A comparative survey of algorithms for frequent subgraph discovery

Varun Krishna*, N. N. R. Ranga Suri and G. Athithan

Centre for Artificial Intelligence and Robotics, DRDO Complex, C.V. Raman Nagar, Bangalore 560 093, India

Graph mining is a well-explored area of research where frequent subgraph discovery is an important problem. To get an understanding of various frequent subgraph discovery algorithms and to assess their suitability to various application scenarios, it is important to establish a common framework for their study. The present article addresses this need by bringing out a classification scheme emphasizing the intrinsic characteristics of these algorithms. The classification scheme is based on the search strategy, the nature of the input, and the completeness of the output of these algorithms. A short discussion on a few more recent algorithms is also included. An experimental evaluation that explores the relevance and applicability of a subset of these algorithms for some current application scenarios is furnished for completeness.

Keywords: Algorithms, graph mining, frequent subgraphs, subgraph discovery.

GRAPH-BASED representations of real-world problems have been helpful due to their improved clarity and efficient use in finding the solutions[1–4]. This has opened up a new domain in the field of data mining and knowledge discovery, i.e. graph mining. Graph mining has been one of the most explored and evolving research topics in the data-mining domain[4–6]. Some of the important problems addressed by graph mining are pattern matching, clustering and subgraph discovery, to name a few.

Subgraph discovery is one of the well-addressed problems in the graph-mining domain. It is further divided into various sub-tasks depending on the features of the subgraphs to be discovered. The two main sub-tasks are the frequent subgraph discovery[7] and dense subgraph discovery[8–10]. Among these, frequent subgraph discovery has been more popular and happens to be one of the most explored problems today. Frequent subgraph discovery algorithms have been extensively used in various application domains such as computational biology[11], which includes discovering motif structure for protein interactions[12], finding residue packing patterns from protein structures[13] and in chemical carcinogenesis[14]. These algorithms are found to be useful in security threat assessment and analysis as well[15,16].

*For correspondence. (e-mail: varunkrishna007@gmail.com)

The frequent subgraph discovery problem can be defined as the process of finding those subgraphs from a given graph or a set of graphs which have frequent or multiple instances within the given graph or the set of graphs. Among the algorithms that have been developed in this area, SUBDUE[17] is one of the early examples. Agrawal *et al.*[18] have done pioneering work that laid the basis for these algorithms that were developed later in this field. The complexity of the frequent subgraph discovery problem is evident from the diversity of approaches employed in developing various algorithms for its solution.

The frequent subgraph discovery problem has been addressed from many directions using various approaches, including a priori strategy[18], inductive logic programming (ILP)[19,20] and pattern growth approach[21]. 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. It is with the intention to assist in this task that we propose to establish a common framework for analysing various properties of these algorithms. Such a framework is intended to help application developers in making an informed choice that suits the application environment and its requirements. The following factors that connect the solutions and their algorithms to the application requirements help in constructing this framework.

- 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.
- Nature of the input – The algorithms are of two types based on the input they take. The first type takes in a single large graph as input, whereas the second type takes a set of small graphs as input.
- 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.

As part of our framework, we propose to bring out a classification scheme on the basis of the above three factors.

Efforts have been made in the past that address some of the factors brought out above. A summary of the literature presented by Washio and Motoda[7] describes various algorithmic approaches for frequent subgraph discovery. However, they do not touch upon the other significant factors such as input and output types. Similarly, a more recent work by Aggarwal and Wang[5] has mentioned some algorithms for frequent subgraph discovery. However, the focus of their work is on graph mining in general, covering other sub-problems in this domain. Worlein *et al.*[22] have presented a study on three algorithms based on DFS strategy. However, they have not mentioned about many other algorithms in their study. This has led us to conclude that no formal method of classifying the frequent subgraph mining algorithms can be found in any of these efforts.

The study presented here is relevant in this context as it focuses on classifying the algorithms based on three different factors and explains each one of them in detail. The first factor is based on the search strategy used; the second is based on the nature of the input and the third on the completeness of the output. The criteria for choosing these three factors for classifying frequent subgraph discovery algorithms are based on their role in determining applicability of these algorithms for different application scenarios. A quick overview of the organization of this article is as follows.

We first focus on various existing algorithms for frequent subgraph discovery. Then we establish a scheme for classification of these algorithms based on the search strategy, nature of the input, and completeness of the output. This is followed by a discussion on some latest algorithms for frequent subgraph mining. Next an experimental evaluation of these algorithms on some benchmark datasets is presented. We conclude with some directions for future research in the domain of frequent subgraph mining.

Highlights of existing algorithms

The problem of frequent subgraph discovery has its roots in the early nineties with the formulation of the algorithm for the market basket-analysis[18]. This work was followed by the development of many new concepts and algorithms, and now we have numerous algorithms for frequent subgraph discovery exploiting various algorithmic approaches and other factors.

The frequent subgraph discovery process can be divided into two major steps. The first step is to find a subgraph based on a search strategy. The next step is to find matching ones in the given graph/graphs using subgraph isomorphism. In general, the following four aspects influence the execution of these algorithms and also the output generated by them.

- Graph representation;
- Subgraph generation;
- Algorithmic approach;
- Frequency evaluation.

