Efficient Mining of Frequent Subgraph with Connectivity Constraint

Hyun S. Moon[1] **Kwang H. Lee**[2] **Doheon Lee**[2]
[1]*Division of Computer Science, KAIST, Taejeon, Korea*
[2]*Department of BioSystems, KAIST, Taejeon, Korea*
Email: {shmoon,khlee,dhlee}@biosoft.kaist.ac.kr

ABSTRACT:

The goal of data mining is to extract new and useful knowledge from large scale datasets. As the amount of available data grows explosively, it became vitally important to develop faster data mining algorithms for various types of data.

Recently, an interest in developing data mining algorithms that operate on graphs has been increased. Especially, mining frequent patterns from structured data such as graphs has been concerned by many research groups. A graph is a highly adaptable representation scheme that used in many domains including chemistry, bioinformatics and physics. For example, the chemical structure of a given substance can be modelled by an undirected labelled graph in which each node corresponds to an atom and each edge corresponds to a chemical bond between atoms. Internet can also be modelled as a directed graph in which each node corresponds to an web site and each edge corresponds to a hypertext link between web sites. Notably in bioinformatics area, various kinds of newly discovered data such as gene regulation networks or protein interaction networks could be modelled as graphs.

There have been a number of attempts to find useful knowledge from these graph structured data. One of the most powerful analysis tool for graph structured data is frequent subgraph analysis. Recurring patterns in graph data can provide incomparable insights into that graph data.

However, to find recurring subgraphs is extremely expensive in computational side. At the core of the problem, there are two computationally challenging problems. 1) Subgraph isomorphism and 2) Enumeration of subgraphs. Problems related to the former are subgraph isomorphism problem (Is graph A contains graph B?) and graph isomorphism problem (Are two graphs A and B the same or not?). Even these simplified versions of the subgraph mining problem are known to be NP-complete or Polymorphism-complete and no polynomial time algorithm has been existed so far. The later is also a difficult problem. We should generate all of 2^n subgraphs if there is no constraint where n is the number of vertices of the input graph. In order to find frequent subgraphs from larger graph database, it is essential to give appropriate constraint to the subgraphs to find. Most of the current approaches are focus on the frequencies of a subgraph: the higher the frequency of a graph is, the more attentions should be given to that graph. Recently, several algorithms which use level by level approaches to find frequent subgraphs have been developed.

Some of the recently emerging applications suggest that other constraints such as connectivity also could be useful in mining subgraphs : more strongly connected parts of a graph are more informative. If we restrict the set of subgraphs to mine to more strongly connected parts, its computational complexity could be decreased significantly.

In this paper, we present an efficient algorithm to mine frequent subgraphs that are more strongly connected. Experimental study shows that the algorithm is scaling to larger graphs which have more than ten thousand vertices.

1 INTRODUCTION

The goal of data mining is to extract new and useful knowledge from large scale datasets. As the amount of available data grows explosively, it became vitally important to develop faster data mining algorithms for various types of data.

Recently, an interest in developing data mining algorithms that operate on graphs has been increased. Especially, mining frequent patterns from structured data such as graphs has been concerned by many research groups. A graph is a highly adaptable representation scheme that used in many domains including chemistry, bioinformatics and physics. For example, the chemical structure of a given substance can be modelled by an undirected labelled graph in which each node corresponds to an atom and each edge corresponds to a chemical bond between atoms. Internet can also be modelled as a directed graph in which each node corresponds to an web site and each edge corresponds to a hypertext link between web sites. Notably in bioinformatics area, various kinds of newly discovered data such as gene regulation networks or protein interaction networks could be modelled as graphs.

There have been a number of attempts to find useful knowledge from these graph structured data. One of the most powerful analysis tool for graph structured data is frequent subgraph analysis. Recurring patterns in graph data can provide incomparable insights into that graph data.

