ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 3, March 2013

Algorithms for Frequent Subgraph Mining

Sujata J. Suryawanshi[1], Prof. Mrs. S. M. Kamalapur[2]

Computer Engg. Department, KKWECOE, Nashik[1]

Computer Engg. Department, KKWECOE, Nashik[2]

Abstract*:* Due to increasing number of complex objects, Data mining algorithms are facing the challenges. To model such complex object, Graph a natural data structure is used. Graph mining is an important research area within the domain of data mining. A graph is a general model to represent data and has been used in many domains like cheminformatics and bioinformatics. Mining patterns from graph data base is difficult task than mining pattern from data set, sequences or tree, because graph related operations, such as subgraph testing, generally have higher time complexity. This paper gives the comparative study of frequent subgraph mining algorithms. In this paper different issues are discussed like graph representation, searching strategy, and Graph Summarization.

Keywords: Data mining, graph mining, pattern summarization.

I. INTRODUCTION

Data mining is the procedure of extracting knowledge from raw data. In recent years, there has been an increased interest in developing data mining algorithms that operate on graphs. Data can also be represented by various means. Structured data and semi-structured data are naturally suited to graph representations. Graph mining is an important research area within the domain of data mining. Most important concept in Graph mining is to find frequent subgraph from graph database.
Frequent subgraph mining is the processor of extracting all frequent subgraphs from graph dataset who have occurrence count greater than or equal to the specified threshold. Following figure gives example of frequent subgraph mining, where figure 1(a) and 1(b) represents any two graphs and figure 1(c) gives frequent subgraph which is present in both graphs.

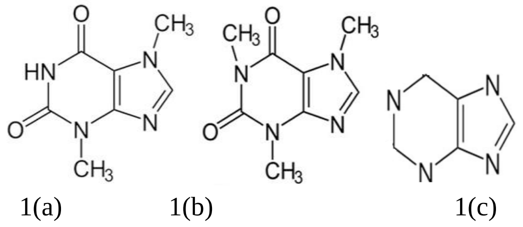

Fig. 1(a) Theobromine, 1(b) Caffeine and 1(c) Frequent subgraph**.**

The rest of the paper is structured as follows: Section II describes frequent subgraph mining algorithms, advantages and disadvantages of each. Also describes algorithms for summarizing graph patterns. Finally, we draw conclusion in Section III.

II. FREQUENT SUBGRAPH MINING ALGORITHMS

There are several efficient frequent sub graph mining algorithms have been proposed. Efficient frequent sub graph mining algorithm is an algorithm which gives small number of graphs as a result from large graph database. There is an algorithm which gives approximate graph mining based on Spanning tree [15]. The frequent sub graph mining algorithms comes under two different types :

A. *Algorithms using BFS Search Strategy*

These types of algorithms come under Apriori based approach. In these approaches before mining any of the sub graphs of size k+1, mining of sub graphs with size k needs to be completed, where the size of the graph is defined by the number of its vertices. In this method, since the candidates are generated in level-wise manner, the BFS search strategy must be used. To generate the biggest candidate for frequent sub graphs, frequent sub graphs of size k are merged together to generate a graph of size k+1.
Major algorithms with this approach are:

1. *AGM Algorithm:* AGM [2] is the Apriori based graph mining algorithm. This algorithm uses adjacency matrix for graph representation. Searching technique used in this algorithm is BFS Strategy. BFS mines all isomorphic subgraphs, whereas DFS does not do so and therefore, DFS consumes less memory. Largest graph of chemical compound discoved by AGM algorithm have the size of 13 atoms. AGM algorithm can efficiently mine frequently induced subgraphs in given graph dataset. Though it mines frequent subgraphs, still it generates multiple candidates due to use of BFS strategy.

2. *FSG Algorithm:* FSG[3] is also an Apriori based graph mining algorithm. In this algorithm, edges in graph considered as frequent items in traditional itemset. Hence

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 3, March 2013

size of graph can be increases only by adding single edge to the subgraph. Every time candidate subgraphs are generated by adding edges to previous subgraph. Hence candidate subgraph generated at current stage must be increase in size than the last stage subgraph. FSG uses sparse graph representation to store input graph transactions, intermediate candidate graphs and frequent subgraphs. Adjecency list representation is used to store each transaction graph. To check unique subgraph, FSG uses canonical labeling technique in which each graph have unique canonical label. Canonical labels of two graphs are same only when both graphs have same topological structure and same edges and vertices labels. This technique is used to check isomorphic graphs.

