A Survey of Frequent Subgraph Mining Algorithms

$PratikMandlecha(20161172), SailendraDS(20161216)$

***Abstract*—Modeling data as a graph has proved an efficient approach considering huge amount of data and large number of relationships among them. The process of mining data sets, represented as graph structures, called graph mining is widely studied in bioinformatics, computer networks, chemical reactions, social networks, program flow structures, etc.Frequent Subgraph Mining is defined as finding all the subgraphs in a given graph that appear more number of times than a given value.Many graph mining algorithms have been proposed in recent past researchers; all this algorithms rely on a very different approach so its really hard to say that which one is the most efficient and optimal in the sense of performance. This paper discovers present FSM techniques and tries to give their comparative study.**

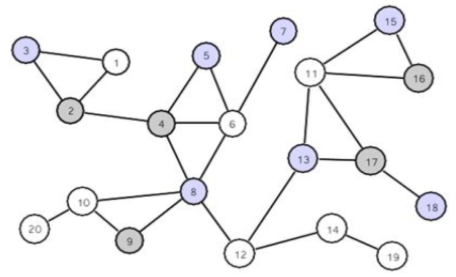
Keywords: Graph mining, Apriori based approach, Pattern growth approach, Graph, Frequent Subgraph Mining.

I. Introduction

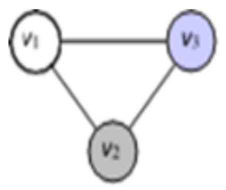
THE vast interconnections between real world objects can be best modeled as graph databases which try to mimic those relationships in consistent and intuitive way. As the size of data grow and data become less structured, graph databases become extremely helpful in understanding scenarios such as retails suggestion engines, protein-protein interaction networks, social network monitoring, fraud detection etc. Graph mining, one of the novel approaches for mining the structured data represented by graph, has gained much attention in the last few decades. Most studied graph mining techniques are as follows: (1) Graph Classification, a type of supervised learning, (2) Graph Clustering, a type of unsupervised learning and (3) Subgraph Mining, which we will discuss here. Frequent Subgraph Mining is defined as finding all the subgraphs in a given graph that appear more number of times than a given support threshold. It is a widely studied problem as it results in the recurrent structures, themes or ideas in the given graph database which can be used further for performing other graph mining applications such as graph classification, graph partitioning, graph clustering, graph correlations etc. There are two main categories in which algorithms for graph mining problem have been designed, as follows:

1)***Transactional graph setting*:** In which a set of graphs is the input to the algorithm and a frequent subgraph is the one that appears in a pre-specified number of those transactional graphs.

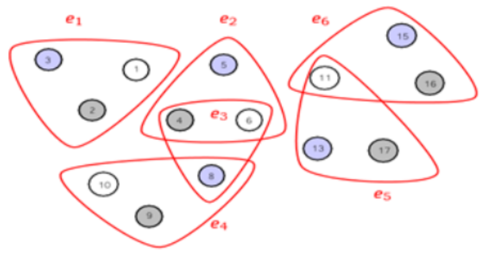
2) ***Single graph setting (mono-graph setting)*:** In which a single large graph is the input to the algorithm and a frequent subgraph is the one whose embeddings occur for a pre-specified number of times in the graph.



: (a) Sample Graph



(b) Pattern



(c) Possible Subgraphs

Fig.1. Pattern mining for finding 3-cycles in a Graph.

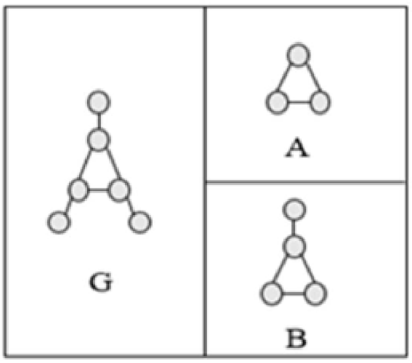
We will mainly study and compare existing works on Subgraph Mining. Thus we hereby present a comparative survey on Frequent Subgraph Mining (now on referred as FSM) and FSM Algorithms.The main performance parameters used for the evaluation of a FSM algorithms are:

- **Size of the largest subgraph found.**
- **Number of the graph isomorphism computation.**
- **Total number of candidate subgraphs generated**
- **Amount of time required by the algorithm. Here time is the total time spent on CPU or the total of the time spent on data mining and support computation.**

The rest of the paper is organized as follows.
Section-2 briefly explains the terminologies used in graph theory.
Section 3 presents the phases in FSM approach.
Section 4 is FSM Algorithms Survey.
Section 5 has Algorithmic Approach Based Classification discussing various algorithms in FSM.
Section 6 has the conclusion of the work.

