A COMPARATIVE STUDY OF FREQUENT SUBGRAPH MINING ALGORITHMS

K.Lakshmi[1] and Dr. T. Meyyappan[2]

[1.] Department of MCA, Sir M.Visvesvaraya Institute of Technology, Bangalore.
lakshmi_kes@rediffmail.com
[2.] Department of Computer Science and Engineering, Alagappa University,Karaikudi.
meyslotus@yahoo.com

ABSTRACT

Data mining algorithms are facing the challenge to deal with an increasing number of complex objects. Graph is a natural data structure used for modeling complex objects. Frequent subgraph mining is another active research topic in 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 databases is challenging since graph related operations, such as subgraph testing, generally have higher time complexity than the corresponding operations on itemsets, sequences, and trees. Many frequent subgraph Mining algorithms have been proposed. SPIN, SUBDUE, g_Span, FFSM, GREW are a few to mention. In this paper we present a detailed survey on frequent subgraph mining algorithms, which are used for knowledge discovery in complex objects and also propose a frame work for classification of these algorithms. The purpose is to help user to apply the techniques in a task specific manner in various application domains and to pave wave for further research.

KEYWORDS

Frequent subgraph mining, Isomorphism, Pattern growth, Apriori

1. INTRODUCTION

Knowledge discovery in complex objects involves understanding the relationship between their components. Examples are the Machine learning in domains such as bioinformatics, drug discovery, adverse drug events and web data mining. Graphs are natural data structures to model such relations, with nodes representing objects and edges the relationships between them. In this context, finding similarity between graphs is important. Simple ways of comparing graphs which are based on pair wise comparison of nodes or edges, are possible in quadratic time, yet may neglect information represented by the structure of the graph.

As interaction networks are graphs, where each node represents for example, a protein and each edge represents the presence of an interaction, Conventionally there are two ways of measuring similarity between graphs. One approach is to perform a pair wise comparison of the nodes and/or edges in two networks, and calculate an overall similarity score for the two networks from the similarity of their components. This approach takes time quadratic in the number of nodes and edges, and is thus computationally feasible even for large graphs. However, this strategy is flawed in that it completely neglects the structure of the networks, treating them as sets of nodes and edges instead of graphs. A more principled alternative would be to deem two networks similar if they share many common substructures, or more technically, if they share many common subgraphs. To compute this, however, we would have to solve the so-called

DOI : 10.5121/ijitcs.2012.2203

subgraph isomorphism problem which is known to be NP-complete, i.e., the computational cost of this problem increases exponentially with problem size, seriously limiting this approach to very small networks. Many heuristics have been developed to speed up sub graph isomorphism by using special canonical labelings of the graphs; none of them, however, can avoid an exponential worst-case computation time.

2. PRIMER ON GRAPH THEORY

A graph G consists of a set of nodes (or vertices) V and edges E. Let n denotes the number of nodes in a graph and m the number of edges in a graph. An attributed graph is a graph with labels on nodes and/or edges; we refer to labels as attributes. In our case, attributes will consist of pairs of the form (attribute-name, value). The unnormalized adjacency matrix A of G is defined as

$$[A]_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise} \end{cases}$$

where vi and vj are nodes in G. If G is weighted then, A can contain non-negative entries other than zeroes and ones. i.e Aij e (0,¥) if (vi,vj) e E and zero otherwise. Let D be a nxn diagonal matrix with entries Dii = Sj Aij. The matrix P := AD-1 is called the normalized adjacency matrix. Let X be a set of labels which includes the special label e. An edge labeled graph G is associated with a label matrix L e X nxn , such that Lij = iff (vi,vj) . A walk w of length k – 1 in a graph is a sequence of nodes v1, v2, · · · ,vk where (vi−1, vi) e E for 1 < i £ k. w is a path if vi ¹ vj iff i ¹ j, j e {1, . . . , k} . Alternatively, walks are often referred to as paths; paths are then named simple, unique or loopless paths, which may lead to some confusion. To clarify the difference for the remainder of this article, we define a path to be a walk without repetitions of nodes. A cycle is a walk with v1 = vk, a simple cycle does not have any repeated nodes except for v1. A Hamilton path is a path that visits every node in a graph exactly once. An Euler path is a path that visits every edge in a graph exactly once.If a graph is undirected, and that the vertices and edges in a graph are labeled. The labels of an edge e and a vertex v are denoted by l(e) and l(v) respectively. Each vertex (or edge) of a graph is not required to have a unique label and the same label can be assigned to many vertices (or edges) in the same graph. Given a graph G = (V, E), a graph Gs = (Vs , Es) is a subgraph of G if Vs Í V and Es ÍE, and is denoted by Gs ÍG. The sub graph Gs is said to be covered by G. If a sub graph Gs Í G is isomorphic to another graph H, then Gs is called an embedding of H in G. In this report, a sub graph is often called a pattern. The total number of embeddings of Gs in a graph G is called the raw frequency of Gs . Two graphs G1 = (V1, E1) and G2 = (V2, E2) are isomorphic if they are topologically identical to each other, that is, there is a vertex mapping from V1 to V2 such that each edge in E1 is mapped to a single edge in E2 and vice versa. In the case of labeled graphs, this mapping must also preserve the labels on the vertices and edges. When a set of graphs {Gi } are isomorphic to each other, they all are said to belong to the same equivalence class. When the equivalence class of Gi represents an edge, the class is called an edge-type. Given two graphs G1 = (V1, E1) and G2 = (V2, E2), the problem of sub graph isomorphism is to find an isomorphism between G2 and a sub graph of G1. In other words, the sub graph isomorphism problem is to determine whether or not G2 is embedded in G1.

