Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 47 (2015) 197 – 204

Procedia Computer Science

Frequent Subgraph Mining Algorithms – A Survey

T.Ramraj[a], R.Prabhakar[b]

[a]*Assistant Professor, Department of Computer Science and Engineering, Coimbatore Institute of Technology, Coimbatore - 641014, India*
[b]*Emeritus Professor, Department of Computer Science and Engineering, Coimbatore Institute of Technology, Coimbatore - 641014, India*

Abstract

Graphs are common data structures used to represent / model real-world systems. Graph Mining is one of the arms of Data mining in which voluminous complex data are represented in the form of graphs and mining is done to infer knowledge from them. Frequent sub graph mining is a sub section of graph mining domain which is extensively used for graph classification, building indices and graph clustering purposes. The frequent sub graph mining is addressed from various perspectives and viewed in different directions based upon the domain expectations. In this paper, a survey is done on the approaches in targeting frequent sub graphs and various scalable techniques to find them.

Peer-review under responsibility of organizing committee of the Graph Algorithms, High Performance Implementations and Applications (ICGHIA2014)

Keywords: Candidate Graphs, Support, Apriori-based, Pattern-growth.

1. Introduction

In recent years, many authors have deployed many algorithms and tools for converting voluminous data into useful and meaningful information [1].Frequent graph mining is one of the arms of such techniques when the data are represented in the form of graphs [2].Identification of frequent graphs / sub graphs in a graph database or in a single large graph is a part of frequent graph mining which can be used for classification tasks [3], graph clustering and building indices. Frequent sub graph discovery is a process of identifying frequently occurring sub graphs from a set of graphs (graph database) or a single large graph with frequency of occurrence is no less than a specified threshold. Since subgraph discovery is

*T.Ramraj Email:ramraj1521@gmail.com

Peer-review under responsibility of organizing committee of the Graph Algorithms, High Performance Implementations and Applications (ICGHIA2014)
doi:10.1016/j.procs.2015.03.198

a portion of FSM (frequent subgraph mining), the term 'FSM' is used in the rest of this paper. The frequent sub graph mining is addressed from various perspectives based upon the requirement and domain expectations. Also it is viewed from various directions using various approaches. For example, Borgelt and Berthold [4] studied HIV-screening dataset and found active chemical structures in it by contrasting the support of frequent graphs between various different classes. Deshpande et.al [5] classified chemical structures by considering frequent patterns as a cardinal feature. Huan et.al [6] studied protein structure families by applying frequent graph mining techniques. To perform graph search in a faster manner, Yan et.al [7] used frequent graph patterns as indexing features.

Also in most of the chemical applications, the end-user is not only interested in finding frequent graphs (which reveal about the prediction of biochemical activities) but in identifying significant patterns (which may serve as catalysts for some biochemical activities) which seldom occur. So with regard to graph classification, still investigations are carried out to identify *which substructure (frequent sub graphs / significant sub graphs) has the cardinal influence (biggest contribution) to the classification*. In this paper, a survey is limited on enunciating few existing scalable techniques to find frequent sub graphs in a graph database / in a single large graph.

2. Strategy for finding Frequent Subgraphs

As stated earlier, frequent sub graphs are more useful in classification tasks, graph clustering and characterization of graph sets. As the sub graph size decreases drastically, the graph pattern size grows exponentially. This may tend to drag some serious issues like

i. Identification of frequent sub graphs may take more time.

ii. More sub graphs information may possibly hinder the task of identifying graphs which are more interesting but not frequent and which are not but still frequent.

To effectively solve the first task, scalable algorithms are recommended which generate frequent sub graphs in a search space [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 and 21]. The second task can be solved by preciously identifying significant graphs / patterns in a given graph database or in a single large graph and how frequently they occur [22, 24, 25, 26]. It is also to be noted that, finding frequently occurring sub graphs in a graph directly implies to the problem of enumerating sub graphs in a given graph which is a NP-Hard Problem [8].

