Classification and Analysis of Frequent Subgraphs Mining Algorithms

Mohammad Reza Keyvanpour
Department of Computer Engineering, Alzahra University, Tehran, Iran
Email: keyvanpour@alzahra.ac.ir

Fereshteh Azizani
Department of Computer Engineering, Islamic Azad University, Qazvin Branch, Qazvin, Iran
Email: fereshteh.azizani@gmail.com

Abstract— **In recent years, data mining in graphs or graph mining have attracted much attention due to explosive growth in generating graph databases. The graph database is one type of database that consists of either a single large graph or a number of relatively small graphs. Some applications that produce graph database are as follows: Biological networks, semantic web and behavioral modeling. Among all patterns occurring in graph database, mining frequent subgraphs is of great importance. The frequent subgraph is the one that occurs frequently in the graph database. Frequent subgraphs not only are important themselves but also are applicable in other aspects of data analysis and data mining tasks, such as similarity search in graph database, graph clustering, classification, indexing, etc. So far, numerous algorithms have been proposed for mining frequent subgraphs. This study aims to create overall view of the algorithms through the analysis and comparison of their characterizations. To achieve the aim, the existing algorithms are classified based on their graph database and their subgraph generation way. The proposed classification can be effective in choosing applications appropriate algorithms and determination of graph mining new methods in this regard.**

Index Terms—**Graph database, Data mining, Graph mining, Frequent subgraph**

I. Introduction

Presenting data as graph make expressing the existing connection between data as natural. This characteristic of graphs causes the increasing application of them for modeling complex structures such as images [8], chemical components [20], protein structure [29], biological networks [28], social networks [41], web [30] and XML documents [5]. Due to the speed of creating and increasing the number of graphs of modeling of complex structures, it is necessary to use a method to analysis efficiently this extensive amount of data. This makes data mining among graphs or graph mining an important field in data mining.

Among all patterns occurring in graph database, mining frequent subgraphs is of great importance. The frequent subgraph is the one that occurs frequently in the graph databases. Frequent subgraphs not only are important themselves but are applicable in other aspects of data analysis and data mining tasks, such as similarity search in graph database, graph clustering, classification, indexing, etc. In classification and similarity search, using frequent subgraphs as feature leads to exact results and high scalability [7, 38, 39]. Clustering of spaces with high dimension is very challenging. As the exploration of frequent subgraphs in subsets of dimensions are easily possible, we can use frequent subgraphs as a solution in the clustering of subspace and clustering of spaces with high dimension [31]. Efficient search in graph database is an unavoidable issue in many applications including detection of cancer structures. Large amount of data in these databases makes the ordinal research and one-to-one test of the objects impossible and inefficient. By using frequent subgraphs mining strategies via indexing of frequent graphs, search speed can be increased considerably [37].

Mining frequent subgraphs is an iterative process consisting of two main steps [24]. The first step is candidate generation. In this step, the subgraphs that are probably iterative or they are frequent candidates are generated. The next step is counting step. In this step, the frequencies of generated candidates in database are counted. To do this, the interested candidate should be searched in graph database. The approach of this step is different depending upon the type of graph database on which graph mining process is done. If database is just including one single large graph, the number of occurrences of subgraph is counted in it. But if database is consisting of multiple small graphs, the number of graph occurrences is not important in a special graph and it is equal to the number graphs in which that subgraph is existing [17]. The generated algorithms for mining on a single large graph can be applied for a set of graphs but the opposite is not true. In both cases, counting the number of occurrences of candidates requires the investigation of subgraph isomorphism that is an NP-complete issue [10] and it is costly for great databases.

Considering the increasing importance of frequent subgraphs, this paper attempts to create a general view toward frequent subgraphs mining algorithms by introducing these algorithms and presenting their features. As most of the existing researches in this field are focusing on candidate generation step and they attempt to create algorithms that can mine the minimum number of subgraphs at a limited time, the main focus of this paper is on different kinds of candidate generation approaches. Here, the existing algorithms are classified based on their candidate generation method. Applications require the assessment of the existing methods for the selection of suitable algorithms. Although direct and comprehensive assessments of frequent subgraphs mining algorithms are not possible, this paper tries to guide the researchers in the selection of their required algorithm by presenting their obvious features in the form of a table.

The rest of this paper is organized as follows. Section 2 is dedicated to frequent subgraph mining, basic concepts and general definitions of frequent subgraphs mining. Section 3 classifies the different kinds of frequent subgraphs mining algorithms based on their graph database and their subgraph generation way. Section 4 presents analogous assessment of algorithms. Finally the paper is concluded in Section 5.

