Indian Journal of Science and Technology, *Vol 9(44), DOI: 10.17485/ijst/2016/v9i44/95055, November 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Significant Subgraph Mining with Representative Set

D. Kavitha[1*], V. Kamakshi Prasad[2] and J. V. R. Murthy[3]

[1]PVPSIT, Vijayawada - 520007, Andhra Pradesh, India; kavitha_donepudi@yahoo.com
[2]Department of Computer Science and Engineering, JNTUH College of Engineering, JNTU Hyderabad - 500085, Telangana, India; kamakshiprasad@jntuh.ac.in
[3]Department of Computer Science and Engineering, JNTU Kakinada, Kakinada – 533003, Andhra Pradesh, India; mjonnalagedda@gmail.com

Abstract

Objectives: To mine significant subgraphs with user specified objective functions from a set of graphs that are useful for understanding the intrinsic characteristics of data in a scalable approach. **Methods/Statistical Analysis:** A large number of candidate subgraphs generated during mining process causes both computational and statistical problem. In this paper, Significant SubGraph Mining-SSGM proposes an algorithm to find significant subgraphs by using a small set of representative patterns - *coreset* that overcomes these problems. Furthermore, an edge graph notation is used to represent a graph that enables to mine patterns directly without using separate mining algorithm. **Findings:** The number of possible candidates is generally exponential in search space and techniques employed are mostly focussed on monotonic property. The proposed algorithm offers simple, yet efficient optimizations to significantly improve performance by pruning the search space and exploring representative graphs. It avoids enumeration of all frequent subgraphs which cause redundancy and extreme mining time. The identified *coreset* elements are extended that provide optimal solution patterns and adopted edge graph notation mine subgraphs directly. **Application/Improvements:** Experimental results shows that the proposed algorithm is effective and efficient for mining significant subgraphs in terms of computational cost, scalability and time over existing methods. The algorithm can be applied to find different types of significant patterns in a scalable manner by using any objective function according to the problem domain including support, correlation measure and feature set.

Keywords: Frequent Graphs, Objective Function, Representative Set, Statistical Significance, Subgraph Mining

1. Introduction

Mining graph data, the extraction of useful knowledge in the form of graphs has more significance as graph is a powerful way to represent structured and complex data[1]. Attributed graphs[2,3] can be used to address various problems such as graphical symbol recognition, shape analysis, protein structure analysis, computer network monitoring, web data analysis, social networks, XML data[4–12] and so on. A useful task to perform on this type of applications is frequent subgraph mining.

Frequent subgraph mining problem can be defined as finding a set of frequent subgraphs from a set of graphs in a graph database or from a large single graph. A subgraph is said to be frequent if it exists in at least T graphs in a data set D, where T is a user defined frequency support. Frequent subgraphs are useful to accomplish basic mining tasks such as description that provides concise and succinct summarization and discrimination, classification, clustering and building indices on graph data. Mining frequent subgraphs has been extensively studied and various efficient graph based algorithms such as AGM, FSG, FFSM, gSpan, Gaston, SPIN[13–19] etc. are developed. A common feature of all these algorithms is that they completely enumerate all the frequent subgraphs. The large number of extracted frequent subgraphs may not be useful to get good classification accuracy or to determine essential characteristics such as disease identification and drug discovery applications. Rather, a small set of significant subgraph patterns are useful to learn important

*Author for correspondence

characteristics and structures hidden in data sets more easily. Finding such significant patterns that appear at low frequency threshold from the database is computational and statistical problem.

1.1 Threshold Bottleneck

Generally in mining process, at first, all frequent subgraphs are mined and are used to select significant patterns based on user defined objective function. Clearly, this two-step traditional process is not scalable to find significant subgraphs due to the reason - low frequency threshold has to be set for many objective functions which generates an exponential search space and slow mining process. Hence, mining frequent patterns with low threshold becomes the bottleneck of the mining process in Figure 1. At high value threshold, a user has to sacrifice the quality and quantity of discovered significant patterns. Generally most significant and discriminate patterns are less in number when compared to frequent patterns; hence, high threshold value suppresses mining of all significant patterns.