3. Approaches in targeting Frequent subgraph discovery problem

The approaches for identifying FSM generate candidate sub graphs which are used to count how many instances are present in the given graph database.

3.1 Candidate Generation

It is one of the phases in frequent subgraph mining in which candidate subgraphs are systematically generated. Some of the strategies which identify candidate subgraphs are listed below:

1. *Level- wise Join:*
   Two subgraphs of size 'k' are combined together to form a (k+1) candidate subgraph.
2. *Rightmost path extension:*
   In this candidate generation strategy, vertices are added on the rightmost path of a k-subtree to form (k+1) subtree.

And other strategies such as Extension and Join, Right-and-Left tree Join and Equivalence class based extension are also used in candidate generation phase.

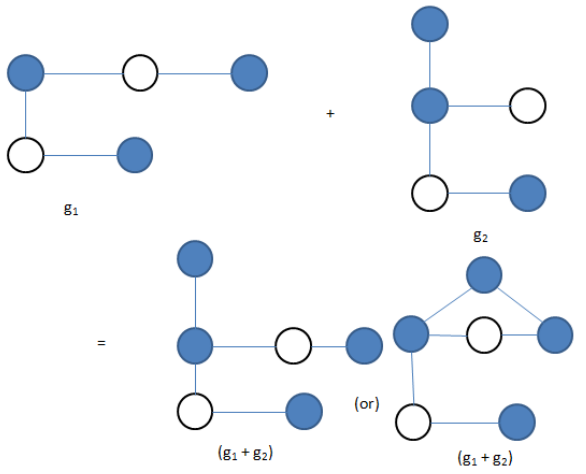

Figure 1: Two subgraphs of size 'k' are merged together to form candidate subgraphs of size 'k+1'

.

There are two approaches by which frequent sub graphs can be found in a given graph / database.

- Apriori – based approach
- Pattern-growth approach

The quint essential behind apriori –based sub graph discovery problem is to join two frequently occurring subgraphs namely G_1 and G_2 and check whether the resultant graph (whose size is one vertex more than the two sub graphs $G_1$and G_2) is frequent or not(Figure 1).

3.2 Apriori – Based Algorithms

The apriori – based approach is similar to frequent item-set mining and it is Recursive [27].Some of the apriori -based frequent sub graph mioning algorithms are listed below.

- AGM[11]
- FSG[12]
- Edge disjoint path-join algorithm[13]

AGM

This algorithm generates candidate graphs, merges any two candidate graphs at an instant and check whether the resultant graph is a sub graph in a given graph / graph database or not.Here, the size of a graph is denoted by the number of vertices present in that graph. Two graph of size 'k' can be merged together to form a resultant graph of size 'k+1'.One or more resultant graphs of size 'k+1' is again fed into the apriori algorithm to obtain a resultant graph of size 'k+2'. So, in each and every iteration of this algorithm, two graphs (arbitrarily chosen from the candidate set) are merged together to form a resultant graph whose size is increased by one vertex. In short, it is a *vertex-based candidate generation* algorithm.

//Apriori Algorithm

Function apriori(S^k, D, min_sup)
//D – Graph Database
//S^k – frequent subgraphs of size 'k'
// min_sup – minimum support value
// L_{k+1} - list of graphs whose size is 'k+1'
[
S^{k+1} ← NULL;
For all $g_i \in S^k$
[
for all $g_j \in S^k$

```
                [
                        L_{k+1} ← form as many as graphs possible by combining g_i and g_j : the size of the
                        resultant graph is 'k+1'
                ]
        ]
        For all g_u ∈ L_{k+1}
        [
                If [ (g_u is frequent in D) AND ( g_u ∉ S^{k+1}) ]
                Then
                        S^{k+1} ← S^{k+1} ∪ g_u;
                End if
        ]
        If [ S^{k+1} <> NULL ]
        Then
                apriori( S^{k+1}, D, min_sup );
        End if
]// end function
```