II. Frequent Subgraph Mining

If D is the entry database, the frequent subgraphs mining aims to mine graphs with more support value in comparison with predetermined threshold [1]. The support of the subgraph G_S is denoted by *sub(G$_s$)* and is given as

$$sup(G_s) = \frac{\text{The number of graphs of } D \text{ included } G_S}{\text{The total number of graphs in } D}. \quad (1)$$

If Gs is subgraph of graph G, then $sup\ (G) \leq sup\ (G_S)$. Indeed, it can be said that the subgraph frequency is inverse with its length. The subgraphs without this feature- by their length increased their frequency is increased- are called irrelevant subgraphs. This feature of support criterion can be used in premature pruning of infrequent subgraphs [13]. Because if a subgraph has less support than threshold (not frequent), the next created subgraphs from it will have less support considering the support criterion feature (it is not frequent) and they can be pruned.

Frequent subgraphs mining is usually starting with frequent nodes and edges. Indeed, in the first step frequent subgraphs are mined by one node or one edge. This is easily possible through counting the frequency of the existing nodes and edges in database deleting the ones with support value less than threshold. In the next stage a new subgraph is generated by adding new nodes and edges to the subgraph resulted in the previous step. As it is possible that the new generated subgraph is not frequent, it is called candidate subgraph or shortly candidate [32]. Then the frequency of the candidate or the generated candidates are evaluated and the ones in which support value is more than threshold or equal with it, are returned as the input of the next step. This process continues till the time of execution of algorithm reaches a predetermined limit or all the frequent subgraphs are mined. In the followings two important concepts are presented in this area.

Subgraph isomorphism- To determine the frequency of candidate, it is necessary to mine subgraphs that are *isomorphic* with the candidate in *D*. Two graph G_1=(V_1, E_1) and G_2=(V_2, E_2) are *isomorphic* if there is a mapping from V_1 to V_2 such that each edge in E_1 is mapped to a single edge in E_2 and vice versa. In labeled graphs, the labels should be added to the mapping. An *automorphism* is an isomorphism mapping where G_1= G_2 [21]. Graph G_1 is *subgraph isomorphic* with G_2, if G_1 is *isomorphic* with a subgraph of G_2. *Isomorphism* is evaluated by two methods of exact and approximate. In exact method, two subgraphs are *isomorphic* if they are topologically identical to each other. But in approximate method, two graphs are *isomorphic* even in case of partial differences [4, 15].

Canonical labeling– The investigation of subgraphs isomorphism has high computational complexity. To remove this problem by the labels of nodes and edges, to each subgraph one unique code is attributed. This code is called *canonical labeling*. Thus, instead of evaluating the similarity of two graphs, it is adequate to review that two graphs have similar canonical labeling or not [9]. Calculation complexity of canonical labeling is at worst exponential. Different algorithms by defining different canonical labeling try to reduce this problem.

III. Classification of Candidate Generation Approaches

Three main challenges of candidate generation process are: (i) redundant or isomorphic candidates, (ii) infrequent candidates and (iii) the candidates that not exist in the database. Frequent subgraphs mining algorithms use different approaches to remove or reduce these challenges. In Fig. 1 different algorithms are classified based on their graph database and their subgraph generation way. In the followings these algorithms are introduced based on the method they use for candidate generation.

A. Inductive Logic Programming

In this approach, subgraphs are displayed instead of labeled graph as first order predicates by inductive logic programming (*ILP*) [25]. For example, in a molecular database, first order predicates such as atomel (C, A_1, c), atomel (C, $A2$, s), bond (C, $A1$, $A2$, BT) respectively mean: carbon atom in chemical component of c is denoted by parameter $A1$, Sulphur atom in chemical component of C is denoted by $A2$ parameter and these two atoms are connected by a bond of BT type. This kind of graph representation is of two advantages. First, without considering the type of connectivity of database, we can easily display any database by first order predicates. Second, two nodes in subgraph can be mapped into one parameter in database while, in other subgraphs mining algorithms this is not possible [3]. Thus, the candidate subgraphs mining issue is changed into candidate first order predicates mining.