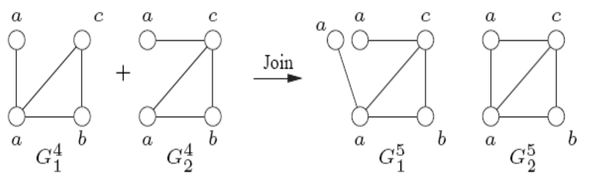

Fig 2: Multiple Candidate Generation

FSG also have some drawbacks. It also generates multiple candidates. FSG uses isomorphism testing technique, is very costly.

B. *Algorithms Using DFS Search Strategy*

These types of algorithms come under Pattern growth approach. Multiple candidate can be generated which is very costly in BFS strategy Hence in this method, for candidate generation DFS search strategy must be used. Major algorithms come under this approach are:

1. *Gspan Algorithm:* Gspan algorithm [4] is based on pattern growth approach. Gspan states for graph Substructure pattern mining. Due to use of DFS strategy and rightmost extension, multiple candidate generation can be reduced in Gspan. It works on labeled simple graphs. Each graph has unique label for each edge and each vertex. It finds frequent subgraphs.

Definition 1 : Frequent Subgraph : If subgraph g has support either greater or equal to the minimum support, then g is frequent subgraph in database D[7]. Instead of searching graphs and testing for isomorphism gspan construct canonical DFS codes. Each graph has different canonical labeling. And if canonical labels of two graphs are same then these two graphs are isomorphic with each other. DFS code is linear order of edges. Mining frequent subgraphs in Gspan is to finding subgraph having minimum DFS code. The Gspan algorithm is easily expanded in other domain like sequences, trees etc.

Gspan is having so many advantages; still it is inefficient with large size graph dataset.

2. *CloseGraph Algorithm:* Instead of mining all frequent subgraphs, CloseGraph algorithm [1] mines all closed frequent subgraphs.

3.

Definition 2: Close Frequent subgraph: A subgraph g is known as closed frequent subgraph in dataset D, if there exist no proper supergraph of g that has same support.

4. *RP-FP Algorithm:* All algorithms described above are used to mine the frequent sub graph from large graph dataset. But the number of frequent graph patterns generated by these graph mining algorithms may be too large to be effectively explored by users, especially when the support threshold is low. Hence to summarize frequent graph patterns by a much smaller number of representative graph patterns, different algorithms are proposed by Jianzhong Li, Yong Liu, and Hong Gao named RP-FP, RP-GD, and RP-Leap [7]. RP-FP algorithm uses close Graph algorithm to mine closed frequent sub graph after getting closed frequent sub graph, using substantial sub graph isomorphism testing, it finds the jump patterns. This algorithm says that jump patterns are representative pattern. Hence all jump patterns are marked as representative patterns. Representative patterns represent all the graphs which are covered by jump pattern. After that RP-FP finds all the graphs which are not covered by jump pattern and compute representative patterns for the same.

5.

RP-FP algorithm [7] uses close Graph algorithm to mine closed frequent sub graph after getting closed frequent sub graph, using substantial sub graph isomorphism testing, it finds the jump patterns. This algorithm says that jump patterns are representative pattern. Hence all jump patterns are marked as representative patterns. Representative patterns represent all the graphs which are covered by jump pattern. After that RP-FP finds all the graphs which are not covered by jump pattern and compute representative patterns for the same.

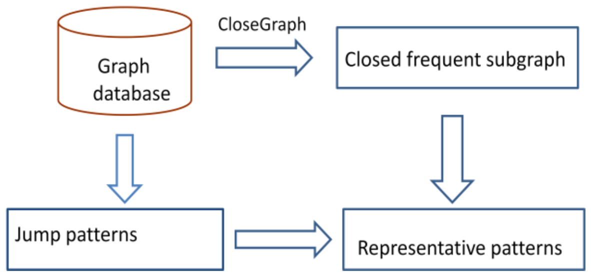

Fig : 3 Architecture of RP-FP Algorithm

Drawback of RP-FP is that to find representative patterns, it required to scan the closed frequent sub graph twice. Hence it is not efficient in case of large closed frequent sub graph. Because of this authors Jianzhong Li, Yong Liu, and Hong

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 3, March 2013

Gao implements RP-GD algorithm to mine small set of representative patterns efficiently.

