Specific Biases for Mining Frequent Substructures

Akihiro Inokuchi[1,2], Takashi Washio[2], and Hiroshi Motoda[2]

[1] Tokyo Research Laboratory, IBM Japan,
1623-14, Shimotsuruma, Yamato, Kanagawa, 242-8502, Japan
inokuchi@jp.ibm.com
[2] The Institute of Scientific and Industrial Research, Osaka Univ.,
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047 Japan
{washio, motoda}@ar.sanken.osaka-u.ac.jp

Abstract. Data mining to derive frequent subgraphs from a dataset of general graphs has high computational complexity because it includes the explosively combinatorial search for candidate subgraphs and subgraph isomorphism matching. Although some approaches have been proposed to derive characteristic patters from graphs, they limit the graphs to be searched within a specific class. In this paper, we propose an approach to conduct a complete search of various classes of frequent subgraphs in a massive dataset of labeled graphs within practical time, and discuss the relation between the biases and diffusion kernels. The power of our approach comes from the algebraic representation of graphs, its associated operations and well-organized bias constraints to limit the search space efficiently. Its performance has been evaluated using real world datasets, and the high scalability of our approach has been confirmed with respect to the amount of data and the computation time.

1 Introduction

Graph mining which discovers characteristic subgraph patterns embedded in a general graph dataset is an important problem having broad applications. This problem is very difficult to solve in practical time because the search of candidate frequent subgraphs is explosively combinatorial, and includes subgraph isomorphism matching which is known to be NP-complete. For example, WARMR is a system to mine frequent subgraphs based on inductive logic programming (ILP) [3]. It could derive all subgraphs consisting of only a few vertices within tractable time under its application to a raw graph dataset without introducing descriptions of higher level structures.

To alleviate the difficulty of the computation time, some other approaches have introduced certain approximations. The most well known approximation is to apply a greedy search as in SUBDUE [2] and GBI [14, 11]. However, they are not very suitable to many applications requiring a complete search of the results. Another approach is to limit the graphs to be searched within simpler classes. The levelwise version space algorithm proposed by De Raedt et al. limits the search space to frequent paths embedded in the graphs for the tractable computation [4]. However, this is also weak for certain application areas to require the mining of subgraphs but not paths.

To overcome these limitations, some approaches which achieve to mine a complete set of frequent patterns from a massive dataset of graphs or ordered trees have been proposed. Although each method can efficiently discover the patterns, the searched patterns are limited within a specific class. For example, TreeMiner [15] and FREQT [1] can quickly discover all frequent patterns from ordered trees or non-ordered trees. However, they can not be applied to more complex structures such as labeled graphs. On the other hand, AGM [5], FSG [10], and gSpan [13] can mine frequent patterns from a set of graphs. However, they can not efficiently discover frequent patterns of paths and/or trees, because their data structures and their search operations are not dedicated to path and/or tree structure mining.

In this paper, we propose a generic and efficient framework to mine various classes of substructures. By introducing bias for each class of substructure, *e.g.*, connected subgraphs, path structures, ordered subtrees, and general subtrees, to the principle of AGM, the complete search of the frequent substructures of each class is achieved. We call this new framework *Biased Apriori-based Graph Mining (B-AGM)*. We discuss the relation between the biases for frequent subgraph mining and diffusion kernels. We evaluate its performance in terms of the required computation time under various amounts of real world datasets.

2 Problem Definitions and AGM

We use the basic principle of AGM algorithm in our extended framework. By applying some specific biases to the algorithm, our B-AGM discovers frequent subgraphs of various classes. In this section, we explain the problem definitions and the AGM framework. In the next section we propose the biases which enable the graph mining of various classes.