However, to find recurring subgraphs is extremely expensive in computational side. At the core of the problem, there are two computationally challenging problems. 1) Subgraph isomorphism and 2) Enumeration of subgraphs. Problems related to the former are subgraph isomorphism problem (Is graph A contains graph B?) and graph isomorphism problem (Are two graphs A and B the same or not?). Even these simplified versions of the subgraph mining problem are known to be NP-complete or Polymorphism-complete and no polynomial time algorithm has been existed so far. The later is also a difficult problem. We should generate all of 2^n subgraphs if there

is no constraint where n is the number of vertices of the input graph. In order to find frequent subgraphs from larger graph database, it is essential to give appropriate constraint to the subgraphs to find. Most of the current approaches are focus on the frequencies of a subgraph: the higher the frequency of a graph is, the more attentions should be given to that graph. Recently, several algorithms which use level by level approaches to find frequent subgraphs have been developed.

Some of the recently emerging applications suggest that other constraints such as connectivity also could be useful in mining subgraphs : more strongly connected parts of a graph are more informative. If we restrict the set of subgraphs to mine to more strongly connected parts, its computational complexity could be decreased significantly. This is the basic motivation for this work.

1.1 Related Works

Theoretical foundation of graph isomorphism problem was established in 1970s. Given two graphs $G_x(V_x,E_x)$ and $G_y(V_y,E_y)$, the *subgraph isomorphism* problem is to find the subgraph of G_x, $G_{sx}(V_{sx},E_{sx})$ such that G_{sx} is identical to G_y if such a subgraph exist. Ullman showed that the problem is NP-complete and proposed a brute-force algorithm for the problem [4]. More simplified version of this problem is *graph isomorphism* problem and *canonical labelling* problem. Graph isomorphism problem is to determine whether two graphs $G_x(V_x,E_x)$ and $G_y(V_y,E_y)$ are equivalent or not. To find canonical labelling is labelling of vertices for a graph such than two graphs have the same canonical labelling if and only if two graph are isomorphic to each other. These two problems are known to be equivalent and neither NP-completeness proof nor polynomial time algorithm has been presented for the problem [5].

Recently, several subgraph mining algorithms have been presented for varied class of problems. Subgraph mining problems can be classified into two major categories according to the input graph setting : the *single-graph setting* and the *graph transaction setting*. In the single-graph setting, input is a large single graph and the frequency of a subgraph is how many times does it occur in the single graph. Whereas in the graph transaction setting, input is a set of relatively small graphs and the frequency of a subgraph is how many input graphs have the subgraph.

Although a number of algorithms have been developed for both of those settings, Few approaches focus on the connectivity of the target subgraphs to mine. Several applications of the frequent subgraph mining problem have reported that connectivity as well as frequency gives important information about the input graph.

1.2 Our Contributions

In this paper, we present an efficient algorithm for mining subgraphs with higher connectivity. Most of current algorithms focus on connected subgraphs. However in some applications, we are interested in mining more strongly connected subgraphs although their frequencies are not as high as other subgraphs. Connectivity of a graph, the minimum number of vertices to partition the graph, is a good measure to determine whether a graph is strongly connected or not. But connectivity related problems such as k-connected minimum spanning subgraph are often NP-complete for general k with greater than 2. Thus, we focus on biconnected graphs whose properties are well known.

Although there are several algorithms for subgraph mining, the problem considered in this paper is different from previous approaches in following points. First, a large single graph is given as input whereas many small graphs are given in most of previous approaches [10, 12, 13]. Second, we do not summarize or contract the input graph which result in loss of information or incompleteness [7, 8]. Third, we do not restrict target graphs to vertex-disjoint embeddings [6]. Fourth, our algorithm assume that the input graph is unlabeled or have only a few kinds of labels. Current approaches assume that vertices and edges are well-labeled and it greatly reduces computational complexity of the problem [9, 11]. Fifth, current approaches do not consider any topological characteristics of subgraphs to mine.

2 Background

A graph $G = (V,E)$ is made of two sets, the set of vertices V and the set of edges E. Each edge itself is a pair of vertices. Throughout this paper we assume that the graph is unlabelled and undirected. That is, all vertices and edges have the same label. Given a graph $G = (V,E)$, a graph $G_s = (V_s,E_s)$ is a subgraph if and only if $V_s \subset V$ and $E_s \subset E$. An induced subgraph is a subset of the vertices of a graph G together with any edges whose endpoints are both in this subset. Two graphs $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$ are isomorphic if they are topologically identical to each other.