We now review a few popular algorithms that have been used in various applications and have also laid the path for considerable enhancements later.

Graph representation is an important aspect in frequent subgraph discovery algorithms because it has direct and significant influence on memory usage as well as execution time of these algorithms. Various graph representation schemes are available, among which adjacency matrix, adjacency list, hash table and trie are frequently used by the mining algorithms. Adjacency matrix representation is easier to implement, but there can be a considerable waste of memory if the input graph/graphs is sparse. The adjacency list representation consumes less space compared to adjacency matrix. The hash table scheme uses a hash function to map keys with their corresponding values. Its advantage over other representations is the speed with which it retrieves the result. It is useful in the case of very large input graphs. Trie is an ordered tree data structure which can be used to represent graphs. Look-up time in trie is less and is comparatively faster than a poorly designed hash table. Out of the 10 algorithms that we consider here, SUBDUE, HSIGRAM[23] and FFSM[24] use adjacency matrix representation. FSG[25], gSpan[26] and CloseGraph[27] prefer adjacency lists for storing the graphs. Hash table representation is used by Gaston[28] and this accounts for its superior performance over the other algorithms. FARMER[29] uses trie for graph representation. ISG[30] represents graphs in an entirely different manner. It transforms the input set of graphs into item sets which are then represented using edge triplets. In the case of GREW[31], any sparse graph representation is acceptable.

Subgraph generation is the next important aspect in the frequent subgraph discovery process. It can be realized using a number of mechanisms, out of which the most common ones are level-wise search, extension and merging. 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. SUBDUE and FARMER follow this mechanism for subgraph generation. The second mechanism is extension, where an already discovered subgraph is extended by one edge/node at a time. FSG employs this method, whereas gSpan and CloseGraph use a modified version of extension which is called rightmost extension. In rightmost extension, the basic methodology is to extend the graph from the rightmost vertex or from the vertices from rightmost path of the graph. ISG uses a similar approach known as edge triplet extension in which a discovered itemset is extended by adding one edge triplet in each iteration. The third common mechanism is merging in which certain subgraphs, discovered in previous iteration and connected by one or more edges, are merged to obtain a new subgraph. GREW and HSIGRAM use iterative merging for subgraph generation. FFSM uses a combination of merging and extension for subgraph generation. Gaston uses a combination of cycle close refinement and Nauty-based normalization[32]. Cycle close refinement ensures that graphs are generated only from their corresponding spanning trees.

Algorithmic approach is the core element of the frequent subgraph discovery process. There are two types of algorithmic approaches: a priori-based and pattern growth-based. In the a priori-based approach, a subgraph of size $(k + 1)$ is generated from subgraphs of size k using level-wise strategy and subsequent join methods. FSG, HSIGRAM and FFSM are clearly a priori-based methods. 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. ISG carries out frequent subgraph discovery by transforming graphs into itemsets followed by frequent itemset discovery, which is also a priori-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. SUBDUE, gSpan, CloseGraph, GREW and Gaston follow pattern growth approach. Gaston is the fastest among the lot due to the fact that it uses quick-start principle, where paths in a graph are considered first, which are then enumerated to trees and finally trees are enumerated to find the subgraphs.

The other significant aspect concerned with frequent subgraph discovery is frequency evaluation. It is the process of counting the number of occurrences of a subgraph in a large graph/set of graphs, and determining whether it is frequent or not. A subgraph is frequent if its count is greater than a predefined threshold value. Frequency evaluation can be carried out using many techniques. SUBDUE uses minimum description length code and background knowledge to compute the frequency; in fact, this computation also results in finding the best subgraph based on the value obtained. FARMER uses the trie data structure for frequency computation. FSG and 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. CloseGraph and gSpan use DFS lexicographic ordering for frequency evaluation. Here, each graph is mapped into a DFS sequence followed by construction of a lexicographic order among them based on these sequences, and thus a search tree is developed[26]. Now the minimum DFS code obtained from this tree for a particular graph is the canonical label of that graph which helps in evaluating the frequency. In case of HSIGRAM and GREW, the aim is to find the maximal independent set of a graph which is constructed out of the embeddings of a frequent subgraph so as to evaluate its frequency. FFSM uses a sub-optimal canonical adjacency matrix tree for counting the frequency. 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.

In summary, this section talks about 10 popular algorithms for frequent subgraph discovery. The details of these algorithms with respect to the four algorithmic aspects are furnished in Table 1 for a quick reference.

Classification scheme based on three factors

From the very description of each of the above algorithms, a clear distinction can be drawn among them based on various algorithmic aspects. Hence, the emphasis is on classifying them based on some essential factors, viz. the search criteria, nature of the input and completeness of the output. The significance of this classification scheme is that it enables the developers to choose an appropriate algorithm to be used for a particular application based on the resource constraints and other requirements of the application or the environment in which it is going to be used.