2.1 Graph structured data

A graph G consists of 4 sets, the set of vertices V, the set of edges E, the set of vertex labels L_V, and the set of edge labels L_E. When V, E, L_V and L_E are provided as $V = \{v_1, v_2, \dots, v_k\}$, $E = \{e_h = (v_i, v_j) | v_i, v_j \in V\}$, $L_V = \{lb(v_i) | \forall v_i \in V\}$, and $L_E = \{lb(e_h) | \forall e_h \in E\}$, respectively, the graph G is mathematically defined as $G = (V, E, L_V, L_E)$, where multiple vertices can have an identical label, and multiple edges also can have. For the convenience of description, the tuples of G are represented as $V(G)$, $E(G)$, $L_V(G)$, and $L_E(G)$, respectively. The number of vertices, $|V(G)|$, is called the *size* of the graph G.

A graph structure can be expressed by using an adjacency matrix. Let $num(lb(e_h))$ and $num(lb(v_i))$ be natural numbers assigned to an edge label $lb(e_h)$ and a vertex label $lb(v_i)$, respectively for calculation efficiency. Given a graph $G = (V, E, L_V, L_E)$, (i, j)-element $x_{i,j}$ of an adjacency matrix X_k representing the graph G of size k is defined as follows.

$$x_{i,j} = \begin{cases} num(lb(e_h)) & \text{if } e_h = (v_i, v_j) \in E(G) \\ 0 & \text{if } (v_i, v_j) \notin E(G) \end{cases},$$

where $i, j \in \{1, \cdots, k\}$. The vertex corresponding to the i-th row (i-th column) of an adjacency matrix is called the i-th vertex. The graph structure represented by an adjacency matrix X_k is denoted as $G(X_k)$. The representation of the adjacency matrix on an identical graph differs depending on the assignment of rows and columns to the vertices of the graph. To remove this ambiguity, a *canonical form* of adjacency matrices representing an identical graph is introduced.

To mathematically define the canonical form of a graph and for the efficient matrix handling, an adjacency matrix X_k is represented and processed in form of the following code[3].

$$code(X_k) = x_{1,2}x_{1,3}x_{2,3}x_{1,4} \cdots x_{k-2,k}x_{k-1,k},$$

where it is a concatenation of (i, j)-element $x_{i,j}$. Furthermore, the $CODE$ including the vertex labels is defined as

$$CODE(X_k) = num(lb(v_1)) \cdots num(lb(v_k))code(X_k),$$

where it is a concatenation of $num(lb(v_i))$s and $code(X_k)$. The canonical form of the adjacency matrices representing a graph is the unique matrix having the maximum (minimum) $CODE$.

Given a set GD of graph structured data, the support $sup(G_s)$ of a subgraph G_s is defined as the ratio of the number of graphs including G_s to the total number of graph data in the dataset GD. The subgraph having the support more than or equal to a specified *minimum support* is called a *frequent subgraph*. A frequent subgraph of size k is called a frequent k-subgraph. When the dataset of graph structured data and the minimum support are given as input, the data mining of graph structure is to derive all frequent subgraphs in the dataset [6].

2.2 AGM algorithm

We proposed an approach named AGM (Apriori-based Graph Mining) algorithm in which the knowledge representation and the search operations are highly dedicated to the graph structured data mining [5, 7]. The AGM algorithm is so generic that it can discover not only frequent connected subgraphs but also frequent unconnected subgraphs. AGM framework derives all frequent subgraphs based on the monotonic property of the support measure. Based on this property, frequent subgraphs are derived stepwisely in ascending order of their sizes beginning with frequent 1-subgraphs. Fig. 1 is the outline of the AGM algorithm. First, a 1×1 adjacency matrix representing a vertex is generated for every vertex label appearing in the dataset, and they are substituted for C_1. Next, the support of each candidate frequent subgraph is calculated by accessing the database, and only the frequent subgraphs are retained in a set F_1. Subsequently, Generate-Candidate function generates the candidate frequent subgraphs of size $k + 1$ from frequent k-subgraphs in F_k, and they are substituted for C_{k+1}. This process is repeated until C_k becomes empty. Finally, the complete set of frequent subgraphs is provided.

[3] This definition can be expanded to directed graph [7].