Given a sub graph Gs and a graph G, two embeddings of Gs in G are called identical if they use the same set of edges of G, edge-disjoint if they do not have any edges of G in common, and vertex-disjoint if no vertices of G in common. Given a set of all embeddings of a particular sub graph Gs in a graph G, the overlap graph of Gs is a graph obtained by creating a vertex for each non-identical embedding and creating an edge for each pair of non-vertex-disjoint embeddings. Contraction of an edge e = uv of a graph G = (V, E) is to merge two endpoints u and v together into a new vertex w by removing the edge e, while keeping all the other edges incident to u and v. The remaining edges that used to be incident to either u or v are connected to w after the contraction. The newly added vertex w represents the original edge e. Note that, if there are multiple edges between two vertices u and v, the contraction of e removes only e. The rest of the multiple edges between u and v become loops around w after the contraction. A subtree of an undirected graph G is an acyclic connected subgraph of G. A subtree T is a spanning tree of G if T contains all nodes in G. Given a graph G, there are many spanning trees. A canonical spanning tree of G is a maximal spanning tree defined on a total order on the trees. A spanning tree is a tree that has paths connecting each node with every other node of the graph. A trie is a data structure that stores the information about the contents of each node in the path from the root to the node, rather than the node itself.

3. OVERVIEW OF FREQUENT SUBGRAPH MINING

This section provides a generic overview of the process of FSM. Any frequent subgraph mining process involves 3 aspects , i) graph representation ii) subgraph Enumeration and iii) frequency counting.

3.1 Graph Representations

The simplest mechanism whereby a graph structure can be represented is by employing an adjacency matrix or adjacency list. Using an adjacency matrix the rows and columns represent vertexes, and the intersection of row i and column j represents a potential edge connecting the vertexes vi and vj . The value held at intersection < i, j > typically indicates the number of links from vi to vj . However, the use of adjacency matrices, although straightforward, does not lend itself to isomorphism detection, because a graph can be represented in many different ways depending on how the vertexes (and edges) are enumerated (Washio & Motoda 2003). With respect to isomorphism testing it is therefore desirable to adopt a consistent labeling strategy that ensures that any two identical graphs are labeled in the same way regardless of the order in which vertexes and edges are presented (i.e. a canonical labeling strategy). A canonical labeling strategy defines a unique code for a given graph.

Canonical labeling facilitates isomorphism checking because it ensures that if a pair of graphs are isomorphic, then their canonical labeling will be identical [5] (Kuramochi & Karypis 2001). One simple way of generating a canonical labeling is to flatten the associated adjacency matrix by concatenating rows or columns to produce a code comprising a list of integers with a minimum (or maximum) lexicographical ordering imposed. To further reduce the computation resulting from the permutations of the matrix, canonical labeling are usually compressed, using what is known as a vertex invariant scheme (Read & Corneil 1977), that allows the content of an adjacency matrix to be partitioned according to the vertex labels. Various canonical labeling schemes have been proposed, some of the more significant are described in this subsection.

Minimum DFS Code (M-DFSC): There are a number of variants of DFS encodings, but essentially each vertex is given a unique identifier generated from a DFS traversal of a graph (DFS subscripting). Each constituent edge of the graph in the DFS code is then represented by a 5-tuple: (i, j, li, le, lj), where i and j are the vertex identifiers, li and lj are the labels for the corresponding vertexes, and le is the label for the edge connecting the vertexes. Based on the DFS lexicographic order, the M-DFSC of a graph g can be defined as the canonical labeling of g [11] (Yan & Han 2002).

Canonical Adjacency Matrix (CAM): Given an adjacency matrix M of a graph g, an encoding of M can be obtained by the sequence obtained from concatenating the lower(or upper) triangular entries of M, including entries on the diagonal. Since different permutations of the set of vertexes correspond to different adjacency matrices, the canonical (CAM) form of g is defined as the maximal (or minimal) encoding. The adjacency matrix from which the canonical form is generated defines the Canonical Adjacency Matrix or CAM[4][5][8](Inokuchi et al. 2000,2002; Kuramochi & Karypis 2001; Huan et al. 2003).

3.2 Subgraph Enumeration

The current methods for enumerating all the subgraphs might be classified into two categories: one is the join operation adopted by FSG[5] and AGM [4] and another one is the extension operation . The major concerns for the join operation are that a single join might produce multiple candidates and that a candidate might be redundantly proposed by many join operations. The concern for the extension operation is to restrict the nodes that a newly introduced edge may attach to. Equivalence class based extension (Zaki 2002, 2005) is founded on a DFS-LS representation for trees. Basically, a (k + 1)-subtree is generated by joining two frequent k-subtrees. The two k subtrees must be in the same equivalence class . An equivalence class consists of the class prefix encoding, and a list of members. Each member of the class can be represented as a (l, p) pair, where l is the k-th vertex label and p is the depth-first position of the k-th vertex's parent. It is verified, in Zaki (2002), that all potential (k + 1)-subtrees with the prefix [C] of size (k − 1) can be generated by joining each pair of members of the same equivalent class [C]. Equivalence classes can be based on either prefix or suffix.

3.3 Frequency Counting

Two Methods are used for graph counting: Embedding lists (EL) and Recomputed embeddings (RE)**.** For graphs with a single node we store an embedding list of all occurrences of its label in the database. For other graphs a list is stored of embedding tuples that consist of (1) an index of an embedding tuple in the embedding list of the predecessor graph and (2) the identifier of a graph in the database and a node in that graph. The frequency of a structure is determined from the number of different graphs in its embedding list. Embedding lists are quick, but they do not scale very well to large databases. The other approach is based on maintaining a set of active" graphs in which occurrences are repeatedly recomputed.