II. Definitions

Definition 2.1: A graph is a set of vertices and edges representing different objects in data and relationships between objects respectively. The object definition may depend on the type of dataset being used.
Definition 2.2: Labeled Graph is a precise type of graph where every vertex and edge has a tag or label associated with it. Labels of Vertices is drawn from a set of labels named as VertexLabel(LV) similarly Labels of edges is drawn from a set of labels named as EdgeLabel(LE).
Definition 2.3: A graph is **connected** if one or more path exist between every vertex pair in the graph else graph is **disconnected** graph. Connected graph is always Uni-component graph i.e. has only one component.
Definition 2.4: Isomorphic graph : Two graphs Gx = (Vx, Ex) and Gy = (Vy, Ey) are isomorphic if they are topologically identical to each other, i.e., there is a mapping from Vx to Vy in such a way that each edge in Ex is mapped to a single edge in Ey and vice versa.
Definition 2.5: Downward Closure property states that for every graph G, A and B, where A is a subgraph of B and G is the input graph, the support of subgraph B in G should never be greater than support of subgraph A in G. Maintaining downward closure property is of crucial importance because it helps in pruning out the search space and without it exhaustive search is unavoidable.



Here in **Fig 2**, graph A is subset of graph B, which are subgraphs of the given graph G. The graph G contains the embeddings of subgraphs A and B. According to the nave definition support of subgraph A is 1 in G and support of subgraph B is 3 (but not independently) in G. Considering auto-morphism further doubles up the supports.This has violated the downward closure property due to overlapping of embeddings.

III. Overview of FSM

FSM process is defined as the process of finding all the frequent subgraphs from the input database that appear more than the given support threshold. With reference to literature related to FSM algorithms it was identified that this process mainly consists of three aspects.

3.1. Graph Representations

Graph representation aspect of FSM is a mechanism with which a graph can be represented. Simplest Graph representation mechanism is adjacency list or adjacency matrix. In adjacency Matrix representation, rows and columns of that matrix represent a node or vertex. A positive intersection of ith row and jth columns tells us that there is an edge present connecting Vi and Vj. In easy words value ¡Vi, Vj¿ gives number of edges present between Vi & V j, i.e. if its 0 then Vi & Vj arent connected else if the value is n then n links or edges are present between these pair of two vertices. Since Graphs can be represented in many diverse ways so for adjacency list its dreadful to detect isomorphism. This weakness can be overcome by canonical naming because it derives unique code for each graph and this strategy ensures that two identical graphs are labeled similarly thus it make sure that canonical labels of two or more graphs are identical iff those graphs are isomorphic.

3.2. Subgraph Enumeration

Enumeration is process of calculate/count something. Counting subgraphs one by one is subgraph enumeration. There are currently two categories of graph enumeration
1) Join Operation, &
2) Extension operation.

3.3. Frequency Counting

Graph counting is done in two ways Embedding Lists (EL), and Recomputed Embedding. Single node graph is stored in an EL as list of every label occurrence in database while other graphs are kept in EL. This list contains i) index of embedding tuple in EL of predecessor graph, ii) identifier of graph (subgraph) and node in the main graph. Subgraph frequency is determined by count of different graph in its Lists. Lists are quick responsive but when it comes to scalability EL are not a good choice. Here comes concept of Recomputed Embeddings i.e. RE which keeps an 'active graph's set' in which occurrences are computed repeatedly.

IV. FSM Algorithms: Survey

FSM problem is being studied since later 1990s, since then this problem has been addressed in many ways and directions with many approaches. Among those approaches most popular ones are Apriori-based and pattern growth based approaches. Also algorithms differ in many ways like the input graph type, search strategy used by them, frequency counting method, or graph representation method. Hence many algorithms exist using different approaches. Every algorithm has got some limitations and performs better in specific application scenario and test cases.

V. Basis for Classification

5.1. Algorithmic Approach Based Classification

FSM algorithms are popularly classified into two categories:

5.1.1 Apriori based approach

The basic idea behind Aprioribased FSM Algorithms is to join such two subgraphs which are frequently occurring e.g. G1 and

G2 and check for the resultant graph (whose size is exactly one vertex more than other two frequent subgraphs G1and G2) is frequent or not. Apriori based algorithms are recursive in nature because they need subgraph with k-1 size to determine subgraph with size k. Such algorithms many times result in generation of multiple candidates.

5.1.2. Pattern Based Growth Approach

To prevent the overhead caused by merging two subgraphs in Apriori approach based algorithms, Pattern Growth Algorithms were developed. The pattern Growth FSM algorithm extends frequent graph by adding new edge, in each possible position. Since the pattern-growth approaches uses edge-extension technique and a possible problem with this approach is that same subgraph can be discovered many times. This problem was later overcome by using rightmost extension only in which approach extensions take place at rightmost path only.