Generate-Candidate function consists of three parts, *i.e.*, join operation, subgraph checking operation, and canonical form derivation. In join operation, adjacency matrices of the candidate frequent subgraphs of size $k+1$ are generated by joining two adjacency matrices of frequent k-subgraphs in F_k. Given two adjacency matrices X_k and Y_k in F_k, they are joinable if and only if all of the following conditions are fulfilled.
Condition 1 X_k and Y_k are identical except the k-th row and the k-th column, *i.e.*,

$$X_k = \begin{pmatrix} X_{k-1} & \mathbf{x}_1 \\ \mathbf{x}_2^T & 0 \end{pmatrix}, Y_k = \begin{pmatrix} X_{k-1} & \mathbf{y}_1 \\ \mathbf{y}_2^T & 0 \end{pmatrix}, \text{and}$$

$$lb(v_i \in V(G(X_k))) = lb(v_i \in V(G(Y_k))), \ (i = 1, \cdots, k-1).$$

Condition 2 X_k is the canonical form of $G(X_k)$. This avoids the redundant join.
Condition 3 $CODE(X_k) \geq CODE(Y_k)$ is fulfilled.[4].

If X_k and Y_k are joinable, their join operation is defined as follows.

$$Z_{k+1} = \begin{pmatrix} X_{k-1} & \mathbf{x}_1 & \mathbf{y}_1 \\ \mathbf{x}_2^T & 0 & z_{k,k+1} \\ \mathbf{y}_2^T & z_{k+1,k} & 0 \end{pmatrix},$$

$$lb(v_i \in V(G(Z_{k+1}))) = lb(v_i \in V(G(X_k))), \ (i = 1, \cdots, k), \text{and}$$
$$lb(v_{k+1} \in V(G(Z_{k+1}))) = lb(v_k \in V(G(Y_k))).$$

X_k and Y_k are called the first generator matrix and the second generator matrix of Z_{k+1}, respectively. Two elements $z_{k,k+1}$ and $z_{k+1,k}$ of Z_{k+1} are not determined by X_k and Y_k. The possible graph structures for $G(Z_{k+1})$ are those wherein there is a labeled edge or wherein there is no edge between k-th vertex and $\{k+1\}$-th vertex. Then, $(|L_E(E)| + 1)$ adjacency matrices with $z_{k,k+1} = z_{k+1,k}$ are generated, where $|L_E(E)|$ is the number of edge labels. The adjacency matrix generated under the above conditions is called a *normal form*.

For the necessary condition of $G(Z_{k+1})$ being a frequent subgraph, all induced subgraphs of $G(Z_{k+1})$ must be frequent subgraphs. The application of this condition reduces the candidates. This is done though subgraph checking in $G(Z_{k+1})$ which is described in detail in the literature [6]. After generating the matrices of candidate subgraphs, a database is accessed to calculate their supports. However, the canonical form of these matrices must be identified to collect all counts of the graph, since multiple normal form matrices can represent an identical graph. This is done through canonical form derivation which is described in detail in the literature [5, 7].

3 Extension to Mine Various Class Structures

The original AGM performs the complete mining of the frequent and general subgraphs. However, a version of AGM contains a bias to derive the frequent

[4] In the case that a canonical form is the minimum $CODE$, $CODE(X_k) \leq CODE(Y_k)$ is fulfilled.

```
// $GD$ is a database consisting of graph structured data.
// $F_k$ and $C_k$ is a set of adjacency matrix of frequent and candidate $k$-graphs, resp.
// $minsup$ is minimum support.
 0) Main($GD, minsup$){
 1)   $C_1 \leftarrow$ {all adjacency matrices consisting of one vertex};
 2)   $k \leftarrow 1$;
 3)   while($C_k \neq \emptyset$) {
 4)     Count($GD, C_k$);
 5)     $F_k \leftarrow \{c_k \in C_k | sup(G(c_k)) \geq minsup\}$;
 6)     $C_{k+1} \leftarrow$ Generate-Candidate($F_k$);
 7)     $k \leftarrow k + 1$;
 8)   }
 9)   return $\bigcup_k \{f_k \in F_k | f_k$ is canonical$\}$;
10) }
```

Fig. 1. Outline of Algorithm

induced subgraph only [5, 7]. The induced subgraph of a graph G has a subset of the vertices of G and the same edges between pairs of vertices as in G. To limit the search of the frequent subgraph within this class, the following bias has been applied in the past work. When counting frequency of each candidate frequent subgraph, AGM checks whether it is contained in each graph in a database as an induced subgraph.