6. *RP-GD algorithm:* RP-GD mines a representative set directly from graph databases [7]. Jianzhong Li, Yong Liu, and Hong Gao adopt the idea of online algorithm to devise RP-GD. Whenever some frequent sub graph mining algorithm generates a frequent sub graph, RP-GD attempt to discover representative R from the current representative set RS such that R can cover P where p in any closed frequent subgraph and R is one of the representative pattern from set of representative patterns RS. When there exists no representative in RS that can cover P, build a new representative Rnew that can cover P using some greedy strategies, and put Rnew that is newly discovered representative into RS. Thus, RP-GD can derive a representative set by scanning closed frequent sub graphs once.

If a graph pattern R can represent another graph pattern P, then R and P must have a large degree of similarity. To measure the similarity between two graph patterns, Jaccard distance formula is used.

Definition 3 (Jaccard distance) Let P1 and P2 be two graph patterns. The distance of P1 and P2 is defined in [7] as:

$$D\,(P1, P2) = 1- \frac{|T(p1) \cap T(P2)|}{|T(p1) \cup T(P2)|}$$

Where T (P1) represents the set of graphs in a database which contain the graph pattern P1. |T (Pi)| is the support of Pi.

Definition 4 (jump value) The jump value of a graph pattern is the minimum distance from its proper super graphs pattern in a set of graph patterns.

Definition 5 (δ - jump pattern) Suppose P is a graph pattern in a set of graph patterns S. If JV(P) > δ **,** then P is called a δ-jump pattern in S.

RP-GD algorithms Checks all graph patterns sequentially which are comes as output of CloseGraph algorithm for δ–jump patterns.

1. If graph p in δ–jump pattern then itself is representative pattern and place it into RS.
2. If an output graph pattern P is neither a δ-jump pattern, nor one can find a representative graph pattern covering P in the set of representative graph patterns generated previously, algorithm call P a greedy graph pattern since there is required to construct a new representative graph pattern which can cover P based on some greedy heuristic strategy.

Algorithm uses one of the method to solve above problem:

1. Building a New Representative
2. Searching Existing Representatives.

In case of building new representative and searching existing representative, RP-GD algorithm uses following three heuristic strategies to get less number of representative patterns.

a) Last Succeed First,
b) Reverse-Path-Trace strategy,
c) Nephew-Representative-Based-Cover strategy.
d)

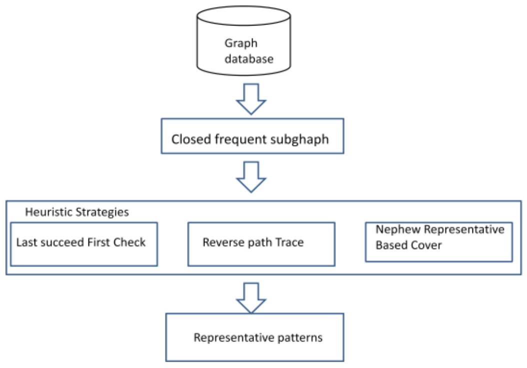

Fig : 4 Architecture of RP-GD Algorithm

Although RP-GD substantially outperforms RP-FP, it still depends on the Close Graph algorithm. In order to cover all closed frequent sub graphs, it is inevitable for RP-GD to check each closed frequent sub graphs sequentially.

7. *RP-Leap:* Jianzhong Li, Yong Liu, and Hong Gao develop more efficient algorithm called RP-Leap. RP-Leap can achieve substantial performance improvement over RP-GD by skipping those branches which contain few representative patterns in the pattern search tree. The framework of RP-Leap is similar to that of RP-GD. However, it performs a major improvement using leap search technique thai is before searching a branch rooted at the current graph pattern s, RP-leap first checks whether this branch can be skipped. Following figure shows architecture of RP-Leap algorithm