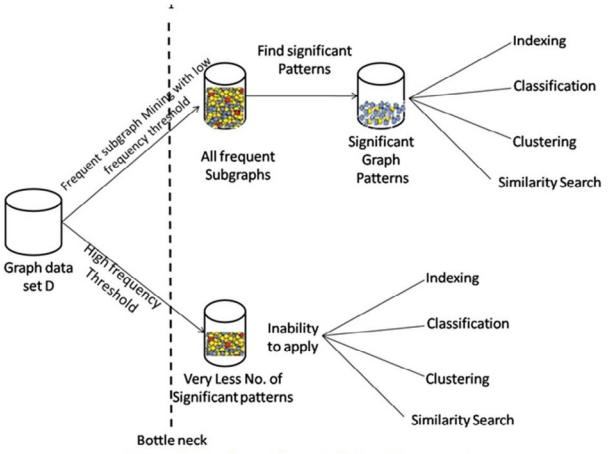

Figure1: Bottleneck to find significant patterns

Here are some existing algorithms that mine significant patterns and find use in applications. An application of boosting for classifying labelled graphs[20], general structures for modeling chemical compounds, natural language texts and bio-sequences was presented. A feature vector[21] representation used to find significant patterns. Unfortunately, all (closed) frequent subgraphs needs to mine first in this practice. The proposed leap search[22] to find patterns, focussed on the databases that can be divided into positive and negative sets. Frequent structures as features[23,24] was used to classify chemical compounds. Frequent graph patterns were also used as indexing features by[25] to perform fast graph search. Pattern-based classification models were demonstrated[26] and in these methods, only distinguishable significant patterns are used, while mining all frequent subgraphs could result in poor performance and low accuracy such as over fitted or under fitted classifiers.[27] discussed mining representative orthogonal graph patterns using α-orthogonal β-representative set to find optimal solution. 21 existing measures to evaluate the significance[28] of the patterns were surveyed. Branch-and-bound search technique was adopted by many of the existing applications with a derived bound for a specific function. Branch and bound search technique may bring poor results as it could fall into local maxima due to very large search space in graph data sets.

In this paper, we propose an algorithm that addresses mining frequent graph patterns which exploit the significant patterns. The proposed algorithm is able to find significant subgraphs in a limited period of time, 1. By using a small set of approximate patterns and 2, By pruning search space. First, a small set of subgraphs referred as *coresets* that play a key role in the extraction of significant graphs are identified. These *coresets* are enough to mine significant frequent patterns which are effectively useful in applications such as graph indexing, classification and clustering[29]. Then, edge subgraphs are created as the representatives of actual graphs. They can be used to mine frequent patterns directly with specified graph size, reducing the cost of candidate generation and incremental growth. Pruning search space is an effective technique to complete the process in time. Size based, support based and structural pruning are three important methods in pruning search space. Pruning search space is itself employed in the proposed process of mining. Adopted *coresets* concept avails the structure based and support based pruning technique. Edge subgraphs have been used to mine frequent patterns directly at specified sizes, thus, size based pruning is implemented.

In summary, our contributions are:

- A novel frame work to mine significant frequent subgraphs in a data set is proposed. It is based on an idea to avoid enumeration of all frequent subgraphs which cause redundancy and extreme mining time.
- It offers a heuristic search with novel optimizations to significantly improve performance by

pruning the search space by adopting *coresets* and exploring approximate graphs.

- To enumerate the possible subgraphs of any size, edge graph notation is adopted to mine subgraphs directly at specified size and it avoids exploration of all frequent subgraphs at small sizes.
- A variation of FSG is proposed to mine all frequent subgraphs than in traditional manner.

2. Preliminary Concepts

Graph: A labelled graph is a quadruple $G = (V, E, L, l)$ where V is a finite set of vertices, E is a set of edges and E $\subset$ $V \times V$, L is a set of labels and $l(V \rightarrow L)$ *is* a function that assigns a unique label to each vertex of graph G.

Subgraph: A *graph* $G' = (V', E')$ is a subgraph of another graph $G = (V, E)$ iff $V' \subseteq V$, and $E' \subseteq E \wedge ((v_1, v_2) \in E' \rightarrow v_1, v_2 \in V')$.

Subgraph Isomorphism: For two labelled graphs G and H, a subgraph isomorphism is a bijection $f: V(G) \rightarrow V(H)$ such that $\forall \; v \in V(G), L(v) = L'(f(v))$ and $\forall \; (u, v) \in E(G) \Leftrightarrow (f(u), f(v)) \in E(H)$ and $L(u, v) = L'(f(u), f(v))$ where L and L' are labels of G and H respectively. This mapping preserves labels on the vertices and edges.

Frequency: Given a graph dataset $D = \{G_1, G_2, .G_n\}$ and a subgraph g, the frequency f of g, is $f = \frac{|Gi|}{|D|}$.

Frequent Subgraph: Given a graph dataset $D = \{G_1, \ldots G_n\}$, a subgraph g and a frequency threshold T, a subgraph g with frequency $f = \frac{|Gi|}{|D|}$ is said to be frequent if and only if $f > T$ w.r.t $|D|$

Significant Subgraph: Given a graph data set $D = \{G_1, G_2, \ldots G_n\}$ and an objective function F, a general problem definition for mining significant graph patterns can be formulated in two different ways: 1. Find all subgraphs g such that $F(g) \geq T$ where T is a significance threshold; or 2. Find a subgraph g^* such that $g^* = \text{argmax}_g F(g)$.