Based on search strategy

We can classify various frequent subgraph discovery algorithms on the basis of the search strategy used to mine the graphs. The search strategy is of two types, BFS and DFS. The advantage with BFS approach is that it guarantees discovery of all frequent subgraphs above a minimum threshold given relaxed memory and time constraints. The reason is that BFS being a level by level search, considers all nodes at a particular level and thus searching in this manner covers all the nodes in the graph. Another advantage is its low vulnerability to redundancy compared to DFS. But it is observed that most of the recent algorithms have focussed on depth first approach since it consumes less space compared to breadth first approach. This is because the number of lists that need to be stored in memory in case of the depth first approach is proportional to the depth of the graph (in case of a set of graphs, it is equal to the depth of the biggest graph), whereas in breadth first approach, it is proportional to the width of the graph (i.e. the maximal number of subgraphs in one level)[22]. The disadvantage with DFS is that the probability of redundancy in subgraphs generated is high.

First, let us consider the algorithms that use the breadth first strategy as shown in Figure 1. SUBDUE is one of the early algorithms that has implemented the breadth first approach. FARMER (initial version) also falls in the same category. Other algorithms that fall under this category include FSG and HSIGRAM. The main drawback

Table 1. Algorithmic aspects of ten popular algorithms

Algorithm	Graph representation	Subgraph generation	Algorithmic approach	Frequency evaluation
SUBDUE	Adjacency matrix	Level-wise search	Pattern growth	Minimum description length code and background knowledge
FARMER	Trie structure	Level-wise search	ILP and a priori-based	Trie data structure
FSG	Adjacency list	One edge extension	A priori-based	Transaction identifier (TID) lists
gSpan	Adjacency list	Rightmost extension	Pattern growth	Depth first search (DFS) lexicographic order
CloseGraph	Adjacency list	Rightmost extension	Pattern growth	DFS lexicographic order
HSIGRAM	Adjacency matrix	Iterative merging	A priori-based	Maximal independent set
GREW	Sparse graph representation	Iterative merging	Pattern growth	Maximal independent set
FFSM	Adjacency matrix	Merging and extension	A priori-based	Sub-optimal canonical adjacency matrix tree
Gaston	Hash table	Extension	Pattern growth	Embedding lists
ISG	Edge triplet	Edge triplet extension	A priori-based	TID lists

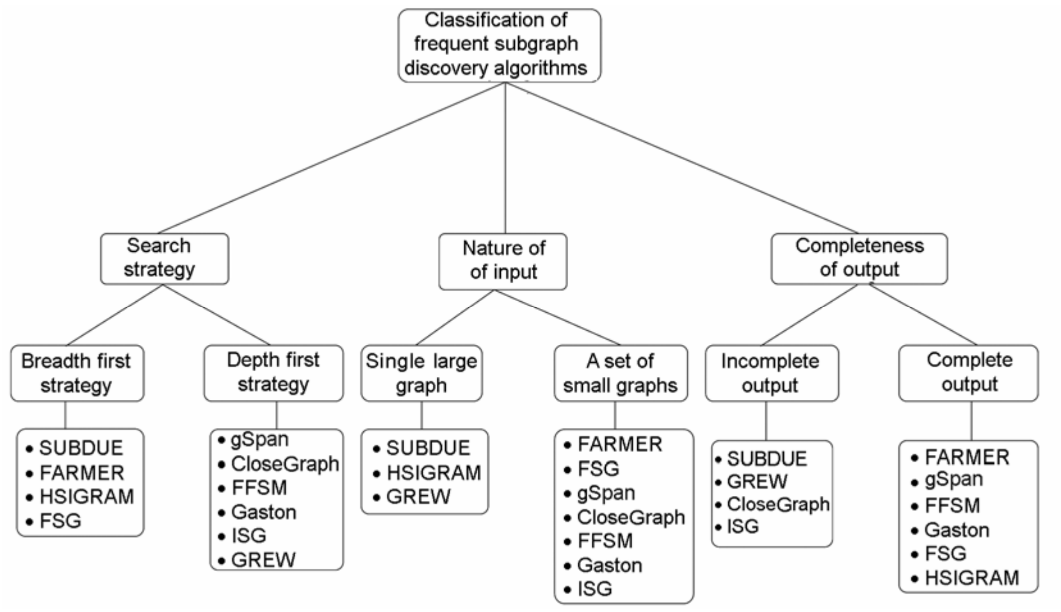
Classification of frequent subgraph discovery algorithms
Search strategy
Nature of input
Completeness of output
Breadth first strategy
Depth first strategy
Single large graph
A set of small graphs
Incomplete output
Complete output
• SUBDUE • FARMER • HSIGRAM • FSG
• gSpan • CloseGraph • FFSM • Gaston • ISG • GREW
• SUBDUE • HSIGRAM • GREW
• FARMER • FSG • gSpan • CloseGraph • FFSM • Gaston • ISG
• SUBDUE • GREW • CloseGraph • ISG
• FARMER • gSpan • FFSM • Gaston • FSG • HSIGRAM

Figure 1. Classification of frequent subgraph discovery algorithms.

with all these algorithms is that they are slower than their counterparts that use the DFS strategy. But it should be noted that these algorithms have been popular among the frequent subgraph discovery community.

gSpan is the first algorithm implemented using the DFS strategy. CloseGraph and FFSM also follow the same search strategy. Gaston is the most efficient algorithm among the lot and its efficiency is mainly due to its DFS strategy. Some of the recent algorithms have also resorted to DFS because of its advantages, and ISG is one among them.