A graph is connected if there is a path between every pair of vertices in the graph. A graph is biconnected if there are at least two paths between every pair of vertices in the graph. A cycle graph is a graph on n nodes containing a single cycle through all nodes. A path graph is a tree with two nodes of vertex degree 1, and the other nodes of vertex degree 2. In the rest of the paper, we will refer a subgraph of a graph G to an induced subgraph of G. Similarly, A cycle graph of G is an induced cycle subgraph of G and a path graph of G is an induced path subgraph of G if no otherwise specified. A path graph between u and v in G is a path graph of G where u and v are two vertices with degree 1.

A joint graph of two graphs $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$, $G_1 \cup G_2$, is a graph with a vertex set $V_1 \cup V_2$ and an edge set $E_1 \cup E_2$.

3 Mining Biconnected Frequent Subgraphs

In designing an algorithm for efficient mining of biconnected subgraphs, we focus on the fact that a biconnected graph can be obtained by joining several cycle graphs which can be relatively easily enumerated. Biconnected Frequent Subgraph Mining (BFSM) algorithm consist of following steps. First, for a given graph, we enumerate every path graph within size $k/2$. Second, joining two path graphs with the same ends, we get a get of cycle graphs in G. This set is the initial set of biconnected subgraphs. Third, joining two biconnected sub-

graph, we enumerate other biconnected subgraphs until no more join operation is available. The overview of the BFSM algorithm is described in Algorithm 3.

Algorithm 1 BFSM (G,n)

Require: G is an input graph and n is the maximum size of subgraph to find.
Ensure: B is the set of all biconnected subgraphs of G
1: $C \leftarrow$ CycleGraph(G,n)
2: $B \leftarrow$ BiGraph(C,n)

3.1 Enumeration of Cycle Graphs

Cycle graph C_n is a graph on n nodes containing a single cycle through all nodes. For a given graph G, every induced cycle graph in G can be enumerated by joining simple backtracking algorithm. The algorithm begin with an arbitrary starting vertex in G.

Algorithm 2 CycleGraph $(G=(V,E),n)$

Ensure: C is a set of all cycle graphs in G within size n
1: $C \leftarrow \emptyset$
2: **for all** u,v such that $u,v \in V$ **do**
3: Let $P(u,v)$ be the set of all path graphs between u,v within size $n/2$
4: **for all** $G_1=(V_1,E_1), G_2=(V_2,E_2)$ such that $G_1,G_2 \in P(u,v)$ **do**
5: **if** $V_1 \cup V_2 = \{u,v\}$ **then**
6: $C = C \cup \{G_1 \cup G_2\}$
7: **end if**
8: **end for**
9: **end for**

3.2 Enumeration of Biconnected Graphs

In this section, we will propose an algorithm that enumerates all biconnected subgraphs of an input graph G. The algorithm begins with the set of all induced cycle graphs of G. The idea is that by joining two induced biconnected subgraphs, we get a new induced biconnected subgraph. The algorithm is described in Algorithm 3.2.

3.3 Correctness

In this section, correctness of Algorithm 3 will be shown. We will prove that the proposed algorithm enumerates all induced biconnected subgraphs of G within size n by showing following two. First, joining two induced biconnected subgraphs of G as described in line 6 and 7 in Algorithm 3.2, we always get a induced biconnected subgraph. Second, any induced biconnected subgraph of G can be constructed by a join operation of induced cycle graphs in G.