Coreset: A *coreset*[30] is a small subset CS with respect to the given graph dataset D that approximates the original set D, that is finding the *coreset* CS for the given objective function F provides an approximate solution for the problem on the dataset D.

Similar Graphs: Similarity between two graphs is a symmetric binary function F × F → [0, 1]. The similarity based on the maximum common subgraph[31] is given as: $sim(Gi,Gj) = |Gc|/max(|G_i|,| G_j|)$, where Gc is the maximum common subgraph of Gi and Gj.

3. Mining Significant Graph Patterns

3.1 Overview of Significant Graph Pattern Mining

As mentioned earlier, interesting patterns are not simply those that occur frequently, instead they are the patterns that contain some statistical significance based on the defined objective function. Figure 2 shows a sample dataset of amino acids. An amino acid consists of a central carbon atom attached to a carboxyl group (-COOH), an amino group (-NH_2), a hydrogen atom and a side group (-R), giving the general formula R-CH-NH_2-COOH. Amino acids differ with each other based on the atoms composed in the side group. The characteristics and distinguishing of amino acids are determined by side group and they may vary in size, charge, added atom type etc. The carbon atom of each amino acid is attached with four different groups and thus is asymmetric. *Asparagine* and *glutamine* are uncharged but have polar amide groups with extensive hydrogen-bonding capacities. *Arginine* and *lysine* amino acids are positively charged. Based on the traditional frequent subgraph mining technique, the largest frequent subgraphs that can be extracted with support frequency of 50% is shown in Figure 2. It can be observed that these frequent subgraphs have less significant information that might be useful for characterizing the amino acids. Instead, mining subgraphs associated with subset of atoms which have significance in other words which satisfies objective function is required.

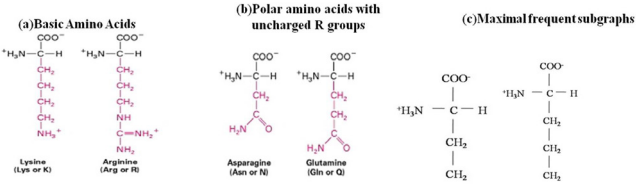

Figure 2: Dataset of Amino Acids

In order to mine significant patterns, the proposed algorithm using a small set of approximate subgraphs (*coreset*) according to their support and significance

from the data set. The elements of *coreset* are used as basic candidates to find optimal patterns. Here, optimization problem is solved directly. We consider any rational objective function such as frequency support, statistical significance, discriminative ratio, structural correlation and correlation measure as a general definition to mine significant patterns. The user can give his perceptions to define the significance. Objective of the proposed concept is to design a general mining framework that can be applicable to a variety of applications. Next we describe the proposed technique to capture the set of approximate subgraphs that generates significant patterns.

3.2 Introduction to *Coreset*

Finding approximate solutions is one of the classical techniques to solve hard problems. "Small" amount of "most significant" information is extracted from the given data and perform the computation on this extracted data to get the solution of the original problem. A *coreset cs* of a set *S* with respect to problem domain is a small subset that represents the original set *S* and approximates the solution result set, that is, identification of the representative set known as *coreset* provides an accurate solution for the original set *S*. *Corsets* were introduced in computational geometry as a small subset of points used to find approximate solutions. In the problem, *coreset* of a data set *D* are a set of small subgraphs that are embedded and approximates the resultant frequent subgraphs, in the sense that *coreset* provides an approximate solution. The advantage of introducing the concept of *coreset* is that an optimal solution can be computed and fasten the computation of discovering significant frequent substructures for the given data set by applying any fast approximation algorithm.

For the given graph data set *D*, *coreset* is the set of frequent subgraphs that can guarantee the generation of all significant frequent subgraphs with minimum support *T*. Minimum support excludes generation of infrequent subgraphs. Anti-monotone property effectively prunes search space. Based on anti-monotone property any supergraph *g'* of an infrequent graph *g* is also infrequent. *Coreset* can be defined as the characteristic representatives of original graph.

Selection of coreset elements is application dependent and may require domain knowledge. To select appropriate substructures as coreset the points to be considered are: 1. The significant patterns are to be statistically frequent, 2. *Coreset* elements are such that these preserve structural information of data set and approximations of result sets and 3. Elements of *coreset* are with minimal overlapping structure.