Based on nature of input

Another classification of the algorithms is based on the input they take (Figure 1). This is due to the fact that we can consider two scenarios when dealing with frequent subgraph mining. In the first case, we can consider a set of graphs or a graph database from which we can mine frequent subgraphs. In this case, even when there are a large number of graphs, each individual graph would be small in size. In the second case, we have a large input graph and we have to mine out frequent subgraphs which repeat at various sections of this single graph. Here the size of the input graph can be very large and in turn mining patterns can be more tedious than in the first case. There are a wide variety of applications for both these problem types. In case of a telephone call graph or a social network, the graph representation would have a single large graph from which frequent subgraphs have to be mined. The applications of a set of graphs or multiple graphs occur in the case of molecular databases, medical datasets, and chemical compound structures, to name a few.

Among the algorithms that fall under the single input graph category, chronologically SUBDUE is the first algorithm. HSIGRAM is another algorithm that takes in a single large sparse graph as input. GREW also falls in the same category. It is to be noted that few algorithms are available in this category as the application domains dealing with graph databases outnumber the applications dealing with a single graph.

The algorithms that fall under the second class, i.e. those which take a set of graphs as input, form a bigger list. Out of the ten algorithms we have considered, FARMER is the earliest one developed for graph databases. FSG is another algorithm where a set of graphs is considered as input. gSpan, which works using the DFS strategy, also falls under this class. CloseGraph is more efficient than gSpan and takes in a set of graphs as input. FFSM and Gaston are other DFS-based algorithms which take multiple graphs as input. ISG also mines maximal frequent subgraphs from a set of graphs. Most of the recently developed algorithms come under this class of graph set input type.

Based on completeness of output

The algorithms can be classified into two groups based on the completeness of the search they carry out and the output they generate (Figure 1). The algorithms in the first group do not mine the complete set of frequent subgraphs, whereas those in the second group discover the whole set of frequent subgraphs. There is a trade-off between performance and completeness. In the case of huge inputs, it would be computationally expensive to mine all the frequent subgraphs. Also, the requirement of the problem might be limited to a certain number of frequent subgraphs rather than the complete set. But in certain domains, it may be required to find all frequent subgraphs. In such a case, we have to resort to algorithms mining out complete sets of frequent subgraphs.

SUBDUE is one of the early algorithms that produced an incomplete set of frequent subgraphs. It employs a greedy search approach and there is no backtracking involved, and hence it cannot mine all the frequent substructures. Later GREW was developed, which is also heuristic in nature, but more efficient than SUBDUE. It also scales well when applied on a large input graph. Due to its heuristic nature, GREW cannot mine many frequent subgraphs that are usually discovered by complete frequent subgraph mining algorithms. CloseGraph is another algorithm which does not mine out a complete set of frequent subgraphs. The maximal frequent subgraph discovery algorithm, ISG, also mines an incomplete set of frequent subgraphs.

Several algorithms mine the complete set of frequent subgraphs according to the given criterion or conditions. FARMER, which follows the ILP approach, has been one of the earliest of the lot which can mine a complete set of subgraphs. But it is inefficient in terms of computation. FSG, which follows the a priori-based approach, mines a complete set of frequent subgraphs. But its utility in the real-world scenario is limited due to efficiency constraints. gSpan appears to be more efficient than FARMER and FSG. It also mines the complete set of frequent subgraphs. FFSM is another algorithm which can efficiently mine out all the frequent subgraphs. It adopts a DFS search strategy and mines all the connected subgraphs. Gaston, since it applies the quick start principle, is the most efficient in the lot and finds the complete set of frequent subgraphs above a given threshold. HSIGRAM is not as efficient as Gaston, but mines all the frequent subgraphs from a single large graph.

Some current trends

In the previous section, we focussed on some algorithms that have been used or are still in use across various application domains for frequent subgraph discovery. In this section, we attempt to cover some current trends in this area and the algorithms developed using these trends. These algorithms are also steadily establishing their place in various applications.

Random search strategy

In some applications, the conventional BFS and DFS strategies may fail to find all the subgraphs for a large graph dataset within a given time-frame. Another issue with these strategies is that they might be unsuccessful in mining some important subgraphs. Such situations can be handled with the use of a heuristic random search strategy, where subgraph generation is employed by extending a random edge or a vertex. An algorithm implementing random search strategy is ORIGAMI[33], which finds maximal orthogonal subgraphs instead of all frequent subgraphs.

Principal component analysis

Principal component analysis (PCA)[34] is a technique involving the mathematical transformation of a set of connected variables to a small number of independent variables. The recent research trends show that PCA can be applied in finding frequent subgraphs. gPCA algorithm[35] uses Lanczos algorithm[36] to implement the PCA technique. In the gPCA algorithm, the approach is not to find all frequent subgraphs, but to mine out informative patterns from the graphs corresponding to the significant principal components.

Application-specific

Frequent subgraph discovery algorithms are usually generic in nature. These algorithms can be applied across various domains by tweaking the core algorithm to the application requirements. But now, there seems to be a

Table 2. Description of molecular datasets used

Dataset	Graph	Distinct edge labels	Distinct node labels	Number of average edges in a graph	Number of average nodes in a graph	Number of maximum edges in a graph	Number of maximum nodes in a graph
Compound_422	422	2	21	42	40	196	189
Tumour	42,247	3	67	29	27	229	223
HIV	42,646	3	63	47	45	252	250

change in this approach. There has been a rise in the development of application-centric algorithms which are most suited for certain applications and efficient in solving the problem associated with that domain. Grou Miner[37] is one such algorithm, which is applicable to object-oriented programming. It is used for discovering usage patterns of one or more objects or classes, wherein objects or classes are represented in the form of labelled directed acyclic graphs. The whole problem domain is transformed into graph representation and then Grou miner is applied. Grou Miner has wide applications in software development, as it can help discover useful code skeletons and also anomalies.