In the following subsection, we propose further biases which enable the graph mining of various classes based on the AGM framework as depicted in Fig. 2. We call this framework *B-AGM*. A bias for a specific class of the graph structure consists of the dedicated definitions of the canonical form and the join operation. By choosing appropriate bias on the platform of the AGM framework, the complete mining for the frequent subgraphs of the objective class we are seeking for is defined.

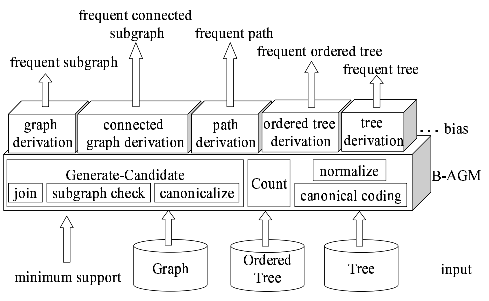

Fig. 2. B-AGM Framework

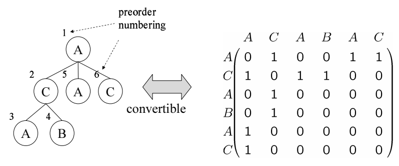

Fig. 3. Tree Coding Method

3.1 Bias for ordered subtree derivation

This bias derives frequent ordered trees included in a forest.

Canonical Form

The definition of canonical form is altered from the original. When the preorder natural numbers which are assigned to the vertices of an ordered tree are consistent with the columns and rows of its adjacency matrix, the matrix is defined as the canonical form of the ordered tree as shown in Fig. 3. This is because the numbering of the preorder in the ordered tree uniquely specifies the topological tree structure. Accordingly, the definition of the preorder tree represents the canonical form. Under this definition, AGM can search only canonical form matrices but not non-canonical form matrices. This extremely enhances the search efficiency. Its proof is omitted due to the space limitation.

Join Operation

The original condition 1 and 2 are retained, and the following new condition 3 and an extra condition 4 are introduced.

Condition 3 In the case that $G(Y_k)$ is connected, $code(X_k) \leq code(Y_k)$ is fulfilled.

Condition 4 $G(X_k)$ is a connected graph[5].

4 Relation between Mining Biases and Diffusion Kernels

In this section, we discuss the relation between our biases for frequent subgraph mining and diffusion kernels. Kernel-based algorithms are paid much attention in the statistical learning community. Konder et. al introduced diffusion kernels for direct use to learning algorithm which was used to quantifying the similarity between vertices in a graph [9]. Diffusion process on a graph for discrete time is modeled as follows. Heat or other substances on each vertex diffuse to its respective neighbors through the corresponding edges with time. Given an undirected graph G, a matrix to determine diffusion process is defined as

$$H_{i,j} = \begin{cases} 1 & \text{for } i \sim j \\ -d_i & \text{for } i = j \\ 0 & \text{otherwise} \end{cases},$$

where d_i is the degree of vertex i, and $i \sim j$ means that there is an undirected edge between i-th and j-th vertices. It is similar to an adjacency matrix. Consider the random field obtained by attaching independent, zero mean variance σ^2 random variables $Z_i(0)$ to each vertex i. The random variables diffuse a fraction α to the neighbor at discrete time steps $t = 1, 2, \cdots$; that is,

$$Z_i(t+1) = Z_i(t) + \alpha \sum_{j \in V : i \sim j} (Z_j(t) - Z_i(t)), \tag{1}$$

where V is a set of vertixes of the graph. Assuming $T(t) = (1 + \alpha H)^t$, $Z(t) = (Z_1(t), \cdots, Z_{|V|}(t))^T$ can be written as $Z(t) = T(t)Z(0)$. An exponential kernel K for a graph is defined as $K = e^{\beta H}$. The (i,j)-element of K is used to quantifying a similarity between i-th and j-th vertices of a graph, which means if structures of neighborhoods of two vertices are similar, quantities of diffusing substances should also be similar after infinity time steps.