For the application of amino acids Figure 2, frequent subgraphs of k-size are shown in Figure 3. The subgraphs 3(a), 3(b), 3(c) and 3(d) are with the frequency support 100%, 3(e) is with 75% and 3(f) and 3(g) are with 50% frequency support. In general these frequent subgraphs are used to generate supergraphs and to identify frequent as these subgraphs are frequent. From these subgraphs the *coreset* elements are identified that have statistical significance and representativeness. Along with these another important factor to find optimized *coreset* substructures is structure of the elements. If the search structure is examined horizontally, it can be observed that the subgraphs along the neighbour branches likely to have similar composition and frequencies which differ only in their deterministic features in Figure 3. The structural composition of (a), (b), (c) and (d) varies only by one edge and the frequency support is almost identical. Accordingly, in search space, neighbour branches show strong similarity in pattern composition. The neighbour subgraphs which have structural overlapping can be discarded as the elements of the *coreset* to reduce redundant supergraphs generation. *Coreset* elements are identified in such a way that they represent effective patterns to extend significant patterns, there by pruning the search space using the structural and support pruning methods.

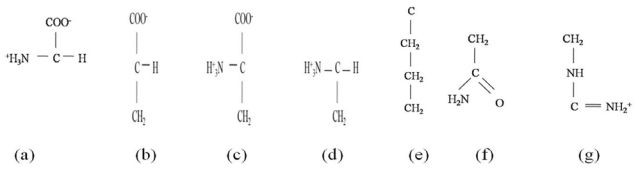

Figure 3: k-size frequent subgraphsof the graph set D shown in figure 2

3.3 Preposition

Let $C_1,..,C_n$ be the elements of *coreset* of data set *D* and $F_1,...F_n$ be the frequent subgraphs of data set *D*, then frequent subgraphs of $C_1...C_n$ be the exactly same as $F_1,...F_n$ frequent graphs in data set *D*.

4. SSGM→ Significant Subgraph Mining Framework

This section starts with the presentation of Significant Subgraph Mining algorithm to mine significant sub-

graphs. Then *findcoreset*() algorithm which identifies optimal subgraphs as elements of *coreset* is explained. Then notation of edge graphs is introduced that is used to represent actual graphs in edge format. Enumeration of significant subgraphs using *SubgraphExtnsn()* algorithm is discussed in detail as perfect extension of a subgraph reduces redundant supergraph generation by not losing significant patterns.

Algorithm: Significant Subgraph Mining (SSGM)

Input: A graph dataset *D*

Output: Significant patterns

Begin

1. $F^1 \leftarrow$ All frequent 1- edge subgraph in *D* in Canonical edge form.
2. $C^K \leftarrow$ *findcoreset* (F^1).
3. for each *G* belongs to *D* do.
4. $S \leftarrow SubgraphExtnsn(n, F^1, C^K)$.
5. *result* ← 0.
6. while $S \neq \emptyset$ do.
7. for each *s* in *S* do.
8. if *s* is not alreadv in *result then.*
9. *result* ← *result* $\cup$ *s*

End

Step 1: Initially, all frequent 1-edge graphs of the data set *D* are enumerated. A 1-edge graph is said to be frequent if its frequency support is greater than the threshold defined for data set *D*. i.e. its occurrence should be at least in *T* number of graphs in data set *D*. Based on anti-monotone property, only these edges participate in frequent subgraphs. (step 1). Then algorithm *findcoreset* (F^1) is executed to identify the elements of *coreset* that are further used to find significant graphs. *findcoreset*(F^1) algorithm will be elaborated later in the next portions of the work. Discovered *coreset* is used as input to generate significant graphs by using *SubgraphExtnsn()* algorithm (step 4). Significant subgraphs are generated by iteratively executing *SubgraphExtnsn()* algorithm for each graph *G*, in the data set *D*. At this stage significant graphs are generated for the given *coreset* and the last step prunes out redundant patterns if any appeared.

4.1 Finding Representative Set

The proposal is to mine significant elements and use them to identify significant subgraph patterns in the data set. In main algorithm the frequent edges generated at step 1 are the input to this algorithm. All frequent k-edge subgraphs are generated at first and then elements of *coreset* are identified which possess the properties explained in the introduction of *coreset*. Along with those the *coreset* elements in the proposed algorithm capture structural information of the patterns so that the graph occurrence is certain in the outcome. The elements of *coreset* are further used to construct result sets. The process of identifying *coreset* is defined in the below algorithm.

Algorithm: ***findcoreset()***

Input: $F^1 \leftarrow$ All frequent 1- edge subgraph in D in Canonical edge form.

Output: *Coreset CS.*

$S^1 \leftarrow$ All frequent 1- edge subgraph in D in Canonical edge form.