Frequent directed labelled subgraph discovery

The trends discussed above are mainly concerned with undirected frequent subgraph discovery. An emerging trend is the directed labelled subgraph discovery. DIGDAG[38] is an algorithm aimed at discovering directed acyclic subgraphs. mSpan[39] is another algorithm which mines all the frequent labelled directed subgraphs.

Experimental evaluation

We consider three algorithms, FSG, gSpan and Gaston, for comparison and evaluate them on the basis of their performance on three benchmark datasets. The choice of these three algorithms is based on the fact that they are complete in terms of the output they deliver and are designed to take a set of graphs as input. Along with this, each of the above algorithms selected has a unique feature which distinguishes them from other algorithms. FSG is one of the pioneer algorithms that works using the BFS strategy. gSpan was the first algorithm developed using DFS strategy. Gaston is the latest and the fastest algorithm for frequent subgraph discovery. Hence these three algorithms have been considered in this experimental evaluation.

Three molecular datasets corresponding to computational biology and chemical informatics domain are considered in this experimentation. The DTP human tumour cell line screen (CANSO3SD)[40] dataset consists of 42,247 molecules. Each molecule corresponds to a graph, where atoms are represented using nodes and the bonds between them are represented by edges. The HIV dataset[41] has 42,687 molecules, each one of which corresponds to a graph. The third dataset, Compound_422 (ref. 27), is comparatively small in size, but the average number of edges and the average number of nodes per graph are considerably larger to that of the tumour dataset. More details about these three datasets are furnished in Table 2.

Experimental set-up

All our experiments have been carried out on a 32 bit Linux system with 4 GB memory and 3.0 GHz Intel processor. The executable program for the FSG algorithm was obtained from the homepage of Karypis[25]. Similarly, the executables for gSpan and Gaston were obtained from Yan and Han[26], and Nijssen and Kok[28]. The benchmark datasets considered for this experimentation had to be transformed to a format that is acceptable by these executable programs. Forty-one graphs from the HIV dataset have been dropped since the executable of gSpan permitted only a maximum number of 254 nodes per graph in a dataset.

Performance evaluation

We had earlier discussed about a few popular algorithms for frequent subgraph discovery. From this discussion, we can clearly draw a distinction among these algorithms in terms of their performance. Since we have a mix of algorithms which act on a single graph as input as well as a set of graphs as input, we can construct two orderings for these two classes of algorithms based on their performance. The first one is for the single-graph input class. Here SUBDUE is the slowest. HSIGRAM, which uses BFS strategy, is faster than SUBDUE[23]. The fastest of the lot in this class is GREW[31]. The set of frequent subgraph discovery algorithms for a single large graph input case is arranged in increasing order of efficiency as given below:

{SUBDUE, HSIGRAM, GREW}.

The second ordering is for the multiple graph input class. Among the algorithms that we have considered, FSG is the slowest in this class since it employs a BFS strategy. gSpan is faster than FSG[26], but CloseGraph using equivalent occurrence and early termination is more efficient than gSpan[27]. FFSM which was developed later turned out to be faster than CloseGraph[24]. Gaston, which is based on the quick start principle, leads the list as it is the fastest algorithm for frequent subgraph discovery. The set of frequent subgraph discovery algorithms for the multiple graphs input case is arranged in increasing order of efficiency as given below:

{FSG, gSpan, CloseGraph, FFSM, Gaston}.

The performance superiority of Gaston over gSpan was brought out by some authors[22,28], but so far no attempt has been made to bring FSG into perspective. The importance of FSG is that it identifies the complete set of frequent subgraphs using the BFS strategy. So here the aim is to bring out clearly the efficiency dominance of DFS-based algorithms over BFS-based ones. Our experiments on each of the three datasets proved that Gaston was above par in performance over FSG. Even though in case of the DTP tumour dataset, gSpan had almost a comparable efficiency with Gaston, in case of HIV assay, the latter completely outperformed gSpan and FSG. Table 3 shows the relative performance of these three algorithms on Compound_422 dataset. The performance evaluation of FSG, gSpan and Gaston on the tumour dataset is presented in Table 4. Similarly, Table 5 shows the performance comparison of these three algorithms on the HIV dataset.