Lemma 1. *Let $G_1=(V_1,E_1)$ and $G_2=(V_2,E_2)$ be two biconnected subgraphs of G. If $|V_1 \cup V_2| > 1$ then $G_1 \cup G_2$ is also a biconnected subgraph of G.*

Algorithm 3 BiGraph (C,n)

Require: C is a set of all cycle graphs in G within size n
Ensure: B is a set of all biconnected graphs in G within size n
1: $B \leftarrow \emptyset$
2: $T \leftarrow C$
3: **while** $T \neq \emptyset$ **do**
4: Let $G_t=(V_t,E_t)$ be an element of T
5: **for all** $G_b=(V_b,E_b)$ such that $G_b \in B$ **do**
6: **if** $|V_t \cap V_b| > 1$, $|G_t \cup G_b| \leq n$, and $G_t \cup G_b \notin B,T$ **then**
7: $T \leftarrow T \cup \{G_t \cup G_b\}$
8: **end if**
9: **end for**
10: $T \leftarrow T - \{G_t\}$
11: $B \leftarrow B \cup \{G_t\}$
12: **end while**

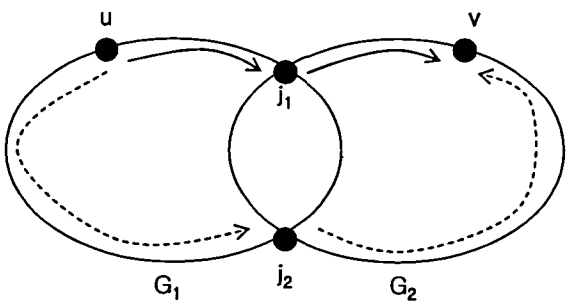


Figure 1: For any $u,v(u \in V_1, v \in V_2)$, there are two vertex disjoint paths between u and v. One is through j_1 (solid arrow) and the other is through j_2 (dashed arrow).

Proof : Since G_1 and G_2 are biconnected, for all vertex pairs (u,v) in G_1, there are always more than two paths between u and v and the same holds for G_2. Thus, it will be shown that for any vertex pairs (u,v) such that $u \in V_1$ and $v \in V_2$, there are always more than two paths. Let j_1 and j_2 be two vertices in $G_1 \cap G_2$. Since G_1 is biconnected, for any vertex $u(u \in V_1)$, there is a cycle including u, j_1, j_2 in G_1. Similarly, for any vertex $v(v \in V_2)$, there is a cycle including v, j_1, j_2 in G_2. Thus, for any $u,v(u \in V_1, v \in V_2)$, path $u \cdot v_1 \cdot v$ and path $u \cdot v_2 \cdot v$ are two vertex disjoint paths from u to v. (See Figure 1) Therefore a graph $G_1 \cup G_2$ is biconnected.

Lemma 2. *Let $B=(V_b,E_b)$ be a biconnected subgraph of a graph G. Then there always exist a set of cycle subgraph of G , $\{C_1,C_2,\cdots,C_n\}$ such that $B = \cup_{k=1}^{n} C_k$.*

Proof : It will be proven by showing that for any edge in B, there is a induced cycle subgraph of G containing that edge. Let $e=(u,v) \in E_b$ be an edge in B. Since B is biconnected, $B'=(V_b,E_b-\{(u,v)\}$ is connected and there exists a shortest path between u and v in B'. Let $u \cdot x_1 \cdot x_2 \cdots x_m \cdot v$ be the shortest path between u and v in B'. Then an induced subgraph of G by a vertex set $\{u,v,x_1,x_2,\cdots,x_m\}$ is an induced cycle subgraph of G.

Corollary 1. *Algorithm 3 enumerates all biconnected subgraphs of an input graph G.*

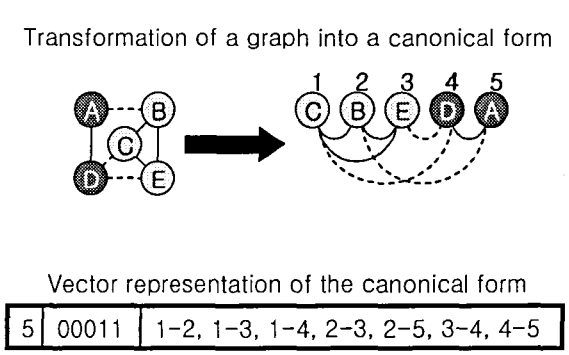


Figure 2: Example of transformation of a graph into the canonical form. The labels on the vertices are labeling of the input and the digits above the vertices are canonical labelings. A vector representation of the canonical form was shown below.