S, $S^j \leftarrow$ a data structure to store intermediate graphs.

Begin

1. $S^j \leftarrow S^1$
2. while $j<k$, for each *s* from S^j do.
3. while $S^1 \neq \emptyset$ do.
4. let e' be last edge of *s* and for each edge *e* in S^1.
5. if *e* can be used to extend *e*' then.
6. *ext* ← *s* <> *e*.
7. if *ext* is not already generated then.
8. $S^j \leftarrow S^j \cup ext$.
9. $S \leftarrow S^j$ All subgraphs with k-edge in edge-format .
10. for each *s* in *S* do.
11. find *fcnt* of *s* in *S*.
19. find *t* and *T* // set threshold *T and t – min and max threshold* based on ranges appeared and user given significance.
20. for each *s* in *S* do.
21. if *s.fcnt* ≥ *t* or *s.fcnt* ≤ *T* then.
22. $CS^k \leftarrow CS^k \cup s$.
23. for each s in CS^k
24. if $sim(s_1, s_2) \leq x$ then.
25. $CS \leftarrow CS \cup s$

End

This algorithm takes frequent 1-edge graphs as input and tries to extend the size of subgraph by adding an edge at a time. During iteration, it generates edge subgraphs and the size of each subgraph is greater than the previously generated subgraph by one edge. In such a way it generates k-edge subgraphs (steps 1-8). The frequency of each of these k-edge subgraphs is identified. Based on the actual frequency in the graph data set and the user defined significance, upper and lower frequency sup-

port thresholds are set and all frequent k-edge subgraphs of the data set *S* are identified (steps 9-22). As we know subgraphs with higher frequencies may not necessarily reflect their respective significance, upper and lower frequency thresholds are identified using normalization technique. Now we have to discard the maximal overlapping elements of the set to reduce redundant supergraphs generation as overlapping elements are relatively not independent. To discard similar graphs graph distance metric (23) along with frequency support is used. According to the statistical frequency and relativeness k^{th} elements are chosen and identified as elements of *corset* (step 24-25). A corset is a set of k-edge subgraphs that are statistically frequent and participate in resultant significant frequent graphs. Identification of *coreset* prunes subgraphs that may generate redundant graphs and that do not satisfy the support constraint as well as which may not become the significant parts in resultant graphs.

4.2 Edge Graphs

Generally significant graph mining algorithms are used along with frequent graph mining algorithms to retrieve frequent significant graphs. We proposed a technique to mine graph patterns.

A graph is an ordered pair $G = (V, E)$ made up of set of vertices V and the set of edges E. The graph is labelled, that is, each vertex and edge has an associated label from the given set of vertex labels $L(V)$ and edge labels $L(E)$. Each vertex (or edge) of the graph may not have a unique label. In other words, many vertices (or edges) of the graph may have same label. Along with the actual labels of vertices in the graph, vertices are labelled by using consecutive integers {1,…,n} where n is number of vertices in the graph. Figure 4 shows an example graph. Actual vertices of the graph *G* are{x,y,x,z,x}. An edge is shown with pair of vertices it connected and the label of that edge (u,v,l). The edges of the above graph are (x,x,a), (x,y,a), (x,y,b),(x,z,c),(x,z,d). It is observed that the label of the edge may be different even if the labels of comprised vertices are same. Vertices are labelled by using integers to identify each and every vertex uniquely and thus to identify edges. Now the edges of graph *G* are represented in the form of (1-3), (1-2), (1-4), (3-4), (2-5).

Let *G* be a graph with a single label per vertex, $S(V_S, E_S, L_S)$ be a subgraph of *G* and all of its vertex labels are unique, i.e., $L_S(v) \neq L_S(v')$for all v and v' in V_S such that $v \neq v'$. To calculate subgraph *S* directly, the *S* to *G Coreset* has to be considered besides the domains of variables by enforcing vertex and edge consistency. Since all vertices of graph have unique labels and the query has unique labels, we can easily identify every vertex that participates in subgraphs and no vertex can appear more than once.

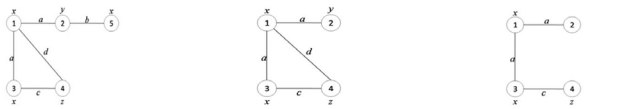

Figure 4: Edge representation of graph *G* and its subgraphs

In actual graph, it is difficult to identify each vertex and edge uniquely as three vertices contain the same label x. That's why, extension of subgraphs and finding embedded edges and vertices are also difficult. The complexity involved in the above tasks can be reduced by using unique labels which are introduced in edge graph notation. According to edge-graph notation used to represent the graph *G*, vertices are represented as (1,x), (2,y), (3,x), (4,z), (5,x) and edges are (1,2,a), (2,5,b), (1,3,a), (3,4,c), (1,4,d). In computation, edge (1-2) represents an actual edge (1,2,a) that in turn (x,x,a). The size of a graph $G = (V, E)$ is defined to be equal to number of edges in a graph $|E|$. Adding an edge $e = (u, v)$ to a *size-k* connected graph results connected $(k + 1)$ sized graph.