[5] Other biases for connected graph, path, and general tree derivations and their corresponding experiments are described in [8].

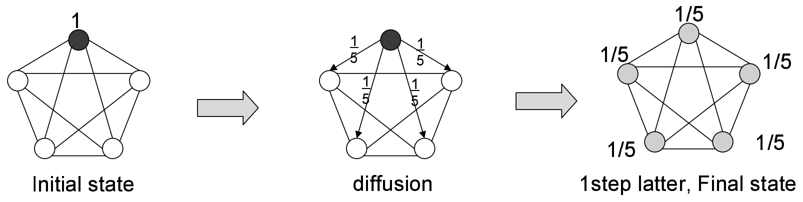

Fig. 4. Diffusion Process

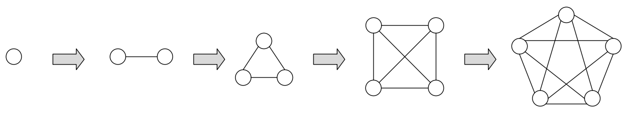

Fig. 5. Candidate Graph Generation

For sake of simplicity, we discuss a complete graph G. Eq. (1) is represented as

$$Z(t+1) = (I + \alpha H)Z(t). \tag{2}$$

Let G be a complete graph with five vertices, and one of them be filled with a certain substance, *i.e.,* $Z_1(0) = 1$, and $Z_i(0) = 0$ for $i = 2, 3, 4, 5$. If it comes to equilibrium in one step, the value of α is $\frac{1}{5}$ according to Eq. (2). It is denoted as $Z(1) = [I + \frac{1}{5}H]Z(0)$. Fig. 4 shows that the $\frac{1}{5}$ of substance diffuses from a vertex filled with the substance to another vertex not containing the substance in one step, and the substance is at equilibrium.

On the other hand, our AGM mines frequent subgraphs in a stepwise manner by expanding vertices. A bias to derive frequent subgraphs which are complete graphs with no vertex and edge labels are defined as follows.

Canonical Form

All matrices generated by the following join operation are canonical forms.

Join Operation

The original condition 1 and 2 are retained, and the following new condition is added. $X_k = Y_k$ and both $z_{k,k+1}$ and $z_{k+1,k}$ of generated matrices are 1.

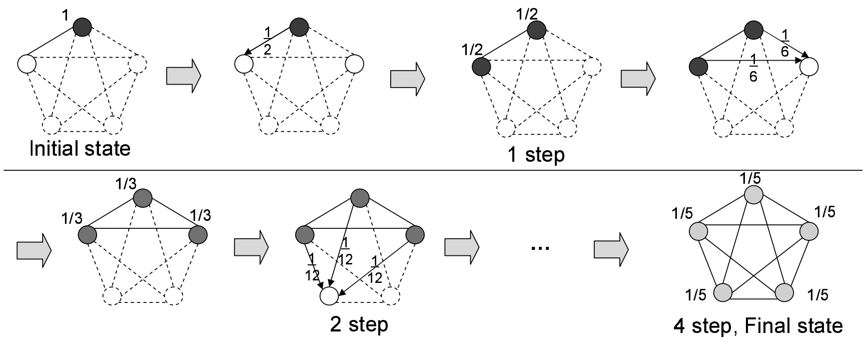

Fig. 6. Diffusion Process (2)

Adjacency matrices of candidate graphs are stepwisely generated as

$$X_1 = (0) \Rightarrow X_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \Rightarrow X_3 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \Rightarrow X_4 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \Rightarrow X_5 = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix},$$

where the number of matrices in each size becomes 1. Assuming A is an adjacency matrix of a complete graph G with no self loops, Eq. (2) is

$$Z(t+1) = [I + \alpha(A - d_i I)]Z(t) = [(1 - \alpha d_i)I + \alpha A]Z(t). \qquad (3)$$