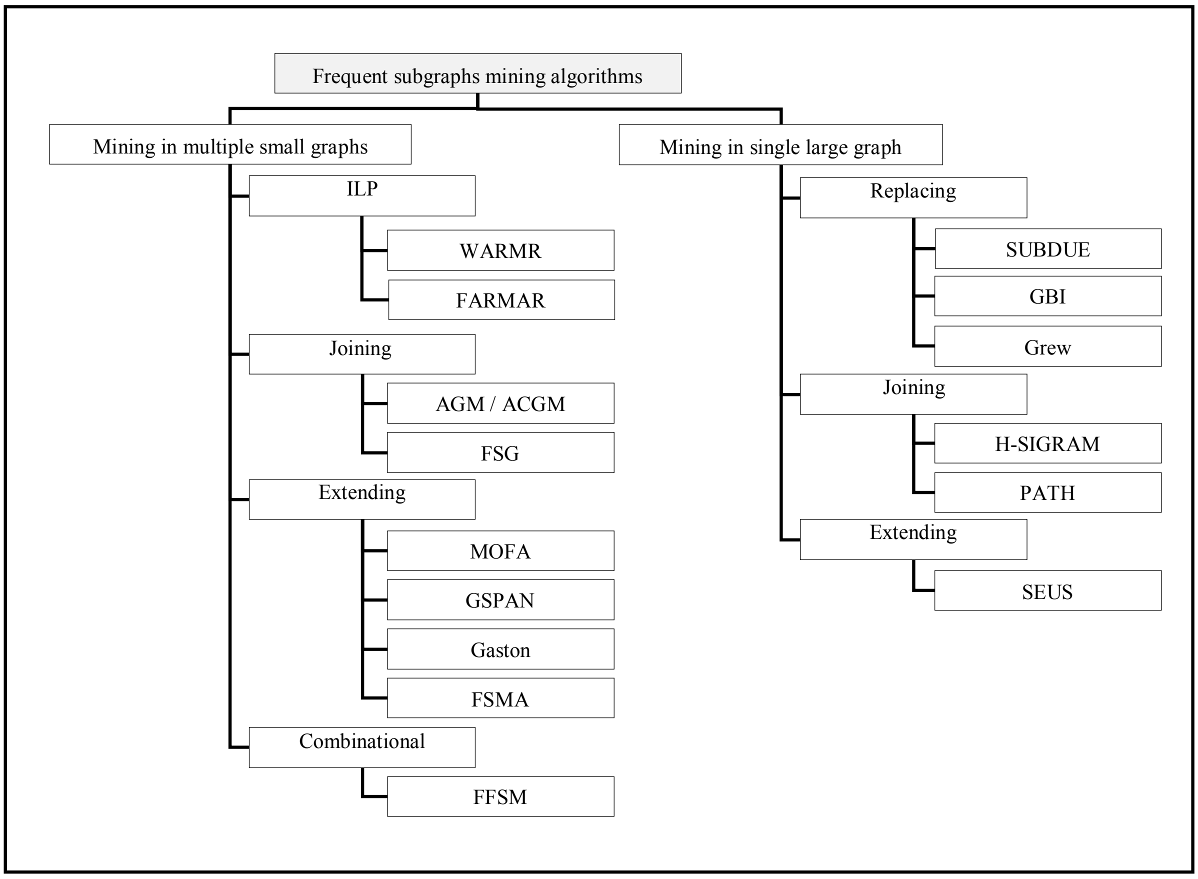

Figure 1. Classification of frequent subgraphs mining algorithms

WARMR [6] is the first system that uses *ILP* strategy for candidate generation. Frequent predicates mining requires equivalence checking. However, as no especial subsumption is attributed for predicates equivalence checking, its calculation complexity is very high. To remove this problem, *FARMAR* [27] uses less strict equivalence criterion. It means that even in case of having partial differences in two predicates, they are considered equivalent. This fact causes that in output there will be multiple equivalent predicates with different forms. The existing challenges in the counting stage of this stagey changes that to unsuitable strategy. The main advantage of this kind of candidate generation method is the high power of displaying detected patterns.

B. Joining

The main idea of candidate generation by join was raised in *Apriori* algorithm [1]. This algorithm is used for mining frequent itemsets and creates frequent itemsets ((k+1) itemsets) by joining two frequent itemsets (k-itemsets). By this method, *Apriori* uses support condition property that was explained in section 2, to delete infrequent k-itemsets. Generalizing this idea and fitting it for graphs mining lead into algorithm Fig. 2 [11]. A lot of algorithms use this approach. The differences of these algorithms are in the type of their building block and in the condition that they use for joining. The building block based on the type of algorithm is node, edge or path.

Candidate generation by this approach includes three kinds of costs. This first cost is related to core identification, second subgraphs joining cost and the third is counting the number of subgraphs frequencies [4]. As in this paper only candidate generation approaches are taken into consideration, only the two first costs reduction strategies are introduced. One of the strategies to reduce these costs is the reduction in the number of candidates. *AGM* [17, 18, 19] by using this idea finds the only *induced* subgraphs of database with definite support. A subgraph $G_S = (V_S, E_s)$ of $G = (V, E)$ is *induced* if E_s contains all the edges of E that connect vertices in V_S.