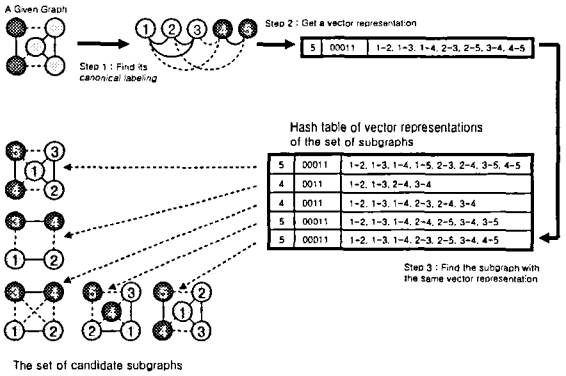


Figure 3: Determination of the isomorphic graph for a given graph. The digits on the vertices are canonical labeling of the graphs.

3.4 Frequency Counting and Frequency Constraint

In order to count frequencies of subgraphs, we should compare structures of subgraphs. Comparing the structures of two graphs is a graph isomorphism problem. Solving this problem involves finding a homomorphism function h, if it exists. For two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, a homomorphism function $h : V_1 \rightarrow V_2$ is a bijection mapping (both a one-to-one and an onto function) that satisfies the following condition: $(v,w) \in E_1$ if and only if $(h(v), h(w)) \in E_2$. We used canonical labelling instead of finding homomorphism function since it is more efficient for our purpose. Canonical labelling of a graph is a permutation of the vertices of the graph which guarantee that canonically relabelled forms of two graphs are the same if and only if they are isomorphic to each other. Even though it requires sub-exponential time in the worst case, the algorithm devised by B. McKay is efficient for addressing the problem of canonical labelling [5].

In the problem of frequent subgraph detection, we have a set of candidate frequent subgraphs. For each enumerated subgraph, we have to determine which of the candidates is isomorphic to the generated subgraph. We use a hash table for this process. We store canonically relabeled graphs in a hash table instead of the graphs themselves. Once a canonical labeling of a graph is found, we can represent the graph as a vector of integers. The vector representation of a graph includes following information.

- The number of the vertices of the graph.
- The labels of the vertices in canonical order.
- Sorted list of the edges in the canonical form.

Figure 2 shows how a graph is represented as a vector form. The first part of the vector representation is the number of vertices (in this example, five). The second part of the vector is the list of colors of the vertices. In the example, 0 means light vertices while 1 means dark vertices. The third part of the vector consists of all edges in the canonical form. For example, 1-2 means that there is an edge between vertex 1 and vertex 2 where 1 and 2 are canonical labels.

For efficient implementation, we built a hash table of vector representations of candidate frequent subgraphs. The steps are: 1) Find canonical labelling of the given graph. 2) Get a vector representation. 3) Find the candidate subgraph isomorphic to the given graph. 4) If found, increase the counter. Otherwise, insert the given graph to the set of candidate frequent subgraphs. The overall procedure is described in Figure 3.

4 Experiment

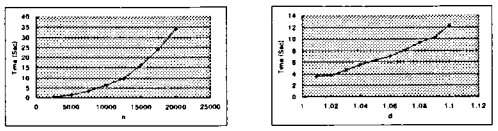
We performed our experimental study using a single processor of a 2GHz Pentium PC with 1GB memory running Linux. The presented algorithm was compiled using g++ with O4 optimization level. We applied our algorithm to synthetically generated graphs. Running time of the algorithm is mainly affected by three factors: size of the input graph, density of the input graph and the size of the subgraphs to mine. In this section, we'll show how these factors affect on the running time of the algorithm.

