Graph-based mining of multiple usage patterns,* ACM, pp. 383-392.

Nijssen S. and Kok, J. 2004. A quickstart in frequent substructure mining can make a difference, *Proc. 2004 ACM SIGKDD international conference of knowledge discovery in databases,* pp. 647-652.

Parsa, S. Mousavian Z. and Vahidi-Asl, M. 2010. Analyzing Program Dynamic Graphs for Software Fault Localization, 5th *IEEE Int. Symposium on Telecommunications.*

Ray-Yaung Chang, Andy Podgurski, 2008. Discovering Neglected Conditions In Software By Mining Dependence Graphs, *IEEE Transactions on Software Engineering*, Vol. 34, No. 5.

Ranu, S. and Singh, A. K. 2009. GraphSig: A Scalable approach to mining significant subgraphs in large graph databases, *Proc. 2009 Int. Conf. Data Mining,* pp. 844-855.

Saigo, H. Kramer, N. and Tsuda, K. 2008. Partial least squares regression for graph mining, *Proc.2008 ACM SIGKDD Int. Conf. Knowledge Discovery in databases,* pp.578-586.

Termier, A. Tamada, Y. Numata, K. Imoto, S. Washio, T. and Higuchi, T. 2007. DIGDAG: a first algorithm to mine closed frequent embedded sub-DAGs, *Proc. mining and learning with graph workshop, Citeseer,* pp. 41-45.

Varun Krishna, N.N.R. Ranga Suri, and Athithan,G. 2011. *A Comparative survey of algorithms for frequent subgraph discovery*, *Current Science,* 100, No. 2, August.

Venetik,N. Gudes,E. and Shimony,S. E. 2002. Computing Frequent Graph Patterns from Semi-structured Data, IEEE ICDM Conference.

Xifeng Yan and Jiawei Han, 2003. CloseGraph: Mining Closed Frequent Graph Patterns, *Proc.2003 ACM SIGKDD Int. Conf. Knowledge Discovery in databases,* pp.286-295.

Yan,X. Yu,P. S. and Han,J. 2003. *Graph Indexing:* A Frequent Structure-based Approach, SIGMOD Conference.

Yan, X. and Han, J. 2002. gSpan: Graph-based substructure pattern mining, *International conference* of Data mining.

Yan, X. Cheng, H. Han, J. and Yu, P.S. 2008. Mining significant graph patterns by scalable leap structure, *Proc.2008 ACM SIGKDD Int. Conf. Knowledge Discovery in databases,* pp.433-444.