4 A SURVEY OF FSM ALGORITHMS

The frequent subgraph discovery problem has been addressed from many directions using various approaches, including a priori strategy and pattern growth approach. Also the algorithms differ in the type of input graphs, search strategy they use and method of representation of graphs etc. Hence, there exist many algorithms based on different approaches. This makes the task of identifying a suitable algorithm for any given application scenario an involved process. In this paper, we present a survey and propose to establish a framework for classification of these algorithms to help in understanding and analyzing various properties and limitations of few of these algorithms. A quick reference of 26 frequent subgraph mining algorithms is presented in Table 1 and Table 2. Other than that five algorithms which are extensions of existing algorithms is also studied in the following sections.

Table 1. Algorithms based on Pattern growth approach

S.No	**Algorithm**	**Input type**	**Graph represen-tation**	**Subgraph generation**	**Frequency counting**	**Nature of output**	**Limitations**
1.	MOFA [2002]	Set of graphs	Adjace-ncy list	Rightmost extension	DFS	All frequent subgraphs	Frequent graphs not exactly frequent.
2.	GSpan [2002]	Set of graphs	Adjacency list	Rightmost extension	DFS	frequent graphs	Not scalable
3.	CloseGraph [2003]	Set of graphs	Adjacency list	Rightmost extension	DFS	Closed Connected frequent graphs	Failure detection takes lot of time overhead
4.	Gaston [2004]	Set of graphs	Hash table	Extension	Embedding lists	Maximal frequent sugraphs	Interesting patterns may be lost
5.	SUBDUE [2005]	Single large graph	Adjacency matrix	Level-wise search	MDL	Complete set of frequent subgraphs	Extremely small no. of patterns
6.	Gapprox [2006]	Single large graph	Edge Triplet	Approximation	DFS	Frequent approximate patterns	
7.	HybridGMiner[2006]	Set of Graphs	Adjacency list	Embedding Lists	DFS	Frequent subgraphs	Extension of existing frequent pattern is complicated

							and needs extra memory
8.	MSPAN [2009]	Set of graphs	Adjacency list	Rightmost extension	DFS	Frequent subgraphs	Works on labelled graphs only.
9.	FCP Miner [2009]	Set of graphs	Canonical Labeling	Rightmost extension	DFS	Frequent Correlated subgraphs	Works on labelled graphs only.
10.	RING [2009]	Set of graphs	Adjacency Matrix	Invariant Vectors	DFS	Representative graphs	Needs post processing
11.	SCMiner [2009]	Set of partially labelled graphs	Canonical Labeling	Rightmost extension	DFS	Partially labelled frequent subgraphs	Works on labelled graphs only.
12.	Graphsig [2009]	Set of graphs	Feature Vector	Extension	DFS	Frequent significant patterns	Needs post processing
13.	JPMiner [2009]	Set of graphs	Adjacency list	Rightmost extension	DFS	Frequent jump patterns	Some -times much smaller set of jump patterns.
14.	RP-GD [2010]	Set of graphs	Adjacency list	Rightmost extension	DFS	Representative graphs	Time for summari-zing the patterns is more than that for mining
15.	TSP [2010]	Set of graphs	Adjacency list	Extension	TSP tree	Closed Temporal frequent subgraphs	Extra overhead to check closed property

16.	RP-FP [2010]	Set of graphs	Adjacency list	Rightmost extension	DFS	Representative graphs	Overhead for summarizing

Table 2. Algorithms based on Apriori approach

S. No	**Algorithm**	**Input type**	**Graph representation**	**Candidate generation**	**Frequency counting**	**Nature of output**	**Limitations**
1.	FARMER [1999]	Set of graphs	Trie structure	Level-wise search ILP	Trie data structure	Frequent subgraphs	Inefficient
2.	HSIGRAM [1999]	Single large graph	Adjacency matrix	Iterative merging	Maximal independent set	Frequent subgraphs	Ineffecient
3.	AGM [2000]	Graph database	Adjacency matrix	Vertex extension	Canonical labelling	Frequent subgraphs	Np-complete
4.	FSG [2001]	Set of graphs	Adjacency list	One edge extension	Transaction identifier (TID) lists	Frequent connected subgraphs	Largely distinct labels on edges needed
5.	FFSM [2003]	Set of graphs	Adjacency matrix	Merging and extension	Sub-optimal CAM tree	Frequent subgraphs	Np-complete
6.	ISG [2004]	Set of graphs	Edge triplet	Edgetriplet extension	TID lists	Maximal Frequent subgraphs	Incomplete set of Graphs
7.	SPIN [2004]	Set of graphs	Adjacency matrix	Join Operation	Canonical Spanning Tree	Maximal frequent subgraphs	Needs entire DB scan
8.	GREW [2004]	Single large graph	Sparse graph representation.	Iterative merging	Maximal independent set	Maximal frequent subgraphs	Misses many interesting patterns

9.	Dynamic GREW [2005]	Dynamic graphs	Sparse graph representation.	Iterative merging	Suffix trees	Dynamic patterns in frequent subgraphs.	Extra overhead to identify dynamic patterns
10	MUSE [2009]	Uncertain set of graphs	Adjaceny Matrix	Disjunctive normal forms	DFS coding scheme	Frequent subgraphs	Frequent subgraphs are not exact.