5.2. Search Strategy Based Classification

There are two very popular search strategies applied to find out FSGs and do not require any explanation:
A. BFS (Breadth First Search) or Level wise search strategy
B. DFS (Depth First Search) strategy

5.3. Nature of Input Based Classification

The algorithms are classified in two types based on the correctness of the input. One algorithm takes in exact graph sets as input, whereas the second type of algorithms take an uncertain set of graphs as input. Another possibility of classification is based on type of input graph. One type of algorithm takes input as a single large graph, whereas the second type of algorithms takes set of small graphs as input. The third possible correctness of the graph data where it is exact or uncertain.

5.4. Totality of Output Based Classification

Based on the set of the FSGs discovered, the algorithms are of two types. The first variety returns the complete set of FSGs, whereas another variety of solution returns a partial set of FSGs

VI. Existing FSM Algorithms

WARMR is probably first algorithm for FSM proposed in 1998 was an Apriori based solution uses ILP and level wise i.e. BFS and generates possible candidates based on subgraph size however this solution missed many FSGs. [1]

AGM is an Apriori approached solution developed in 2000, makes use of BFS strategy. So, this algorithm generates candidate graph based on vertex which increases substructure or subgraph size. To serve the requirement of graph representation, AGM uses adjacency matrix concept. Experiment was performed on large chemical compound dataset. AGM accurately mined FSG from input dataset however its complexity was high because of multiple candidate generation. [2]

AGM is an Apriori approached solution developed in 2000, makes use of BFS strategy. So, this algorithm generates candidate graph based on vertex which increases substructure or subgraph size. To serve the requirement of graph representation, AGM uses adjacency matrix concept. Experiment was performed on large chemical compound dataset. AGM accurately mined FSG from input dataset however its complexity was high because of multiple candidate generation. [2]

Farmer algorithm coined in 2001 is a Breadth-First Apriori algorithm which uses trie data structure for graph representation. This approach finds a subgraph and calculate greater possible subgraphs by adding an adjacent edge of child in every possible way. Farmer performs much better than WARMR, the previous mining algorithm. [3]

MoFa FSM Algorithm developed in 2002 is a Lex-DFS along with structural pruning based frequency counting Pattern Growth based solution which takes Set of Graphs as inputs and produces All Frequent Subgraphs (Now onwards referred as FSG) as output. It uses Adjacency list representation of Graph. This solution worked well on large chemical atomic structures. One restraint in this algorithm was that its generated frequent graphs arent exactly frequent. [4]

CloseGraph FSM Algorithm proposed in 2003 is a technical advancement of **gSpan** [5] algorithm & takes Set of Graphs as inputs and produces Frequent Graphs which are closed and connected in nature as output. Frequency counting is done in LexDFS way uses Adjacency list for graph representation. It uses concept of subgraph enumeration for candidate generation. Time overhead for failure detection is one of its notable drawback. [6] Kuramochi et al. [7] in 2004 tossed concept of heuristic FSM algorithm termed as **GREW** to escalate limitations of contemporary methods. GREW operates on bulky graph, finds patterns for every attached subgraph which has vertex-dis-joint embedding in enormous number. Their results proved GREW to be efficient and scalable as well. An extended version of this approach was further surfaced in 2006 as **Dynamic GREW** [8] which extends its working on dynamic graphs.

FSG is Apriori based FSM algorithm in which edges are considered as frequent item-set. Thus size of graph can be increased only by adding solo edge to the subgraph and new candidate graph will always be greater in size than preceding ones. FSG uses sparse representation of graph for storing candidate graphs and frequent subgraphs. Adjacency list representation is used for storing input graph. Canonical label technique is used to verify whether two graphs are isomorphic graphs or not. Isomorphic testing performed in FSG is very costly and multiple candidates are also generated during candidate generation process. [9]

Another solution floated same year with **JPMiner** by Yong Liu et al. in 2009 which gets set-of-Graphs as inputs and yields frequent jump patterns as output. Similar to mSpan in JPMiner also Frequency counting is based on Lex-DFS approach & Adjacency list are used for graph representation. Input dataset was a normal graph dataset. This algorithm sometimes generates much smaller set of jump pattern which result in increase of time complexity of algorithm.

Temporal Subgraph Patterns (TSP-algorithm) a novel approach by Hsieh et al. in 2010 uses Heterogeneous Information Networks (HIN) where input is dynamic Set-of-Graphs and Closed Temporal FSG are output. Frequency counting is done by Temporal Subgraph Patterns tree. It uses Adjacency list for graph representation. This algorithm takes extra overhead to check closed temporal patterns which result in increase of time complexity of algorithm.