We have also evaluated the algorithms based on the number of frequent subgraphs generated by them (corresponding to different minimum support values). As expected, with a decrease in the minimum support value, the number of frequent subgraphs generated increases, which is plotted in Figure 2. On the basis of the experimental values obtained in terms of the execution time,

Table 3. Experimental results with Compund_422 dataset

Minimum support (%)	Number of subgraphs discovered*	FSG *T* (sec)	gSpan *T* (sec)	Gaston *T* (sec)
60	16	0.1	0.08	0.03
50	29	0.2	0.11	0.04
40	56	0.3	0.12	0.06
30	119	0.4	0.14	0.07
20	932	1.2	0.29	0.21
10	15,966	12.9	2.00	1.75

Table 4. Experimental results with Tumour dataset (CANSO3SD)

Minimum support (%)	Number of subgraphs discovered*	FSG *T* (sec)	gSpan *T* (sec)	Gaston *T* (sec)
60	36	15.8	9.51	3.76
50	49	20.5	10.43	4.64
40	88	30.0	11.82	6.08
30	155	45.2	14.19	8.35
20	363	71.5	19.86	13.81
10	1496	143.7	38.25	33.15

Table 5. Experimental results with HIV dataset (AIDO99SD)

Minimum support (%)	Number of subgraphs discovered*	FSG *T* (sec)	gSpan *T* (sec)	Gaston *T* (sec)
60	125	74.6	23.72	13.02
50	230	119.9	30.97	19.76
40	405	184.0	40.62	29.41
30	827	306.1	62.92	46.99
20	2,144	582.6	118.60	93.15
10	10,970	1672.8	505.60	352.19

*The three algorithms that we considered for experimentation have discovered the same number of subgraphs corresponding to different minimum support values on each of these datasets.

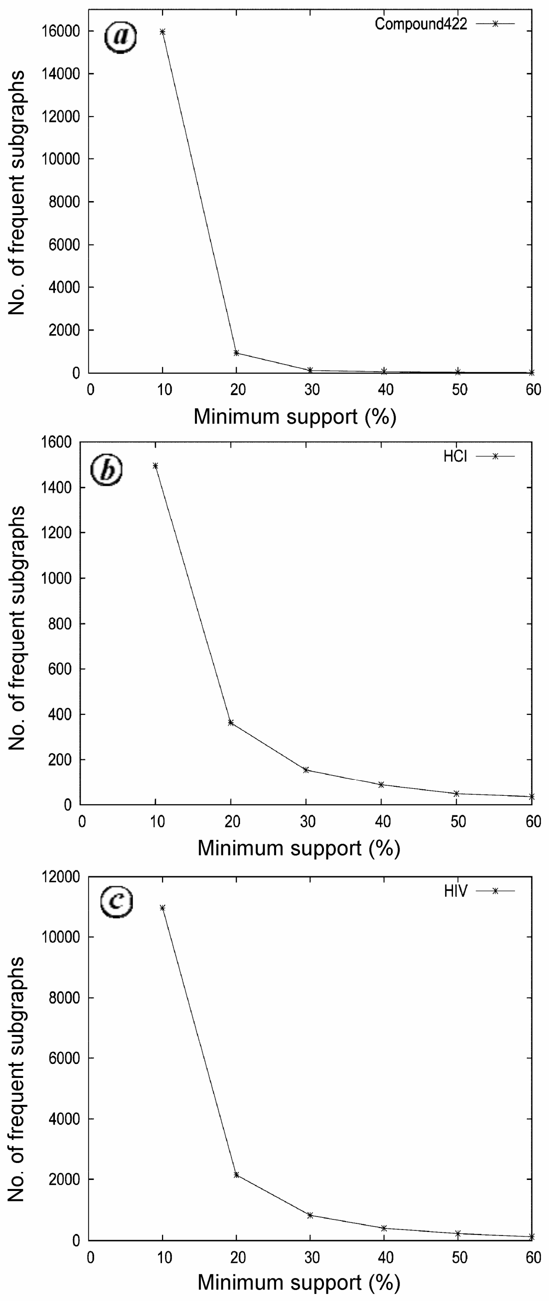

Figure 2. Number of frequent subgraphs present in (***a***) Compound_422, (***b***) Tumour and (***c***) HIV datasets.