4.1 Classification based on Algorithmic approach.

It is widely accepted that FSM techniques can be divided into two categories: (i) Apriori-based approaches, and (ii) pattern growth-based approach.

4.1.1 Apriori Based Approach

Apriori-based frequent substructure mining algorithms share similar characteristics with Apriori-based frequent itemset mining algorithms. The search for frequent graphs starts with graphs of small “size”, and proceeds in a bottom-up manner. At each iteration, the size of newly discovered frequent substructures is increased by one. These new substructures are first generated by joining two similar but slightly different frequent subgraphs that were discovered already. The frequency of the newly formed graphs is then checked. The Apriori-based algorithms have considerable overhead when two size-k frequent substructures are joined to generate size-(k+1) graph candidates. Typical Apriori-based frequent substructure mining algorithms are discussed in the following paragraphs.

The AGM[4] algorithm uses a vertex-based candidate generation method that increases the substructure size by one vertex at each iteration. Two size-k frequent graphs are joined only when the two graphs have the same size-(k – 1).

ISG [15] represents graphs in an entirely different manner. It transforms the input set of graphs into item sets which are then represented using edge triplet. ISG uses a approach known as edge triplet extension in which a discovered item set is extended by adding one edge triplet in each iteration. ISG carries out frequent subgraph discovery by transforming graphs into itemsets followed by frequent itemset discovery, which is also apriori-based. The resultant frequent itemsets are transformed back to subgraphs. In pattern-growth approach, the subgraph generation is carried out by extending the previously discovered subgraph by one node or one edge. ISG use transaction identifier (TID) lists for frequency counting. Each frequent subgraph has a list of transaction identifiers which support it. For computing frequency of a k subgraph, the intersection of the TID lists of (k – 1) subgraphs is computed.

FARMER[18] uses trie for graph representation. In level-wise search, the algorithm finds a subgraph and then enumerates the instances of the subgraph by one adjacent edge in all possible ways. FARMER follow this mechanism for subgraph generation. FARMER, which has been developed as an enhancement to WARMR, an earlier developed algorithm which works on the basis of ILP approach, is based on a combination of a priori and ILP approaches. FARMER uses the trie data structure for frequency computation also.

HSIGRAM [22] uses adjacency matrix representation of graph. HSIGRAM use iterative merging for subgraph generation. In case of HSIGRAM the aim is to find the maximal independentset of a graph which is constructed out of the embeddings of a frequent subgraph so as to evaluate its frequency.

Huan, wang and Prince [7] in 2003 proposed a novel subgraph mining algorithm: FFSM, which employs a vertical search scheme within an algebraic graph framework.It uses a restricted join operation to generate candidates and stores embeddings to avoid explicit subgraph isomorphism testing. It uses a sub-optimal canonical adjacency matrix tree for counting the frequency. Their studies on synthetic and real datasets demonstrated that FFSM achieves a substantial performance gain over the start-of-the art subgraph mining algorithm gSpan.

One fundamental challenge for mining recurring subgraphs from semi-structured data sets is the overwhelming abundance of suchpatternsc. In large graph databases, the total number of frequent subgraphs can become too large to allow a full enumeration using reasonable computational resources. Jun Huan, Wei WangPrins, Jiong Yang, Jan [8] proposed a new algorithm, Spin that mines only maximal frequent subgraphs, i.e. subgraphs that are not a part of any other frequent subgraphs. This may exponentially decrease the size of the output set in the best-case; in our experiments on practical data sets, mining maximal frequent subgraphs reduces the total number of mined patterns by two to three orders of magnitude. It first mines all frequent trees from a general graph database and then reconstructs all maximal subgraphs from the mined trees. SPIN offered very good scalability to large graph databases and at least an order of magnitude performance improvement in synthetic graph data sets. The efficiency of the algorithm is also confirmed by a benchmark chemical data set. This algorithm of compressing large number of frequent subgraphs to a much smaller set of maximal subgraphs.lt is used to investigate demanding applications such as finding structure patterns from proteins in the future.

Michihiro Kuramochi and George Karypis [9] in 2004 proposed a heuristic algorithm called GREW to overcome the limitations of existing complete or heuristic frequent subgraph discovery algorithms. GREW is designed to operate on a large graph and to find patterns corresponding to connected subgraphs that have a large number of vertex-disjoint embeddings. Their experimental evaluation showed that GREW is efficient, can scale to very large graphs, and find non-trivial patterns that cover large portions of the input graph and the lattice of frequent patterns.

Karsten M. Borgwardt, Hans-Peter Kriegel, Peter Wackersreuther [28] investigated how pattern mining on static graphs can be extended to time series of graphs, ie. dynamic graphs. They proposed a framework into which Existing subgraph mining algorithms can be easily integrated and handle dynamic graphs. Experimental results on real-world data confirm the practical feasibility of their approach. In particular, we are looking for subgraphs that are topologically frequent within a large graph and that show insertions and deletions of edges in the same temporal order. It might be used to study frequent motifs in protein-protein interaction dynamics, as well as in social or telecommunication networks.

Lini T Thomas Satyanarayana R ValluriKamalakar Karlapalem [13] in 2006 proposed an algorithm MARGIN that mines maximal frequent subgraphs. MARGIN- Maximal frequent mining has triggered much interest since the size of the set of maximal frequent subgraphs is much smaller to that of the set of frequent subgraphs. The set of candidate subgraphs which are likely to be maximally frequent are the set of -edge frequent subgraphs that have a z-edge infrequent supergraph. The Margin algorithm computessuch a candidate set efficiently and finds the maximal subgraphs by a post-processing step. They have proved that the performance of the Margin algorithm is 20 times faster than gSpan for certain datasets.