In the case of general diffusion, the substance diffuses to all adjacent vertices. However, we assume that the substance diffuses to one of vertices not containing the substance one by one as shown in Fig. 6. First, a vertex is filled with the substance in the initial state as well as Fig. 4. Next, a vertex is added to the vertex with the substance. Subsequently, the substance in the vertex diffuses to the added vertex in the one step. The substance gradually diffuses to the other vertexes. Finally, all of the vertices are filled with 1/5 of the substance. H, α, and d_i change with time in our case, although they are uniform in the general case. Therefore, we rewrite Eq. (2) and (3) as

$$Z(t+1) = [I + \alpha(t)H(t)]Z(t) = [\{1 - \alpha(t)d_i(t)\}I + \alpha(t)A(t)]Z(t). \qquad (4)$$

In concrete terms, $\alpha(t)$, $d_i(t)$ and $A(t)$ is denoted as

$$\alpha(0) = \frac{1}{2},\ \alpha(1) = \frac{1}{3},\ \alpha(2) = \frac{1}{4},\ \alpha(3) = \frac{1}{5},\ d_i(0) = 1,\ d_i(1) = 2,\ d_i(2) = 3,\ d_i(3) = 4,$$

$$A(0) = \begin{pmatrix} 0&1&0&0&0 \\ 1&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{pmatrix},\ A(1) = \begin{pmatrix} 0&1&1&0&0 \\ 1&0&1&0&0 \\ 1&1&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \end{pmatrix},\ A(2) = \begin{pmatrix} 0&1&1&1&0 \\ 1&0&1&1&0 \\ 1&1&0&1&0 \\ 1&1&1&0&0 \\ 0&0&0&0&0 \end{pmatrix},\ A(3) = \begin{pmatrix} 0&1&1&1&1 \\ 1&0&1&1&1 \\ 1&1&0&1&1 \\ 1&1&1&0&1 \\ 1&1&1&1&0 \end{pmatrix}.$$

In general, $\alpha(t) = \frac{1}{t+2}$, $d_i(t) = t + 1$. Hence, Eq. (4) for a complete graph is

$$Z(t+1) = \frac{1}{t+2}[I + A(t)]Z(t). \qquad (5)$$

The substance diffuses to one of vertices not containing the substance one by one in our case, although the substance diffuses to all adjacent vertices in the case of general diffusion. Since to limit the search space to a specific class of frequent subgraphs by a bias corresponds to constraining a matrix H in the diffusion kernel, we can conclude that our biases for frequent subgraph mining are related to diffusion kernels. There is a systematic regularity for the complete graph, and furthermore such systematic regularities for other classes of graphs may similarly exist. If Eq. (5) is generalized for the other classes of the graphs, approximate subgraph matching instead of exact subgraph matching may be able to be applied, when the support values of candidate graphs are calculated. We will investigate other cases that derived subgraphs are not complete graphs in our future plan.

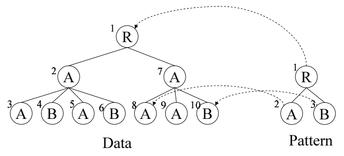

Fig. 7. Examples of Ordered Tree Data & Pattern

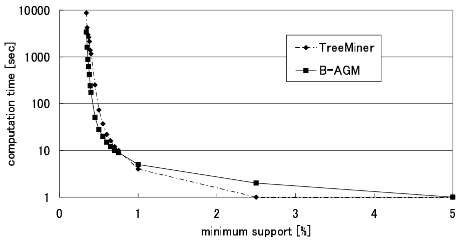

Fig. 8. Minimum Support vs. Computation Time

5 Experiment and Discussion

IBM PC 300PL with Windows 2000 was used for the experiments, with a Pentium III-667 MHz and 192 MB of main memory. Zaki presented experimental results of substructure discovery from a set of logs files over 1 month at the RPI computer science department's web site [15]. After the preprocessing, the dataset had 595,691 user browsing subtree with 13,361 unique labels (web pages). We used the same data provided by Zaki on our experiment and recalculated with the same machine to compare our approach with TreeMiner. TreeMiner can find frequent patterns embedded in tree dataset. It is defined that pattern P discovered by TreeMiner is embedded in tree data D if and only if two vertices in a branch in P are on the same path from the root to a leaf in D. For example, Fig. 7, an ordered tree in the right is embedded in the data. We used the same definition in the comparison of the result to obtain equivalent result. The function to count frequency is changed to adjust to the definition of frequent patterns of TreeMiner in B-AGM. Fig. 8 shows computation time for various minimum support values. Fig. 8 shows that B-AGM is comparable to TreeMiner.