FSG

It adopts with *edge-based candidate generation* method. Here, the size of a graph denotes the number of edges present that graph. Similar to vertex based candidate generation method, two graphs of size 'k' are merged together to form resultant graphs of size 'k+1' which should also be frequent. So in and every iteration, it generates candidate sub graphs whose size is exactly 1 greater than the previous frequent ones. Candidate pruning is also done if the generated candidate does not satisfy the minimum threshold.

```
// FSG Algorithm

Function FSG ( D, min_sup )
// D – Graph database
// min_sup – minimum support threshold
[
        F^1 ←detect all frequent 1-subgraphs in D
        F^2 ←detect all frequent 2-subgraphs in D
        K ← 3
        While [ f^{k+1} ≠ NULL ]
        do
                C^k ← fsg_gen( f^{k-1});
                For each candidate g^k ∈ C^k
                do
                        g^k.count ← 0;
                        for each graph g ∈ D
                        do
                                if [ candidate g^k ∈ g ]
                                then
                                        g^k.count ← g^k.count + 1;
                                end if
                        done
                done
                F^k ← { g^k ∈ C^k : g^k.count ≥ min_sup };
                K ← k + 1;
        Done
        Return F^1,F^2,F^3,…,F^{k-2};
]// end function
```

Edge – disjoint path join algorithm

This algorithm abides with apriori – based approach which uses edge-disjoint paths as building blocks. Here, the measuring factor is the number of disjoint paths (Two paths are said to be disjoint if they do not share a common edge) a graph possesses. Two candidate graphs of 'k' disjoint paths are merged together to form a resultant graph containing 'k+1' disjoint paths.

Comparative study between AGM and FSG

In AGM, the candidate generation of the frequent induced sub graph is constructed by the level-wise search in terms of the size of the sub graph. The adjacency matrix of the graph representation is combined with this efficient level-wise search of the frequent canonical matrix code [28].Overall analysis showed that the computational time complexity of directed graphs is less than that of undirected graphs due to the fact that the possible edge directions in directed graphs bear more sub graph patterns and their frequency will ultimately be less. Also the computational complexity of small graphs is lesser than larger graphs. Experiments reported in [16] showed that AGM achieves good performance in dense synthetic graph datasets and took 40 minutes to 8 days (approx.) to tabulate all frequent sub graphs in a dataset containing 300 chemical compounds, as the minimum support threshold varies between 10% to 20%.

In FSG, frequent 1-edge and 2-egde graphs are enumerated and from these two basic sets candidate graphs whose size greater than the previous ones are generated. It only finds subgraphs that are connected. This is beneficial because it is necessary to consider disconnected combinations of frequent sub graphs. It uses canonical labelling (A canonical code is a unique code of a given graph and it is always be the same no matter how the graphs are represented, as long as those graphs have the same topological structure and the same labelling of edges and vertices) to compare two sub graphs. Computing canonical labels is equivalent to determining isomorphism between graphs. If two graphs have the same canonical labelling then they are isomorphic with each other and further it is under investigation *whether they belong to class P or NP-Complete*[29].

3.3 Pattern Growth Approach

A pattern-growth approach extends a frequent sub graph by adding an extra edge in every possible position. The overhead in joining two sub graphs of size 'k'(where 'k' is large) to form a graph of size 'k+1' is avoided in this approach. But the significant drawback here is while adding an extra edge in every possible position, the same sub graph can be discovered many times leading to duplication in candidate generation phase. This can be considerably eliminated by using rightmost extension technique. Pattern-Growth approach algorithm include SPIN[17], Mofa[4], gSpan[35], FFSM[16], and Gaston[19].

MoFa: (Molecular Fragments Identification Technique)

It finds core structures (sub graphs) which are found in all given molecular structures and generates an embedding list. In the next level, each and every structure present in the embedding list is extended by adding an edge in all possible ways which generates different structures. It uses pruning technique to suppress support computation and performing extension operation only to the embedding list structures.