ZhaonianZou, Jianzhong Li, and Shuo Zhang [17] in 2010 proposed an algorithm for Mining Frequent Subgraph Patterns from Uncertain Graph Data. In many real applications, graph data is subject to uncertainties due to incompleteness and imprecision of data. Mining such uncertain graph data is semantically different from and computationally more challenging than mining conventional exact graphdata. A novel model of uncertain graphs is presented, and the frequent subgraph pattern mining problem is formalized by introducing a new measure, called expected support. An approximate mining algorithm called MUSE (Mining Uncertain Sub graph pattErns), is proposed to find a set of approximately frequent subgraph patterns by allowing an error tolerance on expected supports of discovered subgraph patterns. The algorithm uses efficient methods to determine whether a subgraph pattern can be output or not and new pruning method to reduce the complexity of examining subgraph patterns. Analytical and experimental results showed that the algorithm is very efficient, accurate, and scalable for large uncertain graph databases. This is the first algorithm to investigate the problem of mining frequent sub graph patterns from uncertain graph data.

4.1.2 Pattern-growth approach

In order to avoid the overhead of apriori algorithms, non-Apriority-based algorithms have been developed, most of which adopt the pattern-growth methodology, as discussed below. Pattern-growth-based graph pattern mining algorithms include gSpan by Yan and Han (2002), MoFa by Borgelt and Berthold (2002), FFSM by Huan et al. (2003), SPIN by Huan et al. (2004), and Gaston by Nijssen and Kok (2004). These algorithms are inspired by PrefixSpan (Pei et al. 2001), TreeMinerV (Zaki 2002), and FREQT (Asai et al. 2002) at mining sequences and trees, respectively. The pattern-growth mining algorithm extends a frequent graph by adding a new edge, in every possible position. A potential problem with the edge extension is that the same graph can be discovered many times. The gSpan [11] algorithm solves this problem by introducing a right-most extension technique, where the only extensions take place on the right-most path. A right-most path is the straight path from the starting vertex v0 to the last vertex vn, according to a depth-first search on the graph. Typical pattern growth algorithms are discussed in the following paragraphs.

Frequency counting process for Gaston is carried out with the help of embedding lists, where all the occurrences of a particular label are stored in the embedding lists.

Borgelt and Berthold [10] in 2002 presented an algorithm Mofa to find fragments in a set of molecules that help to discriminate between different classes for instance, activity in a drug discovery context. Yan and Han [11] in 2002 investigated new approaches for frequent graph-based pattern mining in graph datasets and proposed a novel algorithm called gSpan. gSpan is a graph-based substructure pattern mining. This discovered frequent substructures without candidate generation.

Yan and Han [12] in 2003 proposed to mine closed frequent graph patterns. A graph g is closed in a database if there exists no proper subgraph of g that has the same support as g. A closed graph pattern mining algorithm, CloseGraph, is developed by exploring several interesting looping methods. Their performance studies shown that CloseGraph not only dramatically reduces unnecessary subgraphs to be generated, but also substantially increases the efficiency of mining, especially in the presence of large graph patterns.

Yong Liu, Jianzhong Li, Hong Gao [23] in 2009 studied the problem of mining frequent jump patterns from graph databases. They have showed that Mining frequent jump patterns can dramatically reduce the number of output graph patterns, and still capture interesting graph patterns. By integrating the operation of checking jump patterns into the well-known DFS code tree enumeration framework, they presented an efficient algorithm JPMiner for this new problem. Their experimental evaluation of JPMiner using both real and synthetic datasets, showed that the number of frequent jump patterns is much smaller than that of closed frequent graph patterns, and also JPMiner is efficient and scalable in mining frequent jump patterns. Chia-HuiChang and Cheng-Tao Ho in 2006 proposed an hybrid algorithm HybridGMiner which is based on pattern growth approach. It combines the embedding listing and canonical form to take the advantage of generating new frequent patterns from existing patterns. It is a hybrid of Mofa and gspan.

Most of existing frequent subgraph mining algorithms are used to deal with undirected unlabeled marked graph. A few of them aim at directed graph or labeled graph because it is very complex to consider that. But in the real world, a lot of connections have directions and labels, so directed labeled graph mining is more meaningful. Yuhua Li Quan Lin and Duan Yanan [25] Bi in 2009 analyzed a financial network by modelling it as a directed weighted graph. They proposed a new algorithm mSpan for directed labeled graph frequent pattern mining. Based on FP-growth, the algorithm gets a minimum edge code and an abstract node code sequence to identify a directed graph pattern uniquely through minimum extension. It also solved the graph pattern isomorphic problem and the redundant extension problem. Their experiment showed that mSpan can mine all frequent directed, labeled graph patterns.

Shijie Zhang , Jiong Yang and Shirong Li [31] in 2009 proposed RING an integrated method for
frequent representative subgraph mining . It involves a two-step process. The first stage is to compute the pattern distribution. It randomly mine a set of frequent subgraph patterns, cluster the patterns, and select the centers of the clusters as the initial representative patterns. Based on this, we will know how the frequent patterns are distributed approximately. In the second stage, it adopts a depth first searching algorithm to mine representative to a pre-set space limit.

Houqun YANG, Zhongshi HE, Xing WU [32] in 2009 proposed an algorithm based on node pruning, path pruning and structure pruning that applied pattern-weakening support constraints in mining partially labeled subgraphs. The main idea is to push forward the check conditions into the process of mining so that user can interact with the process of mining partially labeled subgraphs, and reach the final object for improving mining efficiency. The results showed that it can eliminate effectively subgraph mining search space by pushing forward the constraint into the process of mining and to a certain extent reduce the mining cost in both space and time.