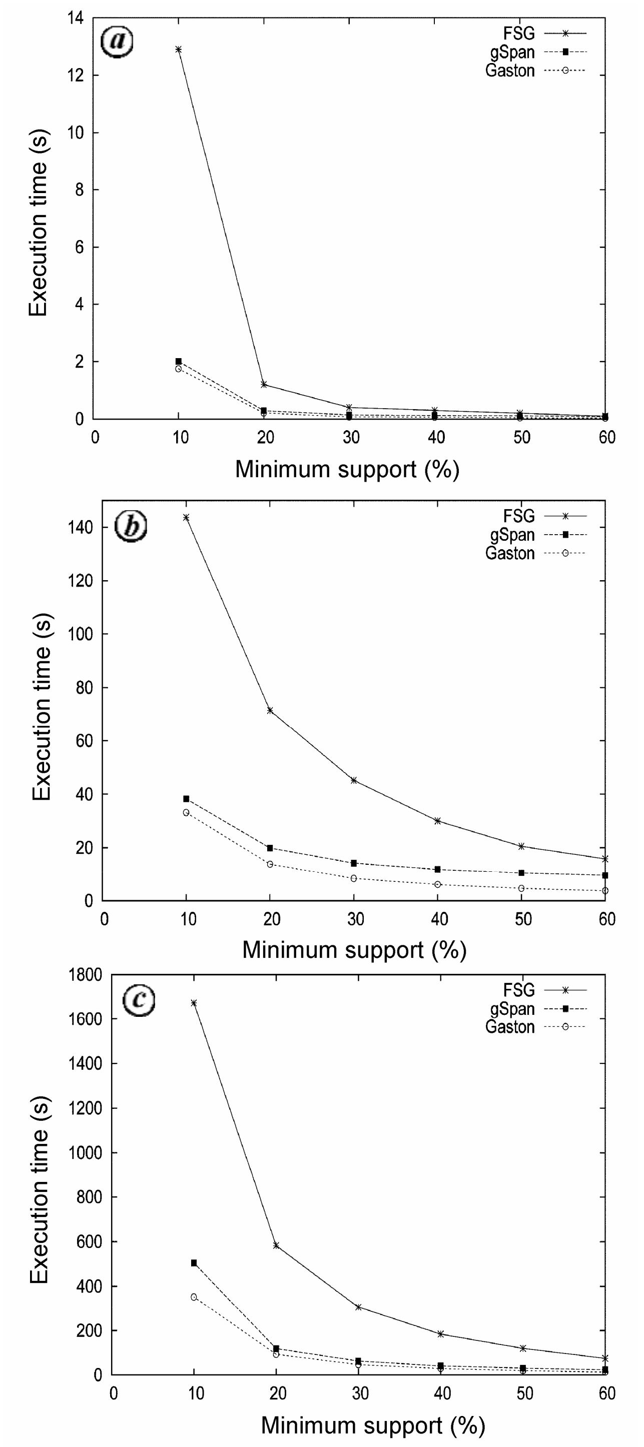

Figure 3. Performance comparison of algorithms on (***a***) Compound_422, (***b***) Tumour and (***c***) HIV datasets.

separate plots for each dataset are produced as shown in Figure 3. A striking observation from these plots is that the execution time decreases drastically with increasing minimum support, as the expected number of frequent subgraphs decreases with increasing minimum support. Another observation is that with decreasing minimum support, the difference in performance between Gaston, gSpan and FSG widens considerably, except in the case of the tumour dataset.

The above experiments reaffirm the fact that Gaston is more efficient than many existing algorithms that discover all frequent subgraphs.

Conclusion

Frequent subgraph discovery is one of the most challenging problems in the graph-mining domain. This article provides a comprehensive description of a few popular and efficient algorithms in this area highlighting various algorithmic aspects. Emphasizing the need to assess the suitability of various frequent subgragh discovery algorithms to different application scenarios, a framework has been proposed classifying them into various categories based on some essential factors. An experimental evaluation of these algorithms has also been reported here bringing out their relative performance, which may help researchers and developers to choose the algorithms that suit best their applications. Apart from the experimental evaluation, this article also provides a discussion on a few recently developed frequent subgraph discovery algorithms. It is evident that recently, there has been a drastic change in the approach towards graph mining, exemplified by ORIGAMI, which does not use any conventional search methodology.

The future prospects for frequent subgraph discovery are manifold in terms of the enhancements possible.

- The concept of discovering the entire set of frequent subgraphs is getting replaced by mining only the significant patterns, which is a subset of the complete set of frequent subgraphs. This can save storage space and execution time, thereby making the algorithm more efficient.
- Another direction is to make use of the inherent parallelism in the algorithm that can help exploit cluster or cloud computing.

With the enhancements in processing power and storage space, more efficient algorithms are expected to be developed which can energise the whole research area of frequent subgraph discovery further.

1. Hanneman, R. and Riddle, M., Introduction to social network methods; http://faculty.ucr.edu/hanneman/
2. Newman, M., Watts, D. and Strogatz, S., Random graph models of social networks. In Proceedings of the Self-Organized Complexity in the Physical, Biological, and Social Sciences, National Academy of Sciences, USA, 2002, vol. 99, pp. 2566–2572.
3. Bharat, K., Chang, B., Henzinger, M. and Ruhl, M., Who links to whom: mining linkage between web sites. In Proceedings of the 2001 IEEE International Conference on Data Mining, California, USA, 2001.
4. Li, Z., Lei, J., Wang, L. and Li, D., A data mining approach to generating network attack graph for intrusion prediction. In Pro-

ceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery, 2007, vol. 4.
5. Aggarwal, C. and Wang, H., *Managing and Mining Graph Data*, Springer Publishing Company, Incorporated, 2010, 1st edn.
6. Tan, P., Steinbach, M. and Kumar, V., *Introduction to Data Mining*, Pearson Education Inc., 2006.
7. Washio, T. and Motoda, H., State of the art of graph-based data mining. *Acm. Sigkdd Explor. Newsl.*, 2003, **5**(1), 59–68.
8. Papadopoulos, A., Lyritsis, A. and Manolopoulos, Y., Skygraph: an algorithm for important subgraph discovery in relational graphs. *Data Min. Knowledge Discovery*, 2008, **17**(1), 57–76.
9. Andersen, R., A local algorithm for finding dense subgraphs. In Proceedings of the Nineteenth Annual ACM–SIAM Symposium on Discrete Algorithms, 2008.
10. Gibson, D., Kumar, R. and Tomkins, A., Discovering large dense subgraphs in massive graphs. In Proceedings of the 31st International Conference on Very Large Databases, 2005, vol. 2, pp. 721–732.
11. Aluru, S. (ed.), *Handbook of Computational Molecular Biology*, Chapman and Hall/CRC, 2006.
12. Ciriello, G. and Guerra, C., A review on models and algorithms for motif discovery in protein-protein interaction networks. *Brief. Funct. Genomics*, 2008, **7**(2), 147–156.
13. Huan, J., Wang, W., Bandyopadhyay, D., Snoeyink, J., Prins, J. and Tropsha, A., Mining protein family specific residue packing patterns from protein structure graphs. In Proceedings of the Eighth Annual International Conference on Research in Computational Molecular Biology, ACM, New York, 2005, pp. 308–315.
14. Dehaspe, L. and Toivonen, H., Discovery of frequent datalog patterns. *Data Min. Knowledge Discovery*, 1999, **3**(1), 7–36.
15. Corley, C., Cook, D., Holder, L. and Singh, K., Graph-based data mining in epidemia and terrorism data; www.eecs.wsu.edu/holder/pubs.
16. Last, M., Markov, A. and Kandel, A., Multilingual detection of terrorist content on the web. In *Lecture Notes in Computer Science*, Springer, Berlin, 2006.
17. Cook, J. and Holder, L., Substructure discovery using minimum description length and background knowledge. *J. Artif. Intell. Res.*, 1994, **1**, 231–255.
18. Agrawal, R. and Srikant, R., Fast algorithms for mining association rules in large databases. In Proceedings of the 20th International Conference on Very Large Data Bases, San Francisco, CA, USA, 1994, pp. 487–499.
19. Muggleton, S., *Inductive Logic Programming*, Academic Press, 1992.
20. Lavrac, N. and Dzeroski, S., *Inductive Logic Programming: Techniques and Applications*, Ellis Horwood, 1994.
21. Han, J., Pei, J. and Yin, Y., Mining frequent patterns without candidate generation. In Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD '00), ACM, New York, NY, USA.
22. Worlein, M., Meinl, T., Fischer, I. and Philippsen, M., A quantitative comparison of the subgraph miners MoFa, gSpan, FFSMC, and Gaston. *Lecture Notes in Computer Science*, 2005, **3721**, 392–403.
23. Kuramochi, M. and Karypis, G., Finding frequent patterns in a large sparse graph. *Data Min. Knowledge Discovery*, 2005, **11**(3), 243–271.
24. Huan, J., Wang, W. and Prins, J., Efficient mining of frequent subgraphs in the presence of isomorphism. In Proceedings of the Third IEEE International Conference on Data Mining, ICDM '03, IEEE Computer Society, Washington DC, USA, 2003.
25. Kuramochi, M. and Karypis, G., Frequent subgraph discovery. In Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM '01), 2001, pp. 313–320.
26. Yan, X. and Han, J., gSpan: graph-based substructure pattern mining. In Proceedings of the 2002 IEEE International Conference on Data Mining, ICDM '02, IEEE Computer Society, Washington, DC, USA, 2002, p. 721.
27. Yan, X. and Han, J., ClosegGaph: mining closed frequent graph patterns. In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'03, ACM, New York, USA, 2003, pp. 286–295.
28. 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.
29. 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.
30. Thomas, L., Valluri, S. and Karlapalem, K., Isg: Itemset based subgraph mining. Technical Report, IIIT, Hyderabad, December 2009.
31. Kuramochi, M. and Karypis, G., GREW – a scalable frequent subgraph discovery algorithm. In Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04), 2004.
32. McKay, B., *Practical Graph Isomorphism*, Citeseer, 1981.
33. Al Hasan, M., Chaoji, V., Salem, S., Besson, J. and Zaki, M., Origami: mining representative orthogonal graph patterns. In Seventh IEEE International Conference on Data Mining, 2007, IEEE, 2008, pp. 153–162.
34. Scholkopf, B. and Smola, A., *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, 2002.
35. Saigo, H. and Tsuda, K., Iterative subgraph mining for principal component analysis. In Conference on Data Mining, ICDM'08, 2008.
36. Lanczos, C., An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand.*, 1950, **45**, 255–282.
37. Nguyen, T., Nguyen, H., Pham, N., Al-Kofahi, J. and Nguyen, T., Graph-based mining of multiple object usage patterns. In Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, ACM, 2009, pp. 383–392.
38. Termier, A., Tamada, Y., Numata, K., Imoto, S., Washio, T. and Higuchi, T., DIGDAG, a first algorithm to mine closed frequent embedded sub-DAGs. In Proceedings of Mining and Learning with Graphs Workshop (MLG'07), Citeseer, 2007, pp. 41–45.
39. Li, Y., Lin, Q., Zhong, G., Duan, D., Jin, Y. and Bi, W., A directed labeled graph frequent pattern mining algorithm based on minimum code. In Third International Conference on Multimedia and Ubiquitous Engineering, 2009. MUE'09, Centre for Artificial Intelligence and Robotics, Bangalore, IEEE, 2009, pp. 353–359.
40. DTP human tumour cell line screen. http://dtp.nci.nih.gov/cancer/cancer_data.html
41. DTP aids antiviral screen. http://dtp.nci.nih.gov/aids/aids_data.html

ACKNOWLEDGEMENT. We thank the Director, CAIR for supporting this work. We also thank the researchers who have shared the executable codes of their algorithms with us.

Received 5 October 2010; accepted 18 November 2010