1-edge subgraphs of graph *G* are represented as {(1-3), (1-2), (1-4), (3-4), (2-5)} shown in Figure 5. Here the edge set of graph is represented and from this, only the possible subgraph extensions can be generated with reference to the actual graph, contrary to the general perception. As a consequence, expensive generation of all combinational subgraphs that don't appear in actual graph are not necessary. Thus set of 2-edge subgraphs generated from 1-edge subgraphs of graph *G* are {(1-3,1-2), (1-3,1-4), (1-3,3-4), (1-2,1-4), (1-2,2-5), (1-4,3-4)} and 5-edge subgraph is {(1-3, 1-2, 1-4, 3-4 , 2-5)} and is same of original graph. If we observe that subgraphs at same edge level share the core subgraph and *k+1-edge* subgraphs differ from *k-edge* subgraphs by one edge only. These edge subgraphs are representatives of original graph and identification of each subgraph can be done uniquely. Extending a subgraph can be done easily and there is no scope of checking for automorphism. These *k-edge* subgraphs can be used directly to find frequent subgraphs at specified size thus reducing the computational cost of the identification and

exploration of frequent subgraphs at lower levels thus pruning the search space based on size. The algorithm used to extend *coreset* elements is explained below.

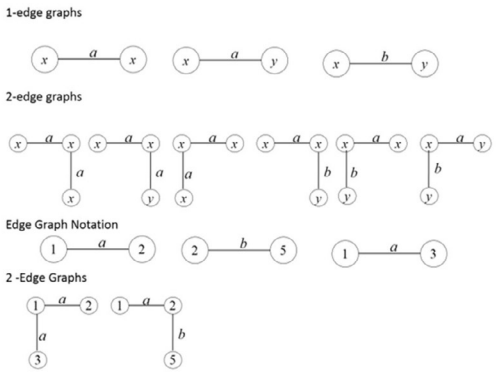

Figure 5: generated subgraphs with edge-graph notation

4.3 k-Edge Subgraph Extraction

Algorithm: ***SubgraphExtnsn()***

Input: *Coreset* of D

Output: Supergraphs of coreset

Begin

1. Let $G^1 \leftarrow$ All 1-edge graphs of *G*. $D = D \backslash \{G\}$
2. $CS_g \leftarrow$ All corset graphs of *G*.
3. *For each s in CS*
4. while $k<n$ or while no *ext* // n- edges of original graph
5. let *e'* be last edge in *s* and for each edge *e* in G^1
6. if *e* can be used to extend *e'* then
7. $ext \leftarrow s <> e$
8. if *ext* is not already generated
9. $S^k \leftarrow S^k \cup ext$
10. $S \leftarrow S \cup S^k$

End.

In this phase, a set of subgraphs of size-k is extracted. Subgraphs of size k+1 are generated by extending an edge of k-edge subgraph. This algorithm first identifies the subset of corset that belongs to the graph G. Then on each subgraph *s* of the corset CS_g algorithm tries to extend one edge at a time. An edge *e* in the set of 1-edge of a graph *G* is used to extend subgraphs. All applicable extensions that have not been previously considered are stored in S^k. Edge canonical form is adopted to exclude already generated extensions. Edge subgraph extension is recursively carried out to further extend edge subgraphs until k-size subgraphs are generated. If the subgraph reaches size k or further extension is not possible, the algorithm will be terminated and the resultant significant patterns are stored in *S*.

5. Experimental Evaluation

SSGM Algorithm is implemented in C language. The experiments are carried out on a Intel® Pentium® Dual CPU T3400 @2.17 GHz with 4GB RAM. To evaluate the performance on real datasets, we used the data sets in a standard graph library available at the[32]. It provides information on the anti-cancer screen tests with different cancer cell lines. From these, each dataset belongs to a specific type of cancer with active and inactive. The dataset is sparse, containing 66 vertices types and four types of edges. The largest graph has 214 vertices and 214 edges, on an average 43 vertices and 45 edges.

Another protein data set is from the Pubchem website[33]. Pubchem is a well maintained collection of various molecules. It consists of 1178 proteins that are divided into 691 enzymes and 487 non-enzymes. Average vertices per graph are 285 and edges are 715. Different vertex labels available are 82.