This algorithm uses node for candidate generation. Indeed, it bonds two frequent candidates with size k, when it has common section with k-1 nodes. Although induced subgraph mining reduces the research space of the candidates, the candidates that are not induced are not distinguished by algorithm. An induced subgraph can be disconnected consisting of separate parts. As in most of the applications connected graphs are considered, limiting research to connected subgraphs doesn't influence the applicability of graph mining. *FSG* [22] algorithm by this idea, only detects connected frequent subgraphs in database. *FSG* uses edge building block and applies different strategies to reduce candidate generation costs. *FSG* stores canonical labels of all (k-1)-subgraphs of a k-graph to reduce the cost of core identification. Therefore, it can calculate the common core between two k-graphs just by computing the intersection between canonical label set of their subgraphs. Joining two subgraphs leads into the generation of multiple graphs. As it is shown in Fig. 3, one of the reasons is the presence of some

```
1. Find all subgraphs composed of a single building block
   and eliminate nonfrequent subgraphs.
2. Find all candidate subgraphs composed of two building
   block and eliminate nonfrequent candidates.
3. In step n:
       a. Generate candidate subgraphs with $n$+1
          building block by merging two subgraphs
          with $n$ building block that have a common
          core with $n$-1 building block.
       b. Eliminate nonfrequent candidates.
       c. Stop when no more frequent subgraphs can
          be generated.
```

Figure 2. Mining frequent subgraphs by joining

automorphism in the core. *FSG* in core identification step, stores the existing automorphisms in each core. So, in join stage, instead of recalculating the automorphisms, the stored automorphisms are used and joining cost is reduced in this way.

FSG for candidate generation joins all the subgraphs with common core. This case creates multiple candidates that makes it inefficient in great database despite all its candidate generation techniques. *HSIGRAM* [21] for the reduction of the number of candidates joins two subgraphs only when there is no common primary subgraph. Two (k-1)-subgraphs of graph G, when among all canonical labels of (k-1)-subgraphs have the smallest canonical labels are called G primary subgraphs. This candidate generation approach reduces the number of irrelevant and redundant subgraphs. Another approach to reduce candidates is increasing the size of building block in the given algorithm in Fig. 2. [11] and [33] by using this idea, use path building block and by less frequencies detect frequent subgraphs (this algorithm is shown in Fig. 1 and Table I as path).

C. Replacing

In this strategy after detecting the frequent subgraph in each stage, the detected subgraph is replaced by a node in the main graph and in the next stage, mining process continues on a new graph obtained from graph

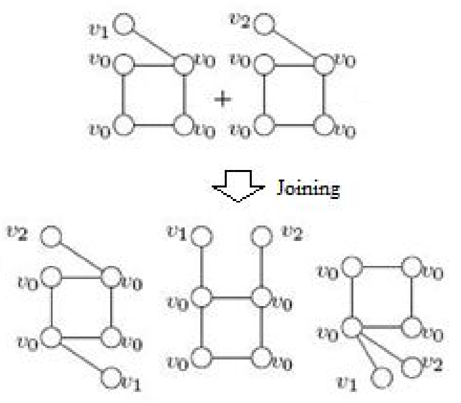

Figure 3. multiple automorphisms of a single core [22].

replacing.Subdue [15] is the first algorithm in this group uses this idea that the higher the number of subgraphs, the more compressed graph is obtained while replacing it with a node. This algorithm is starting by frequent nodes and in each stage extends them by more nodes. To determine the number of candidate's frequencies, total description length is used. Total description length of candidate S in graph G shows compressing power S and is obtained by the following equation:

$$I(S) + I(G|S)\cdot \tag{2}$$

$I(S)$ is the number of the required bits to describe S and $I(G|S)$ is the number of required bits to describe graph G while replacing S with one node. The candidate which minimizes this value, it has high compressing power and is recognized as the frequent subgraph. In this stage the mining process is stopped and the input graph is rewritten by replacing the detected frequent subgraph by a node and the next iteration is started by the new input graph. So, Subdue in each stage, makes input graph more compressed and it is possible to compress the input graph to one node.

GBI [40] has also candidate generation method similar to Subdue. To avoid more compression of the graph (about one node), the size of the explored subgraphs and the compression amount of graph is determined by experimental methods. In this algorithm in each stage the aim is to find the best pair of node that by replacing it, a compressed graph is obtained. The process is shown in Fig. 4. This process continues till the graph size reaches the determined amount. *GBI* despite Subdue stores the existing data in each stage and it can recover the primary graph in each stage of the search.