4.1 Random Graph

We generated random graphs by controlling two parameters. The number of vertices in the graph, and the density of the graph. The density of a graph is a measure to determine how densely vertices are interconnected and represented by $\log_{|V|}|E|$ where $|V|$ is the number of vertices and $|E|$ is the number of edges. The maximum value of the density is 2 since a graph has at most $|V||V-1|$ edges. In mining frequent subgraphs we assume that a target graph is very sparse and its density is often less than 1.1. The result shows that all of biconnected subgraphs within a certain size can be enumerated in reasonable time bound by the proposed algorithm.

5 Conclusion

In this paper we presented an algorithm called BFSM to find biconnected frequent subgraphs from a large single unlabeled



(a) Run time for input graph size ($d = 1.05, k = 6$)

(b) Run time for input graph density ($n = 10000, k = 6$)

Figure 4: Performance of the algorithm for various complexity of random input graph. The result shows scalability of the algorithm for input graph size or input graph density.

graph and evaluated its efficiency and scalability through experiments using randomly generated graphs with different size and density. Our results showed that BFSM is highly scalable and can operate on very large graph with a few labels.

However, as the size of target subgraph to mine grows, its computational cost grows exponentially. Currently, subgraphs whose size is less than 10 can be enumerated and counted in reasonable time bound. Mining of larger subgraphs with lower frequency still remains an open problem.

5.1 Citations and References

References should appear in a separate section at the end of the paper, double-spaced, with items referred to by numerals in square brackets [1]. References must be complete and in the following style:

- Style for papers: Author, first initials followed by last name, title in quotations, periodical, volume, page numbers, month, year [2].
- Style for books: Author, title. Publisher, Location, chapter, and page numbers (if desired) [3].

REFERENCES

[1] S. Dutta. An event-based fuzzy temporal logic. In Proc. 18th IEEE Intl. Symp. on Multiple-Valued Logic, pages 64–71, Palma de Mallorca, Spain, 1988.

[2] M. G. Harbour, M. H. Klein, and J. P. Lehoczky. Timing analysis for priority scheduling of hard real-time systems. IEEE Transactions of Software Engineering, 20(1):13–28, 1994.

[3] Esko Turunen. Mathmatics Behind Fuzzy Logic. Advances in Soft Computing. Verlag, Heidelberg, 1999.

[4] J. R. Ullman. An algorithm for subgraph isomorphism. Journal of the ACM, 23(1):31–42, 1976.

[5] B. McKay. Computing automorphisms and canonical labelling of graphs. Lecture Notes in Mathematics, 686:223–232, 1978.

[6] M. Kuramochi, G. Karypis. Finding Frequent Patterns in a Large Sparse Graph. 2004 SIAM International Conference on Data Mining (SDM04), 2004.

[7] L. B. Holder, D. J. Cook, S. Djoko. Substructure discovery in the SUBDUE system. Proc. of the AAAI Workshop on Knowledge Discovery in Databases, 169–180, 1994.

[8] S. Ghazizadeh and S. Chawathe. SEuS: Structure extraction using summaries. Proc. of the 5th International Conference on Discovery Science, 2002.

[9] N. Vanetik, E. Gudes, S. E. Shimony. Computing frequent graph patterns from semistructured data. Proc. of 2002 IEEE International Conference on Data Mining (ICDM02), 458–465, 2002.

[10] A. Inokuchi, T. Washio, H. Motoda. Complete Mining of Frequent Patterns from Graphs: Mining Graph Data. Machine Learning, 50:321–354, 2003.

[11] C. Borgelt, M. R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. Proc. of 2002 IEEE International Conference on Data Mining (ICDM02), 2002.

[12] J. Huan, W.Wang, J. Prins. Efficient mining of frequent subgraph in the presence of isomophism. 2003 IEEE International Conference on Data Mining (ICDM03), 2003.

[13] X. Yan, J. Han. gSpan: Graph-based substructure pattern mining. Proc. of 2002 IEEE International Conference on Data Mining (ICDM02), 2002.

[14] R. Agrawal, R. Srikant. Fast algorithms for mining association rules. Proc. of the 20th VLDB Conference, 487–499, 1994.

[15] M. Kuramochi, G. Karypis, GREW: A Scalable Frequent Subgraph Discovery Algorithm. International Conference on Data Mining (ICDM04), 2004.