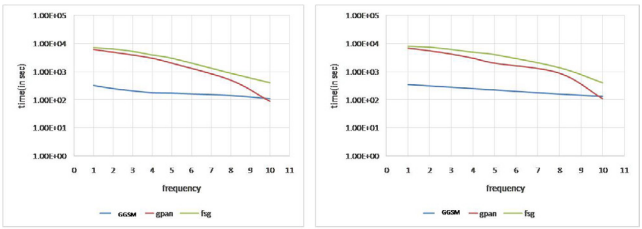

Figure 6: Frequency Vs Time

The scalability of the algorithm against frequency and data set size are examined shown in Figure 6. For these experiments FSG and gSpan are chosen to compare as they are part of the pipeline in the alternative approach. The scalability of proposed algorithm is linear for mining significant subgraphs on the data set. The frequency threshold varied between 1% and 10%. It can be observed that FSG and gSpan grow exponentially whereas the proposed algorithm grows linearly. At higher frequency the gSpan is little bit faster than the proposed one as the proposed algorithm requires computation of *coreset* where as those start mining directly. But, it is negligible when compared with the cost of mining at less frequency thresholds.

Around 30% of computation cost is spent for the computation of *coreset* since we consider all frequent k-edge subgraphs in the data set to identify *coreset*. This is the

reason gSpan is faster than the proposed one at higher frequency threshold. However, when compared with the cost at lower frequencies this cost is negligible.

To demonstrate the scalability of proposed algorithm with the varying database size is shown in Figure 7. The datasets for this experiment are chosen randomly from the cancer data set. The size of dataset is varied between 1000 and 30000 and the frequency threshold is set to 1. The proposed algorithm to find significant subgraphs has shown better performance than FSG and gSpan.

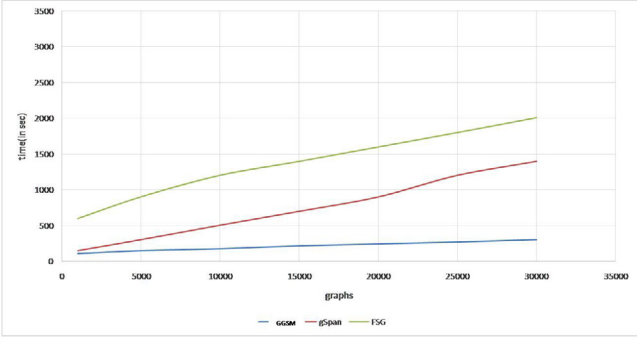

Figure 7: Database size Vs Time

6. Conclusion

The emphasis of proposed algorithm is to provide a scalable approach to mine significant subgraphs at low frequency threshold from graph database thus make available to perform graph applications like classification and indexing. The proposed *coreset* concept evaluate the significance of the subgraphs based on topological structure, frequency and similarity between candidate subgraphs thus reducing exponential search space. The proposed edge subgraph based mining technique is capable of retrieving the required subgraphs directly from the graphs consequently avoiding unnecessary subgraph extensions and reducing computation cost. The results obtained in the preliminary tests confirmed the effectiveness of the proposed algorithm. The proposed algorithm can also be applied to search structures based on the user given constraints for general applications.