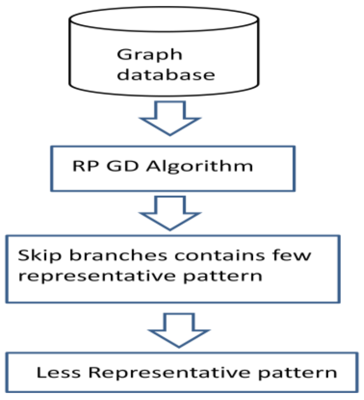
Graph database
RP GD Algorithm
Skip branches contains few representative pattern
Less Representative pattern

Fig : 5 Architecture of RP-Leap Algorithm

III. CONCLUSION

In this paper we have studied graph mining and different algorithms for finding frequent subgraph. From this article it is clear that Algorithms used Pattern growth approach are more efficient in case of time complexity than algorithms used apriory based approach. RP-GD algorithm makes more efficient due to use of three heuristic strategies named Last-Succeed-First-Check , Reverse-Path-Trace strategy, Nephew-Representative-Based-Cover strategy to get less number of representative patterns. RP Leap algorithm is more efficient in case of time required. But RP-Leap cannot give complete coverage of all graphs of dataset as like RP-GD Algorithm, because it skip some of the branches which are have few representatives. Hence as compared to RP-Leap, RP-GD algorithm gives complete coverage of all graphs of dataset.

REFERENCES

[1] X. Yan and J. Han, "CloseGraph: Mining Closed Frequent Graph Patterns," Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), pp. 286-295, 2003.
[2] A. Inokuchi, T. Washio, and H. Motoda, "An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data," Proc. Fourth European Conf. Principles of Data Mining and Knowledge Discovery (PKDD), pp. 13-23, 2000.
[3] M. Kuramochi and G. Karypis, "Frequent Subgraph Discovery," Proc. IEEE Int'l Conf. Data Mining (ICDM), pp. 313-320, 2001.
[4] X. Yan and J. Han, "gSpan: Graph-Based Substructure Pattern Mining," Proc. IEEE Int'l Conf. Data Mining (ICDM), pp. 721-724, 2002.
[5] J. Huan, W. Wang, and J. Prins, "Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism," Proc. Third IEEE Int'l Conf. Data Mining (ICDM), pp. 549-552, 2003.
[6] Liu, J. li, and G. Hong, "Efficient Algorithms for Summarizing Graph Patterns," technical report, Dept. Computer Science, Harbin Inst. of Technology,http://db.cs.hit.edu.cn/reports/2009/DBTR_Summariizng GraphPatterns.pdf,2009
[7] H. He and A.K. Singh, "GraphRank: Statistical Modeling and Mining of Significant Subgraphs in the Feature Space," Proc. Sixth Int'l Conf. Data Mining (ICDM), pp. 885-890, 2006.
[8] J. Huan, W. Wang, J. Prins, and J. Yang, "SPIN: Mining Maximal Fre-quent Subgraphs from Graph Databases," Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), pp. 581- 586, 2004.
[9] X. Yan, H. Cheng, J. Han, and D. Xin, "Summarizing Itemset Patterns: A Profile-Based Approach," Proc. 11th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), pp. 314-323, 2005.
[10] Jianzhong Li, Yong Liu, and Hong Gao "Efficient Algorithms for Summarizing Graph Patterns" Proc. Ieee Transactions on Knowledge and Data Engineering, Vol. 23, NO. 9, September 2011.
[11] http://dtp.nci.nih.gov/docs/3d_database/structural_information/structural_data.
[12] Chuntao Jiang, Frans Coenen and Michele Zito" A Survey of Frequent Sub graph Mining Algorithms".
[13] S. Zhang and J. Yang, "RAM: Randomized Approximate Graph Mining," Proc. Scientific and Statistical Database Management (SSDBM), pp. 187-203, 2008.
[14] X. Yan, H. Cheng, J. Han, and P.S. Yu, "Mining Significant Graph Patterns by Leap Search," Proc. ACM SIGMOD Int'l Conf.
[15] S. Zhang, J. Yang, and V. Cheedella, "Monkey: Approximate Graph Mining Based on Spanning Trees," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE), pp. 1247-1249, 2007. Management of Data, pp. 433-444, 2008.

BIOGRAPHY

Sujata J. Suryawanshi, ME pursuing, K.K.Wagh COE Nashik

Prof. Mrs. S.M.Kamalapur, ME, K.K.Wagh COE Nashik