Cheng-Te Li, Hsun-Ping Hsieh [26] proposed a novel algorithm, TSP-algorithm (Temporal Subgraph Patterns algorithm) to mine the patterns which contain temporal information and forms a connective subgraph. The proposed method recursively grows the patterns in a depth-first search manner. Since the TSP-algorithm only needs to scan the database once and does not generate unnecessary candidates, the experiment results showed that the TSP-algorithm outperforms the modified Apriori on time-efficiency and memory usage in both synthetic and real datasets.

Jianzhong Li, Yong Liu, and Hong GaoWe [27] in 2011 investigated the problem of summarizing frequent subgraphs by a smaller set of representative patterns. They showed that some special graph patterns, called _-jump patterns , must be representative patterns. Based on the fact,they devised two algorithms, RP-FP and RP-GD, to mine a representative set that summarizes frequent subgraphs. RP-FP derives a representative set from frequent closed subgraphs, whereas RP-GD mines a representative set from graph databases directly. Three novel heuristic strategies, Last-Succeed-First-Check, Reverse-Path-Trace, and Nephew-Representative-Based-Cover, are proposed to further improve the efficiency of RP-GD. RP-FP can provide a tight ratio bound but has heavy computation cost. RP-GD cannot provide a ratio bound guarantee but is more efficient than RP-FP.

Sayan Ranu , Ambuj K. Singh [34] proposed a highly scalable technique, called GraphSig, to mine significant subgraphs from large graph databases. It converts each graph into a set of feature vectors where each vector represents a region within the graph. Domain knowledge is used to select a meaningful feature set. It accesses only a small portion of the exponential search space, and groups candidate subgraphs into sets based on their similarity. As a result, frequent subgraph mining can be performed on each set with a high frequency threshold. It is experimentally proved to be scalable.

4.2 Classification based on Search strategy

There are two basic search strategies employed for finding out frequent subgraphs: the breadth first search (BFS) strategy and the depth first search (DFS) strategy.

4.3 Classification based on Nature of the input

The algorithms are of two types based on the exactness of the input they take. The first type takes in a exact graph sets as input, whereas the second type takes a uncertain set of graphs as input. Another possibility is based on the type of the graph. The first type takes in a single large graph as input, whereas the second type takes a set of small graphs as input. The third is based on the correctness of the graph data where it can be accurate or uncertain.

4.4 Classification based on Completeness of the output

Based on the set of the frequent subgraphs discovered, the algorithms are of two types. The first type returns the complete set of frequent subgraphs, whereas the second type returns a partial set of frequent subgraphs.

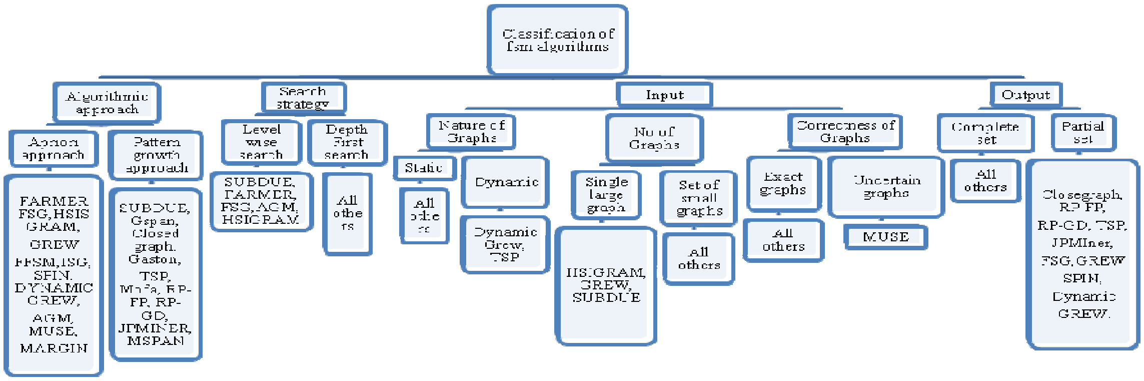

Figure 1. Classification of FSM algorithms

4.5 Extension of Existing algorithms

Many algorithms have been proposed, which are either extensions of the existing algorithms or a hybrid of two or more existing algorithms. FCP-Miner[29], is one such a fast approximate algorithm, which uses a very effective skipping mechanism that eliminates the processing of majority of graphs during the mining. It is shown that the missing pairs of frequent correlated sub graphs due to approximation are only a small set of marginally correlated pairs. Extensive experiments verified that the algorithm is both efficient and effective.

A Hybrid fragment mining algorithm[30] which is a combination of Mofa and FSG is proposed in 2004. It uses a search strategy which is a combination of BFS and DFS. It is proved to be more efficient than both Mofa and FSG.

Chen Chen, Xifeng Yan, Feida Zhu, Jiawei Han[33], investigated the problem of mining frequent approximate patterns from a massive network and proposed a method called gApprox. gApprox not only finds approximate network patterns, which is the key for many knowledge discovery applications on structural data, but also enriches the library of graph mining methodologies by introducing several novel techniques.