RP-FP & RP-GD Both of these algorithms takes in Static Set-of-Graphs and gives Representative Graphs and Jump Pattern as output. Frequency counting is done in popular Lex-DFS order & Adjacency list is used for graph representation. This approach makes use of previously discussed CloseGraph algorithm for FSG mining. These algorithm takes overhead time to summarize the pattern
MultiObjectiveGBDM proved to be a better approach as it used MultiObjectiveGraphDataMining instead of conventional UniObject model and used unary as well as binary metrics for mining task, it implemented previous mining algorithms in multiobjective manner and result were better in each case.
DGP Density-based Graph Partitioning strategy was scalable approach able to mine FSGs of larger graphs with help of MapReduce framework, but in this approach there were few FSG miss compared to previous sequential algorithms but performance was improved considerably
Margin Algorithm presented an approach to find the maximal frequent subgraphs. The candidate set that is likely to be maximally frequent are the n-edge frequent subgraphs having a n+1-edge infrequent supergraph. The MARGIN algorithm computes such a set efficiently and finds the maximal frequent subgraphs by a post-processing step. Experimental results show that our algorithm performs up to three(twenty) times faster than SPIN(gSpan).

VII. Application Areas of FSM

FSM Algorithms are used in several areas like, Web analysis, Chem-Informatics, Wireless Networks, Telecommunication Networks, Financial Network Analysis, Social behavior Analysis, sentiment analysis, protein structure analysis etc. Figure 3 demonstrates various domains in which FSM algorithms are applicable and which Algorithm is suitable for that domain. Social behavioral analysis with FSM has big present and future scope.

VIII. Conclusion

In this Paper, we studied several FSM algorithms, their merits, demerits, behavior and application. FSM is not only part of Computer Science it has its application in real world problems and in many inter-disciplinary domains e.g. cheminformatics, Economics case studies, bioinformatics etc. Lot of work has been done and there is still possibility of ample amount of advancement in present FSM algorithms. There exist less number of research work which are able to work on dynamic or evolving graphs, similarly, parallel implementation of these algorithm will also save lot of our time. Also currently available algorithms just do better from their predecessor but none of them focus on the problem of NPcompleteness of FSM.

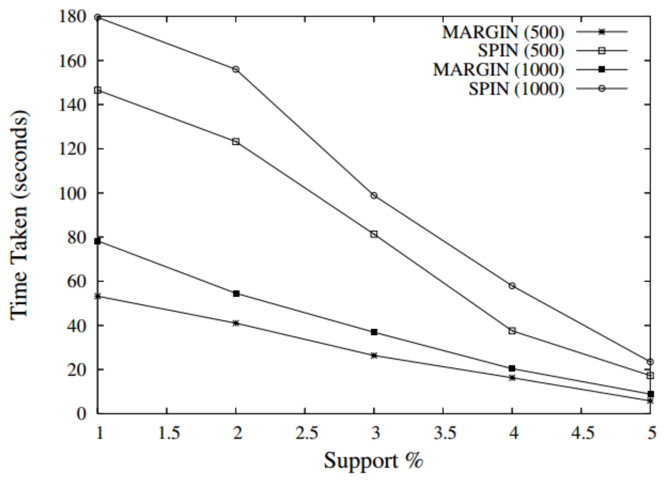


Fig.3. Margin vs Spin.
Table for Comparison of Algorithms on next page

TABLE I. FREQUENT SUBGRAPH MINING ALGORITHMS APPLICABLE ON SINGLE LARGE GRAPH WITH THEIR MERITS AND DEMERITS

Sr. no.	**Name of Algorithm**	**Type of input graph**	**Phase 1**	**Phase 2**	**Phase 3**	**Merits**	**Demerits**
			Candidate Generation technique used	***Support Computation technique used***	***Type of Result Generated***		
1.	SEuS('02)	Directed graph	Apriori	Using summary pointers	Approximate/ Complete	No need to access whole database for support computation	Creating summary becomes overhead at low support threshold.
2.	SiGram('05)	Undirected Sparse graph	Apriori	MIS	Approximate/ Complete	Frequent subgraph discovery methods are exact, approximate and upper bound	Memory exhausts and run-time exceeds boundary for dense graphs.
3.	Vanetik et.al.('06)	Directed and Undirected graph	Apriori	MIS	Complete	Less candidate subgraphs generated, hence less support computation required.	Time complexity is exponential for dense graphs.
4.	G-Miner('09)	Directed and Undirected graph	Pattern Growth	G-Measure	Complete	Takes global distributiveness as a measure of computation	For dense graphs algorithm run out of memory
5.	NODAR('12)	Undirected graph	Pattern Growth	SMNOES	Complete	Reduces subgraph isomorphisms	No major improvements for single graph setting
6.	GraMi('14)	Directed and Undirected graph	Pattern Growth	MNI	Approximate	2 orders of magnitude faster than other algorithms	--
7.	AGrap('14)	Undirected graph	Pattern Growth	MNI	Approximate	Approximately 5*(gApprox subgraphs) are generated at high similarity thresholds	Unable to handle large dense graphs (in order of thousands of vertices)