Subdue algorithm in each stage detects a frequent structure. *GBI* also in each stage, checks one frequent edge. This feature makes them inefficient in dense databases. Because in dense database, the number of input edges are more in comparison with the number of nodes [11] and checking just one candidate in each stage is very unsuitable. In addition, if database includes small subgraph with high frequency , these two algorithms are stopped after detecting it and they don't check the iteration of bigger subgraphs. *GREW* [23] is another algorithm of this group applying some improvements for the problems of two previous algorithms. This algorithm in each moment can search some frequent edges and replace them. To increase the diversity of subgraphs and not just restricted to frequent sub graphs with high frequency, it uses stochastic processes to select pair of nodes.

D. Extending

If we consider search space of subgraphs as a tree consisting of all database subgraphs, as the first level is including no subgraph, the second level is consisting of subgraphs with one edge and K^{th} level is including all the subgraphs with k+1 edge, the joining-based approaches for mining frequent subgraphs in this tree are obliged to use breadth first search. Because these algorithms before mining the (k+1)-subgraphs, mine totally k-subgraphs.

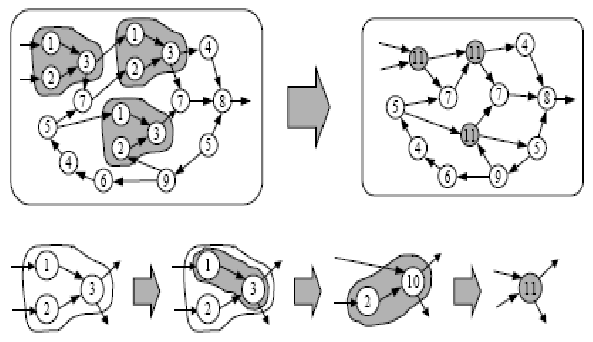

Figure 4. Working method of GBI algorithm [40].

But extending-based approaches are flexible in using search method and besides breadth first search, they can use depth first search. The approaches using depth first search for candidate generation are introduced as without candidate generation appraoches in some of graph mining researches [12, 14]. Because these appraoches instead of generating all k-candidates and checking their iteration, in each stage generate one candidate and check its iteration simultaneously.

In this approach in each stage, frequent subgraph is extended by adding edge in each possible situation of frequent subgraph that was generated in the previous stage. This case creates many redundant candidates. Algorithms of this group use different methods to avoid the creation of redundant candidates. As the existing algorithms in this group are more than the previous groups algorithms [2, 11, 26, 35, 36], just some of them are discussed in this section.

Mofa [2] is the first algorithm of these series created for molecular database but we can use it for the considered graphs. This algorithm uses three kinds of pruning to reduce redundant candidates. These prunings are including: pruning based on size, pruning based on support and structural pruning. In pruning based on size some branches of trees leading into subgraphs, in which the number of edges and nodes are more than predetermined threshold, are pruned. Pruning based on support uses support condition property for pruning. The final and most important kind of pruning in this algorithm is structural pruning that is possible by local numbering. As the existing nodes in subgraph are numbered with the order of their adding to the subgraph. When an edge is added to node *n* in subgraph *Gs*, the next edges only can be added to node *n* or bigger nodes. If we add to node *n* some edges at the same time, the existing edges are ordered based on their label and the label of the existing nodes at their ends are ordered as ascending. Although this kind of numbering method prevents considerably the generation of redundant candidates, the algorithm still generates many redundant candidates and then it uses isomorphism test for redundancy pruning.

gSpan [36] uses a standard label for a graph, called dfs-code. dfs-graph traversal is the order in which nodes are observed. By joining the presence of edges in this order, dfs-code will be the result. Candidate generation in this algorithm is limited into two methods. At first, subgraphs can be extended just in nodes that are in the right path of dfs-tree. Second, *gSpan* stores for each subgraph a list of database graphs including the related subgraph. So, during subgraph extending, instead of mining of the entire database, only the graphs in this list are mined. As these two pruning rules can not totally prevent the generation of redundant candidates, *gSpan* for each subgraph calculates one canonical *dfs*-code and only extends some parts with the minimum canonical code and deletes the remaining.

SEUS [11] for rapid pruning of infrequent candidates uses data structure with the name of *Summary*. This structure is obtained by putting all input graph nodes with similar label in one node and creates a compressive representation of input graph. This data structure is just suitable when the input graph is consisting of a little number of frequent subgraphs with high frequency and it is not suitable in cases where the database are including multiple number of frequent subgraphs with low frequency.