A new general framework, called gPrune, is proposed by Feida Zhu*y* Xifeng Yan*y* Jiawei Han*y* Philip S. Yuz [35] to incorporate all the constraints in such a way that they recursively reinforce each other through the entire mining process. A new concept, Pattern-inseparable Data-antimonotonicity, is proposed to handle the structural constraints unique in the context of graph, which, combined with known pruning properties, provides a comprehensive and unified classification framework for structural constraints. The exploration of these anti - monotonicities in the context of graph pattern mining is a significant extension to the known classification of constraints, and deepens our understanding of the pruning properties of structural graph constraint. The major advantage of this algorithm is it can be applied to both apriori and pattern growth approach.

5. APPLICATION DOMAINS

Graph mining is useful in the areas of chemo-informatics, Bio-informatics, study of biological networks, Web analysis, Traffic analysis in telecommunication networks, social human behavior analysis, financial network analysis and wireless networks. In this paper we present a classification of algorithms based on their suitability to specific application domains.

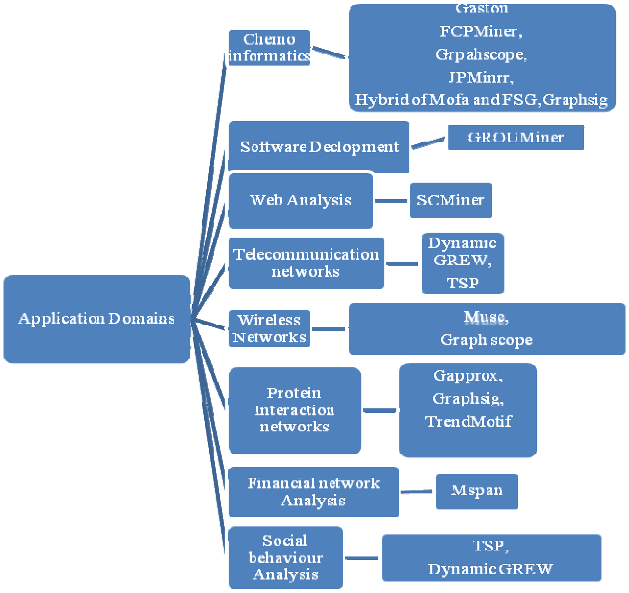

Figure 2. Classification of FSM algorithms based on Application Domains

6. RESEARCH DIRECTIONS

We have abundant literature published in research into frequent pattern mining. But still there are several critical research problems that need to be solved before frequent pattern mining can become a cornerstone approach in data mining applications. First, the most focused and extensively studied topic in frequent pattern mining is perhaps scalable mining methods. Second is the efficiency of the frequent subgraph mining algorithms. Third is not on whether we can derive the complete set of frequent patterns under certain constraints efficiently but on whether we can derive a compact but high quality set of patterns that are most useful in applications and whether we can mine such patterns directly and efficiently. Fourth, to make frequent pattern mining an essential task in data mining. Classification is an essential task in data mining. We can generate frequent patterns in such a way that, they can become input for classification or clustering models.

7. CONCLUSION

In this paper, we present a brief overview of the current status and future directions of frequent pattern mining. There are various inter-disciplinary domains like chemo informatics, bioinformatics etc. where mining of recurrent patterns across large collection of networks is required. Due to increasing size and complexity of patterns in there is a need for efficient graph mining algorithm. With over a decade of extensive research, there have been hundreds of research publications and tremendous research, development and application activities in this domain. Many algorithms for frequent subgraph mining have been proposed so far. Most of the algorithms, they focus only on a static set of graphs. Very few algorithms are for mining patterns from dynamic set of graphs. Also all the algorithms proposed so far, outperform each other, either in terms of memory requirements or in terms of few orders of magnitude of computation time. None of them completely address the issue of NP-completeness of the subgraph mining problem. Also the algorithms mine either a specific set of patterns or a complete set of patterns which may not be significant. So there is a need for an efficient algorithm which can mine significant patterns specific to the application, both from a static or dynamic set of graphs in less than polynomial time. Also the mined interesting patterns can used as input to other data mining tasks such as for classification or clustering for further knowledge discovery. Hence lot of research is required towards the improvements suggested.

8. REFERENCES

[1] D. J. Cook and L. B. Holder, "Substructure discovery using minimum description length and background knowledge" Journal of Artificial intelligence Research, 1, 1994, 231-255.

[2] S. Fortin. The graph isomorphism problem. Technical Report TR96-20, Department of Computing Science, University of Alberta, 1996.

[3] L. Dehaspe, H. Toivonen, and R. D. King. Finding frequent substructures in chemical compounds. In R. Agrawal, P. Stolorz, and G. Piatetsky-Shapiro, editors, Proc. of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-98), pages 30–36. AAAI Press, 1998.

[4] A. Inokuchi, T.Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In PKDD'00.

[5] M.Kuramochi and G. Karypis . Frequent Subgraph Discovery. In ICDM'01.

[6] M. Deshpande, M. Kuramochi, and G. Karypis. Frequent sub-structure based approaches for classifying chemical compounds. In Proc. of 2003 IEEE International Conference on Data Mining (ICDM), 2003.

[7] J. Huan, W.Wang, and J. Prins. Efficient mining of frequent subgraph in the presence of isomorphism. UNC computer science technique report TR03-021, 2003.

[8] J. Huan, W. Wang, J. Prins, and J. Yang. Spin: Mining maximal frequent subgraphs from graph databases. UNC Technical Report TR04-018, 2004.

[9] M. Kuramochi and G. Karypis. GREW A Scalable frequent subgraph discovery algorithm. Technical Report 04-024, University of Minnesota, Department of Computer Science, 2004.

[10] C. Borgelt and M. R. Berhold. Mining molecular fragments: Finding relevant substructures of molecules. Proc. 2nd IEEE Int'l Conf. Data Mining (ICDM '02), pp. 51-58, 2002.