Here multiple reporting of the same substructure (redundancy) arises and it can be suppressed by maintaining a list of frequent substructures and withdrawing new ones that are identical to already known/existing ones.

SPIN: (Spanning Tree based Maximal Graph Mining)

The crux of this algorithm is to mine sub graphs that are not a part of any other frequent sub graph. It uses spanning tree approach to discover maximal frequent sub graphs.First this algorithm generates maximal frequent tree pattern from a Graph database and enumerates the equivalence class of a tree. Then it constructs maximal frequent sub graph from the constructed trees.

gSpan

This algorithm generates a tree-like structure (DFS Code tree) over all possible patterns (subgraphs), in which each and every node represents a DFS Code[35] for a graph pattern. The i^{th} level of the code tree contains DFS Code of all subgraphs of size (i-1). Each (i-1) subgraph is generated by adding one extra edge to subgraphs which are present in the $(i-2)^{th}$ level of the tree.Instead of embedding the entire subgraph in

each and every node it only preserves the transaction list for each discovered subgraph. Also pruning is done by deleting nodes which do not satisfy *minimal DFS Code*[35].

All the above algorithms are based upon Prefix Span[33],TreeMinerV[15] and FREQT[32].

Most of the pattern-growth approaches use edge-extension strategy and a potential problem with edge -extension is that the same subgraph can be discovered many times. Among these algorithms, gSPAN avoids the duplicated graph discovery by right most extension technique, in which the extension is possible only on the right-most path[34]. Any depth first path from a source V_o to an arbitrary sink V_n is declared as the right most path from V_o to V_n.

Apart from frequent sub graph mining algorithms, constraint based subgraph mining have also been proposed. Mining coherent subgraph were studied by Huan et al[6].To discover exact dense frequent sub graphs, Yan et al[31] proposed two algorithms called as Splat and Closecut. Wang et al[23] developed a frequent graph mining algorithm(disk-based) for a voluminous graph database. To mine closed and maximal frequent subtrees, Chi et.al[26] proposed a new algorithm called as CMTreeMiner. There are also some studies to find frequent subgraphs in a large graph. While defining the support of a graph in a large graph, there exists two or more same sub graphs which are frequent and are overlapped (multiple embedding of a subgraph in a large graph may overlap). If *multiple non- identical embeddings* of subgraphs are allowed, then there is a possibility of violation of anti-monocity property (if the k-size subgraph is frequent only if all of its subgrpahs are frequent) which is a cardinal feature for the most frequent mining algorithm. Therefore in order to find an appropriate support definition, Kuramochi and Karypsis[12] coined definitions for overlaps and framed two algorithms namely HSIGRAM (horizontal approach abiding Breadth-First-Strategy) and VSIGRAM (vertical approach abiding depth first strategy) to discover all frequent subgraphs.

3.4 Significant Subgraphs

While deploying graph applications based on frequent graph discovery, first all the frequent sub graphs are mined and then significant patterns are identified and selected based upon objective functions (user-defined)for different graph applications. Inorder to identify frequently occurring sub graphs it is necessary to populate all sub graphs and calculate their *p-value* one after the other. This is a very time consuming process because a low frequency threshold means an exponential pattern set and the mining process is slow. Also most of the frequent patterns are redundant and not worth computing at all. So there are some recent studies to mine significant sub graphs based upon objective functions (user-defined) to overcome the scalability issues. For example, Kudo et al[20] proposed a *branch-and-bound approach, Gboost*, for graph classification and Saigo et al[25] proposed an intuitive mining algorithm based on PLS(partial least square regression) to mine significant pattern. *GrapSig* algorithm is proposed by Ramu & Singh[24] to mine Significant and Frequent subgraph in which graphs are represented as feature vectors. And a structural Leap Search Mining Strategy is proposed by Yan et al.[14] to identify significant Subgraphs.