7. References

1. Cook D, Holder L. Mining Graph Data. John Wiley and Sons Inc; 2007.
2. Ulaganathan PP, Thirusangu K, Selvam B. Super edge-magic total labelling in extended duplicate graph of path. Indian Journal of Science and Technology. 2011 May; 4(5). DOI: 10.17485/ijst/2011/v4i5/30069.
3. Ulaganathan PP, Thirusangu K, Selvam B. Graceful and Skolem graceful labelling in extended duplicate graph of path. Indian Journal of Science and Technology. 2011 Feb; 4(2). DOI: 10.17485/ijst/2011/v4i2/29943.
4. Kumar R, Raghavan P, Rajagopalan S, Sivakumar D. Tomkins A, Upfal E. The Web as a Graph. ACM PODS Conference; NY. 2000. p. 1–10.
5. Maio D, Maltoni D. A structural approach to fingerprint classification. Proceedings of 13th International Conference on Pattern Recognition; Vienna, Austria. 1996. p. 578–85.
6. Eichinger F, Bohm K, Huber M. Improved software fault detection - with Graph Mining. Workshop on Mining and Learning with Graphs; 2008. p. 1–3.
7. Gupta MS, Pathak A, Chakrabarti S. Fast algorithms for top-k personalized pagerank queries. WWW Conference; Beijing, China. 2008 Apr. p. 1225–6.
8. Koyuturk M, Grama A, Szpankowski W. An efficient algorithm for detecting frequent subgraphs in biological networks. Bioinformatics. 2014; 20(1):200–7.
9. Zhou B, Pei J. Preserving privacy in social networks against neighborhood attacks. ICDE Conference; Cancun. 2008. p. 506–15.
10. Das K, Ray J, Mishra D. Gene selection using information theory and statistical approach. Indian Journal of Science and Technology. 2015 May; 8(S9). DOI: 10.17485/ijst/2015/v8iS9/51494.
11. Ramya N. On colourings of wheel graph (Wn). Indian Journal of Science and Technology. 2014 Mar; 7(S3). DOI:10.17485/ijst/2014/v7i3S/50405.
12. Bordino I, Donato D, Gionis A, Leonardi S. Mining large networks with subgraph counting. IEEE ICDM Conference; Pisa. 2008 Dec. p. 737–42.
13. Inokuchi A, Washio T, Motoda H. An apriori-based algorithm for mining frequent substructures from graph data. Principles of Data Mining and Knowledge Discovery; UK. 2000. p. 13–23.
14. Nijssen S, Kok JN. The Gaston tool for frequent subgraph mining. Proceedings of the International Workshop on Graph-Based Tools, Grabats, Electronic Notes In Theoritical Computer Science. Rome, Italy. 2004 Oct; 127(1):77–87.
15. Kuramochi M, Karypis G. An efficient algorithm for discovering frequent subgraphs. IEEE Transaction on Knowledge Data Engineering. 2004; 16(9):1038–105.
16. Yan X, Han J. gSpan: Graph-based substructure pattern mining. Proceedings of 2002 IEEE International Conference on Data Mining; 2002 Sept. p. 1–26.
17. Huan J, Wang W, Prims J. Efficient mining of frequent subgraph in the presence of isomorphism. University of North Carolina Computer Science Technical Report; 2003. p. 1–12.

18. Huan J, Wang W, Prins J, Yang J. SPIN: Mining maximal frequent subgraphs from graph databases. Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; NY. 2004. p. 581–6.
19. Vedanayaki M. Graph mining techniques, tools and issues - A study. Indian Journal of Science and Technology. 2014 Nov; 7(S7). DOI: 10.17485/ijst/2014/v7iS7/61962.
20. Kudo T, Maeda E, Matsumoto Y. An application of boosting to graph classification. Advances in Neural Information Processing Systems 18 (NIPS'04); 2004. p. 1–8.
21. Ranu S, Singh AK. GraphSig: A scalable approach to mining significant subgraphs in large graph databases. Proceedings of IEEE International Conference on Data Engineering; Shanghai. 2009 Apr. p. 844–55.
22. Yan X, Cheng H, Han J, Yu PS. Mining significant graph patterns by scalable leap search. Proceedings of SIGMOD; USA. 2008. p. 433–44.
23. Deshpande M, Kuramochi M, Wale N, Karypis G. Frequent substructure-based approaches for classifying chemical compounds. IEEE Transactions on Knowledge and Data Engineering. 2005; 17(18):1036–50.
24. Kramer S, Raedt LD, Helma C. Molecular feature mining in HIV data. KDD '01: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; USA. 2001. p. 136–43.
25. Yan X, Yu PS, Han J. Graph indexing: A frequent structure-based approach. Proceeding of SIGMOD; NY. 2004. p. 335–46.
26. Cheng H, Yan X, Han J, Hsu C. Discriminative frequent pattern analysis for effective classification. Proceeding of ICDE; Istanbul. 2007. p. 716–25.
27. Hasan M, Chaoji V, Salem S, Besson J, Zaki M. ORIGAMI: Mining representative orthogonal graph patterns. Proceeding of ICDM; NE. 2007. p. 153–62.
28. Tan P, Kumar V, Srivastava J. Selecting the right interestingness measure for association patterns. Proceeding of SIGKDD; 2002. p. 32–41.
29. Wale N, Ning X, Karypis G. Trends in chemical graph data mining. Managing and Mining Graph Data, Springer. 2010 Dec; 40:581–606.
30. Agarwal PK, Peled SH, Varadarajan K. Geometric approximation via *corsets*. International Journal of Engineering Goodman. Welzl, editor. Combinatorial and Computational Geometry, Cambridge University Press; 2005. p. 1–30.
31. Bunke H, Shearer K. A graph distance metric based on the maximal common subgraph. Pattern Recognition Letters. 1998; 19(3-4):255–9.
32. Chen JH, Linstead E, Swamidass SJ, Wang D, Baldi P. Chem DB - Update-full-text search and virtual chemical space. Bioinformatics. 2007; 23(17):2348–51.
33. Available from: http://pubchem.ncbi.nlm.nih.gov