E. Combinational Approaches

The final types of algorithms are algorithms using the ideas of previous approaches that were mentioned before as combinational. Indeed, they are considered as smart approaches that using the advantages of the different candidate generation methods to remove the existing challenges. *FFSM* [16] is the first algorithm of this group that use combination of joining and extending for candidate generation. Joining two subgraphs with common core can leads into the generation of some similar candidates. In addition, a single candidate can be generated by many joining processes. This makes the joining approach inefficient in great databases. In extending approach, extension is only limited to some nodes into which the new edge leads. Thus the algorithm requires much time to check all the nodes. *FFSM* using a different structure for displaying graphs and the combinational method of joining and extending to remove these challenges.

IV. The Evaluation of Frequent Subgraphs Mining Algorithms

Considering the multiple number of the existing algorithms for frequent subgraphs mining, the selection of a good algorithm can be very challenging. As the claims of the proposed algorithms are based on the special database that they have been tested on it, we cannot guarantee their success in other databases and the absolute evaluation of these algorithms are not possible. But there are still some strategies for the evaluation of these algorithms and the selection of appropriate algorithm. These strategies are the following questions:

- What kind of graph is mined?
- Is background knowledge important during mining?
- Is the exact or approximate nature of the result important?
- What is the capacity of the available memory?

- Is it necessary that user during mining manages the mining process?
- How important is losing subgraphs?

Table I.
The obvious properties of frequent subgraphs mining algorithms

Algorithms	Comparison Criteria									
	year	Topology of input graph(s)	Kind of mined subgraphs	Search method	Isomorphism test type	Using of background knowledge	Interaction with user	Search type	Action in dense database	Effective application area
SUBDUE	1994	labeled	connected	greedy	approximate	yes	no	incomplete	bad	Databases without subgraphs with high frequency
GBI	1994	not limited	connected	greedy	exact	no	no	incomplete	bad	Databases without subgraphs with high frequency/ databases with many nodes with same label
Grew	1998	Labeled, undirected	connected	greedy	exact	no	no	incomplete	good	-
WARMR	1998	not limited	connected	breadth first	exact	yes	no	Based on background knowledge	bad	Proof of concept of a framework
FARMAR	1998	not limited	connected	breadth first	approximate	yes	no	Based on background knowledge	bad	Proof of concept of a framework
AGM	2000	not limited	induced	breadth first	exact	no	no	complete	bad	-
FSG	2001	undirected	connected	breadth first	exact	no	no	complete	bad	Databases with variety of node and edge labels
HSIGRAM	2004	Labeled, undirected	connected	breadth first	adjustable	no	no	complete	bad	High-scale sparse graph
PATH	2006	not limited	connected	breadth first	exact	no	no	complete	bad	Data mining on the increasing fraction of online documents
Mofa	2002	Labeled, undirected	connected	depth first	exact	no	no	complete	bad	Molecular databases
gSpan	2002	Labele, undirected	connected	depth first	exact	no	no	complete	bad	-
Gaston	2004	Labele, undirected	connected	depth first	exact	no	no	complete	bad	-
FASM	2008	labeled	connected	breadth first	exact	no	no	complete	good	-
FFSM	2003	Labele, undirected	connected	depth first	exact	no	no	complete	bad	-
SEUS	2006	Labele, undirected	connected	depth first	exact	no	yes	adjustable	bad	-

The above questions are the ones that are suitable in the selection of appropriate algorithm for a specific application. For example, it is better that the applications facing with memory shortage, use candidate generation algorithms with depth first search. Or at times when the speed is more important than exactness, it is better to use approximate isomorphism determination method to increase speed. By answering these questions and the similar questions, we can easily select the best algorithm by the properties of algorithms shown in Table 1. In addition, Table 1 also shows the properties of database in which the related algorithms are undoubtedly having good results.

V.Conclusion

In this paper frequent subgraphs mining algorithms were investigated. At first, these algorithms were classified based on their graph database and their subgraph generation way and then in order to help the applications in the selection of their suitable algorithms, their obvious properties were presented in a form of table. Considering the results, among the existing challenges in this field we can refer to the worst action of most of the algorithms in dense databases, the lack of interaction with the user, the lack of presenting the middle results and time and memory complexities. The presented evaluation is usefull for creating combinational strategies that combine the advantages of different strategies of candidate generation to solve the challenges. Another direction for future research is adding more algorithms and strategies to the above classification The comparison and classification of frequent subgraphs mining algorithms based on the method they use to determine isomorphism is one of the research fields taken a little attention and it can be effective in designing efficient algorithms in this field

References

[1] R. Agrawal and R. Srikant, “Fast Algorithms For Mining Association Rules”, In Proceeding of the 20th Very Large Dada Base . Conference(VLDB'94), Santiago, pp.487-499, 1994.