[11] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. Proc. 2nd IEEE Int'l Conf. Data Mining (ICDM '02), pp. 721-724, 2002.

[12] X. Yan and J. Han. CloseGraph: Mining closed frequentgraph patterns. Proc. 9th ACM SIGKDD Int'l Conf. Knowledge Discov-ery and Data Mining (KDD '03), pp. 286-295, 2003.(closegraph)

[13] L. T. Thomas, S. R. Valluri, and K. Karlapalem. Margin:Maximal frequent subgraph mining. Proc. 6th IEEE Int'l Conf. Data mining (ICDM '06), pp. 1097-1101, 2006.

[14] M. Kuramochi and G. Karypis. Grew-a scalable frequent subgraph discovery algorithm. In ICDM, pages 439–442,2004.

[15]. Thomas, L., Valluri, S. and Karlapalem, K., Isg: Itemset based subgraph mining. Technical Report, IIIT, Hyderabad, December2009.

[16] Kuramochi, M. and Karypis, G., Finding frequent patterns in alarge sparse graph. Data Min. Knowledge Discovery, 2005, (3),243–271.

[17] ZhaonianZou, Jianzhong Li, Hong Gao, and Shuo Zhang : Frequent Subgraph Patterns from Uncertain Graph Data. IEEE Transactions On Knowledge And Data Engineering, Vol. 22, No. 9, September 2010.

[18] Nijssen, S. and Kok, J., Faster association rules for multiple relations. In IJCAI'01: Seventeenth International Joint Conference on Artificial Intelligence, 2001, vol. 2, pp. 891–896.

[19] Nijssen, S. and Kok, J., A quickstart in frequent structure mining can make a difference. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2004, pp. 647–652.

[20] Chang Hun You, Lawrence B. Holder and Diane J. Cook :Graph-based Data Mining in Dynamic Networks: Empirical Comparison of Compression-based and Frequency-based Subgraph Mining inInternational Conference on Data Mining Workshops IEEE , 2004.

[21] Cordella, L.P., Foggia, P., Sansone, C. and Vento, M. 2001. An Improved Algorithm for Matching Large Graphs, In Proceedings of the 3rd IAPR-TC15 Workshop on Graph-based Representation in Pattern Recognition, 149–159.

[22] Chuntao Jiang, Frans Coenen and Michele Zito, A Survey of Frequent Subgraph Mining Algorithms:The Knowledge Engineering Review, Vol. 00:0, 1–31.c 2004.

[23] Yong Liu, Jianzhong Li, Hong Gao, JPMiner: Mining Frequent Jump Patterns From Graph Databases. In the proceedings of Sixth International Conference on Fuzzy Systems and Knowledge Discovery 2009.

[24] Varun Krishna, N. N. R. Ranga Suri and G. Athithan,A comparative survey of algorithms for frequent subgraph discovery, Current Science, Vol. 100, No. 2, 25 January 2011

[25] Yuhua Li Quan Lin Gang Zhong Dongsheng Duan Yanan Jin Wei Bi, A Directed Labeled Graph Frequent Pattern Mining Algorithm based on Minimum Code. In the proceedings of Third International Conference on Multimedia and Ubiquitous Engineering 2009.

[26] Hsun-Ping Hsieh, Cheng-Te Li, Mining Temporal Subgraph Patterns in Heterogeneous Information Networks: In the proceedings of IEEE International Conference on Social Computing / IEEE International Conference on Privacy, Security, Risk and Trust.

[27] Jianzhong Li, Yong Liu, and Hong Gao, Efficient Algorithms for Summarizing Graph Patterns: IEEE Transactions On Knowledge And Data Engineering, Vol. 23, No. 9, September 2011.

[28] Bianca Wackersreuther, Peter Wackersreuther, Annahita Oswald : Frequent Subgraph Discovery in Dynamic Networks, ACM 978-1-4503-0214-2 2010.

[29] Yiping Ke, James Cheng, Jeffrey Xu Yu Efficient Discovery of Frequent Correlated Subgraph Pairs: In the proceedings of Ninth IEEE International Conference on Data Mining 2009.

[30] Thorsten Meinl, Michael R. Berthold, Hybrid fragment miining with *MoFa* and FSG: In the proceedings of IEEE International Conference on Systems, Man and Cybernetics 2004.

[31] Shijie Zhang , Jiong Yang , Shirong Li RING: An Integrated Method for Frequent Representative Subgraph Mining: In the proceedings of Ninth IEEE International Conference on Data Mining 2009.

[32] Houqun YANG, Zhongshi HE, Xing WU, Mining Partially Labeled Subgraphs Using Support Constraints: In the proceedings of Sixth International Conference on Fuzzy Systems and Knowledge Discovery 2009.

[33] Chen Chen, Xifeng Yan, Feida Zhu, Jiawei Han, gApprox: Mining Frequent Approximate Patterns from a Massive Network.

[34] Sayan Ranu , Ambuj K. Singh, GraphSig: A Scalable Approach to Mining Significant Subgraphs in Large Graph Databases, In the proceedings IEEE International Conference on Data Engineering 2009.

Authors

1. K.Lakshmi, Asst. Prof /Dept of MCA, Sir M.Visvesvaraya Institute of Technology, Bangalore.

2. Dr. T. Meyyappan, M.Sc., M.Phil., M.B.A., Ph.D.,
Associate Professor,
Department of Computer Science and Engineering, Alagappa University,Karaikudi,
Tamilnadu.