3.5 Other Applications

Apart from Graph classification, Clustering and indexing FSM has many applications in inter-disciplinary research such as chemical informatics [4], bioinformatics [30]. For example, Maximal subgraph mining is used in discovering structure motifs in a graph of homology protein where they encode the maximal structure commonalities within the group. It also has a cardinal influence in social networking and ego networks as well.

Below tabular column shows a list of FSM algorithms in a 2D matrix form in which i^{th} row and j^{th} column has a value x.

Where (i^{th} row) i – denotes the name of the Algorithm
(j^{th} row) j – candidate generation method
x – candidate graph representation

Approach	Algorithm	Candidate Generation			
		Level-Wise Join	Right-most path Extn.	Extension and Join	Paths, trees and Graphs - Enumeration
A Priori	FSG	Adjacency List	-	-	-
	AGM	CAM	-	-	-
	HSIGRAM	CAM	-	-	-
	FFSM	-	-	CAM	-
Pattern - growth	gSpan	-	Adjacency List	-	-
	Gaston	-	-	-	Hash Table
	MoFa	-	-	Embedding List	-
	SUBDUE	CAM	-	-	-
	CloseGraph	-	Adjacency List	-	-

CAM – Canonical Adjacency Matrix

4. Conclusion

In this paper, few frequent subgraph mining algorithms are discussed. In general, mining frequent graph patterns takes a long time so several methods to explore significant subgraphs without generating the entire pattern set are also considered. These methods usually reduce the computational complexity but still more investigations are being carried out to identify *what kind of subgraphs are most compact and representative* for a given application. Since the generated candidate set is also too large and all the frequent patterns are not useful, new emerging techniques deploy *approximate algorithms* to find frequent sub graphs.

REFERENCES

[1] C.Jiang,F.Coenen, M.Zito,"A Survey of frequent sub graph mining algorithms", Knowledge Engineering Review, 2012.
[2]A.nanopoulus,Y.Manolopoulos,"Mining patterns from graph traversals", Data and Knowledge Engineering 37 (2001) 243-266.
[3]N.Acosta-Mendoza, A.Gago-Alonso,et.al.,"Frequent approximate sub graphs as features for graph-based image classification", Knowledge Based Systems,27 (2012), 381-392.
[4]C. Borgelt and M. R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *Proc. 2002 Int. Conf. Data Mining (ICDM'02)*, pages 211–218, 2002.
[5]M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis,"Frequent substructure-based approaches for classifying chemical compounds", *IEEE Trans. on Knowledge and Data Engineering*, 17:1036–1050, 2005.
[6]J. Huan,W.Wang, D. Bandyopadhyay, J. Snoeyink, J. Prins, and A.Tropsha, "Mining spatial motifs from protein structure graphs", In *Proc. 8th Int. Conf. Research in Computational Molecular Biology (RECOMB)*, pages 308–315, 2004.
[7]X. Yan, P. S. Yu, and J. Han, "Graph indexing: A frequent structure-based approach", In *Proc. 2004 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'04)*, pages 335–346, 2004.
[8] T.Gartner, P.Flachand S.Wrobel, "On Graph Kernels : Hardness results and efficient alternatives", In Proc. of the 16th Annual Conference on Computational Learning Theory, 2003.
[9] Lawrence B. Holder , Diane J. Cook , Surnjani Djoko, "Substructure discovery in the subdue system" , In Proc. of the AAAI Workshop on Knowledge Discovery in Databases, 1994.
[10] Luc Dehaspe , Hannu Toivonen , Ross Donald King , "Finding frequent substructures in Chemical compounds", 1998.
[11] Akihiro Inokuchi, Takashi Washio and Hiroshi Motoda, "An apriori-based algorithm for mining frequent substructures from graph

data", Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery.
[12] M.Kuramochi and G.Karypis, " Frequent subgraph discovery", In Proc of ICDM, 2001.
[13] N.Vanetik et.al.,"Computing frequent graph patterns from semi structured data", In Proceedings 2002 IEEE International Conference on Data Mining. ICDM-2002.
[14] Xifeng Yan, Hong Cheng, Jiawei Han, Philip S. Yu, "Mining significant graph patterns by scalable leap search", Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008
[15] M.J.Zaki," Efficiently mining frequent trees in a forest", In Proc 2002, ACM SIGKDD Int Conf. Knowledge Discovery and Datamining (KDD'02), 2002.
[16] J.Huan, W.Wang and J.Prins, "Efficient mining of frequent subgraph in the presence of isomorphism", In Proc. 2003 Int Conf. DataMining (ICDM'03).
[17] J.Huan, W.Wang and J.Prins, "SPIN: Mining maximal frequent subgraphs from graph databases, In Proc. 2004. ACM SIGKDD Int Conf. Knowledge Discovery and Datamining (KDD'04), 2004.
[18] M.Kuramochi and G.Karypis, "Finding frequent patterns in a large sparse graph", Journal Data Mining and Knowledge Discovery, Volume 11 Issue 3, November 2005 .
[19] S.Nijssen and J.Kok, "A quickstart in frequent structure mining can make a difference", In Proc. 2004. ACM SIGKDD Int Conf. Knowledge Discovery and Datamining (KDD'04), 2004.
[20] T.Kudo et al, "An application of boosting to graph classification", In Advances in Neural Information Processing Systems, 2004.
[21] Bjorn Bringmann and Siegfried Nijssen, "What is frequent in a single graph?".2007
[22] H.He and A.K.Singh, "Efficient algorithms for mining significant sub structures in graphs with quality guarantees",ICDM'07
[23] C.Wang et al, " Scalable mining of large disk base graph databases", In Proc 2004, ACM SIGKDD Int Conf. Knowledge Discovery and Datamining (KDD'05), 2005.
[24] S.Ranu and A.K.Singh, "Graph Sig: A Scalable approach to mining Significant subgraphs in large graph databases", In Proc 2009, ICDE 2009.
[25] H.Saigo et al, " Partial least squares regression for graph mining", In Proc 2008, ACM SIGKDD Int Conf. Knowledge Discovery and Datamining (KDD'08), 2008.
[26] Y.Chi et al, " Mining closed and maximal frequent subtrees from databases of labeled rooted trees", IEEE TKDE,2005.
[27] Charu.C.Agarwal,"Managing and mining graph data"
[28] Inokuchi.A, Washio.T and Motoda, "Derivation of the topology structure from massive graphs data", Discovery Science Proc. of 2nd Intl. Conference , pp 330-332.
[29] R.C.Read and D.G.Corneil, "The graph isomorph disease", Journal of Graph Theory 1: 339-363, 1997.
[30] J.Huan & W.Wang ,"Accurately clasify protein family etal based on coherent subgraph mining ", in Pacific Symposium on Biocomputing – 2004.
[31] X.Yan et al, " Mining closed relational graphs with connectivity constraints", In Proc 2005, ACM SIGKDD Int Conf. Knowledge Discovery and Datamining (KDD'05), 2005.
[32] T.Asai, K.Abe et al, " Efficient substructure discovery from large semi structured data", In Proc 2002, SIAM. Int Conf. DataMining (SDM'02), 2002.
[33] J.Pei et al, " PrefixSpan: Mining Sequential patterns efficiently by prefix- projected pattern growth", In Proc.2001, Int Conf. Data Engineering (ICDE'01), 2001.
[34] Yan and Han, "Close Graph:Mining Closed frequent graph patterns", In Proc.2003 ACM SIGKDD Int Conf. Knowledge Discovery and Datamining (KDD'03), 2003.
[35] Yan and J.Han, "gSpan: Graph-based substructure pattern mining", In Proc.2002 Intl. Conf. on Data Mining (ICDM 02), 2002.