[2] C. Borgelt, MR. Berthold, “Mining Molecular Fragments: Finding Relevant Substructures of Molecules”, In Proceeding of the international conference on data mining (ICDM’02), Japan, pp. 211–218, 2002.

[3] D. Chakrabarti and C. Faloutsos, “Graph Mining: Laws, Generators, and Algorithms”, ACM Computing Surveys, New York, pp.2-69, 2006.

[4] J. Cook and L. Holder, “Substructure Discovery Using Minimum Description Length and Background Knowledge”, Journal of Artificial Intelligence Research, pp. 231-255, 1994.

[5] E. Damiani , B. Oliboni, E. Quintarelli and L. Tanca, “Modeling Semistructured Data by Using Graph-Based Constraints”, In Proceedings of OTM Workshops, pp.22-23, 2003.

[6] L. Dehaspe and H. Toivonen, ” Discovery of Frequent Datalog Patterns”, Data Mining and Knowledge Discovery, pp.7-36, 1999.

[7] M. Deshpande , M. Kuramochi.M and G. Karypis, “Frequent sub-structure-based approaches for classifying chemical compounds”,In Proceedings of the international conference on data mining (ICDM’03), pp. 35–42, 2003.

[8] AD. Doulamis, ND. Doulamis and ND. Kollias, “A Pyramidal Graph Representation for Efficient Image Content Description”, IEEE International Workshop on Multimedia Signal Processing (MMSP), Denmark, pp.109-114, 1999.

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

[10] M. R. Garey and D. S. Johnson, “Computers and Intractability:A Guide to the Theory of NP-Completeness”, W. H.Freemanand Company, New York, 1979.

[11] E. Gudes, E. Shimony and N. Vanetik, “Discovering Frequent Graph Patterns Using Disjoint Paths”, IEEE Transactions on Knowledge and Data Engineering, Los Angeles, pp.1441–1456, 2006.

[12] J. Han, H. Cheng, D. Xin and X. Yan, ”Frequent Pattern Mining: Current Status and Future Directions”, Data Mining and Knowledge Discovery (DMKD’07), 10th Anniversary Issue, pp.55–86, 2007.

[13] J. Han and M. Kamber, Data Mining: Concepts and Techniques, Second edition:Morgan Kaufmann, 2005.

[14] J. Han, J. Pei and Y. Yin, “Mining Frequent Patterns without Candidate Generation”, In Proceedings of the ACM-SIGMOD International Conference on Management of Data, Texas, pp.1–12, 2000.

[15] LB. Holder, DJ. Cook and S. Djoko, “Substructure Discovery in the Subdue System”, In Proceeding of the AAAI’94 workshop knowledge discovery in databases (KDD’94), WA, pp 169–180, 1994.

[16] J, Huan, W. Wang, J. Prins, “Efficient mining of frequent subgraph in the presence of isomorphism”, In Proceeding of the international conference on data mining (ICDM’03), Melbourne, pp.549–552, 2003.

[17] A. Inokuchi, T. Washio and H. Motoda, “An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data”, In Proceedings of the 4th European Conference on Principles and Practice of Data Mining and Knowledge Discovery(PKDD), pp.13–23, 2000.

[18] A. Inokuchi, T. Washio and H. Motoda ,”Complete Mining of Frequent Patterns From Graphs: Mining graph data”, MachineLearning, pp.321–354, 2003.

[19] A. Inokuchi, T. Washio, Y. Nishimura and H. Motoda, “General Framework For Mining Frequent Patterns in Structures”, In Proceedings of the ICDM workshop on Active Mining (AM), Netherlands, pp.23–30, 2002.

[20] A. Kerber, R. Laue, M. Meringer and C. Rücker, “Molecules in Silico: A Graph Description of Chemical Reactions”, Journal of Chemical Information and Modeling, pp.805-817, 2007.

[21] M. Kuramochi.M and G. Karypis, “Finding Frequent Patterns In a Large Sparse Graph”, in Proceedings of the 4th SIAM International Conference on Data Mining (SDM 2004), USA, 2004.

[22] M. Kuramochi, G. Karypis, “Frequent Subgraph Discovery”, In Proceedings of the international conference on data mining (ICDM’01), California, pp.313–320, 2001.

[23] M. Kuramochi and G. Karypis, “GREW: A Scalable Frequent Subgraph Discovery Algorithm”, In Proceeding of the international conference on data mining (ICDM’04), Brighton, pp.439–442, 2004.

[24] T. Meinl, C. Borgelt.C and Berthold.MR, “Discriminative Closed Fragment Mining and Perfect Extensions in MoFa”, In Proceedings of the 2th Starting AI Researchers' Symposium (STAIRS), pp.3-14, Spain, 2004.

[25] S. Muggleton and L. DeRaedt, “Inductive Logic Programming: Theory and Methods”, Journal of Logic Programming, pp.629-679, 1994.

[26] S. Nijssen, J. Kok, “A Quickstart in Frequent Structure Mining Can Make a Difference”, In Proceeding of the ACM SIGKDD international conference on kowledge discovery in databases (KDD’04), Washington, pp.647–652, 2004.

[27] S. Nijssen.S and J. Kok, “Faster Association Rules For Multiple Relations”, In Proceeding of the 17th International Joint Conference on Artificial Intelligence, pp.891-896, 2001.
[28] JA. Peng, LM. Yang, JX. Wang, Z. Liu and Li. Ming, “An Efficient Algorithm for Detecting Closed Frequent Subgraphs in Biological Networks”, in Proceedings of the International Conference on BioMedical Engineering and Informatics, USA, pp.677-681, 2008.
[29] R. Samudrala and J. Moult, “A graph-theoretic algorithm for comparative modeling of protein structure”, Journal of Molecular Biology, USA, pp.287-302,1998.
[30] A. Schenker, M. Last, H. Bunke and A. Kandel, “Classification of Web Documents Using a Graph Model”, In Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR'03), pp.233-251, 2003.
[31] M. Shahriar Hossain and R. A. Angryk., “GDClust: A Graph-Based Document Clustering Technique”, In Proceedings of the 7th IEEE International Conference on Data Mining, pp.417-422, 2007.
[32] T. Washio and H. Motoda, “State of the Art of Graph-Based Data Mining”, ACM SIGKDD Explorations Newsletter, NewYork , pp. 59–68, 2003.
[33] N. Vanetik, E. Gudes, SE. Shimony, “Computing Frequent Graph Patterns From Semistructured Data”, In Proceedings of the international conference on data mining (ICDM’02), Japan, pp.458–465,2002.
[34] M. Worlein, T. Meinl, I. Fischer and M. Philippsen, “A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM and Gaston”, In Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD’05), Portugal, pp.392–403, 2005.
[35] J. Wu and L. Chen, “Mining Frequent Subgraph by Incidence Matrix Normalization”, Journal of Computers, pp.109–115, 2008.
[36] X. Yan, J. Han, “GSpan: Graph-Based Substructure Pattern Mining”, In Proceeding of the international conference on data mining (ICDM’02), Japan, pp.721–724, 2002.
[37] X. Yan, PS. Yu and J. Han, “Graph Indexing: a Frequent Structure-Based Approach”, In Proceedings of the international conference on management of data (SIGMOD’04), Chicago, pp.335-346, 2004.
[38] X. Yan, PS. Yu and J. Han, “Searching Substructures With Superimposed Distance”, In Proceedings of the International conference on data engineering (ICDE’06), PP.888-989, 2006.
[39] X. Yan , PS. Yu and J. Han, “Substructure Similarity Search in Graph Databases”, In Proceedings of the international conference on management of data (SIGMOD’05), pp.766-777, 2005.
[40] K. Yoshida, H. Motoda and N. Indurkhya, “Graph-based Induction as a Unified Learning Framework”, Journal of Applied Intelligence, pp.297–328.
[41] A. V. Zhdanova, L. Predoiu, T. Pellegrini, D. Fensel, “A Social Networking Model of a Web Community“, In Proceedings of the 10th International Symposium on Social Communication , Cuba, pp. 537-541, 2007.

Mohammad Reza Keyvanpour is an Associate Professor at Alzahra University , Tehran, Iran. He received his B.s in software engineering from Iran University of Science &Technology, Tehran, Iran. He received his M.s and PhD in software engineering from Tarbiat Modares University, Tehran, Iran. His research interests include image retrieval and data mining.

Fereshteh Azizani received her B.s in software engineering from Islamic Azad University, Qazvin Branch, Qazvin, Iran. Currently, she is pursuing M.s in software engineering at Islamic Azad University, Qazvin Branch, Qazvin, Iran. Her research interests include data mining and graph mining.