FSG [10], gSpan [13], MolFea [4], TreeMiner [15] and FREQT [1] take a complete search strategy similarly to AGM. FREQT and TreeMiner require $O(n)$ memory to store tree data, where n is the number of vertices in the tree. Although our approach requires $O(n^2)$ memory for the storage, FREQT and TreeMiner can not be applied to graph mining problems. Our proposed method can conduct a complete search of various classes of frequent subgraphs. The advantage of our approach is very flexible and quickly adaptable for new classes of patterns in graph mining. AGM with each bias can derive the complete result within practical time for each class of problem respectively where the performance is far faster than or comparable to the other approaches which are optimized within a specific class [8]. This is because the representation of graphs and its associated operations are well-organized and dedicated to the graph mining. Our method is considered to be highly practical in many applications.

6 Conclusion

We proposed a generic framework for the data mining of graph structures. By introducing additional biases, it can easily derive various types of frequent substructures. We evaluated its performance in terms of the required computation time for the real world datasets. The wide cover on problem classes and the computational efficiency which are superior to the other approaches have been confirmed. We will investigate the relation between our biases for frequent subgraph mining and diffusion kernels in other cases that derived subgraphs are not complete graphs in our future plan.

Acknowledgement

We would like to thank Professor Mohammed Zaki at Rensselaer Polytechnic Institute and Professor Hiroki Arimura and Mr. Tatsuya Asai at Kyushu University for their help and advice.

References

1. Asai, T., Abe, K., Kawasoe, S., Arimura, H., Sakamoto, H. & Arikawa, S. Efficient Substructure Discovery from Large Semi-Structured Data. *Proc. of SIAM Int'l. Conf. on Data Mining*, pp. 158–174, (2002).
2. Cook, D. & Holder, L. Substructure Discovery Using Minimum Description Length and Background Knowledge. *Journal of Artificial Intelligence Research*, Vol.1, pp. 231–255, (1994).
3. Dehaspe, L., Toivonen, H., & King, R. D. Finding frequent substructures in chemical compounds. *Proc. of the 4th International Conference on Knowledge Discovery and Data Mining* (KDD-1998), pp. 30–36, 1998.
4. De Raedt, L. & Kramer, S. The Levelwise Version Space Algorithm and its Application to Molecular Fragment Finding. *Proc. of Int'l. Joint Conf. on Artificial Intelligence*, pp. 853–859, (2001).
5. Inokuchi, A., Washio, T. & Motoda, H. An Apriori-based Algorithm for Mining Frequent Substructures from Graph Data. *Proc. of European Conf. on Principles and Practice of Knowledge Discovery in Databases*, pp. 13–23, (2000).
6. Inokuchi, A., Washio, T., Nishimura, Y. & Motoda, H. A Fast Algorithm for Mining Frequent Connected Graph. *IBM Research Report* RT0448, (2002), February.
7. Inokuchi, A., Washio, T. & Motoda, H. Complete Mining of Frequent Patterns from Graphs: Mining Graph Data, *Machine Learning*, 50 (3): 321-354, March, 2003,
8. Inokuchi, A., Washio, T., & Motoda, H. A General Framework for Mining Frequent Patterns in Structures. *IBM Research Report* RT0513, (2003), February.
9. Konder, R. I., & Lafferty, J. Diffusion kernels on graphs and other discrete input spaces. *Proc. of Int'l Conference on Machine Learning* (ICML-2002), 2002.
10. Kuramochi, M., & Karypis, G. Frequent Subgraph Discovery. *Proc. of the 1st International Conference on Data Mining* (ICDM-2001), pp.313–320, 2001.
11. Motoda, H. & Yoshida, K. Machine Learning Techniques to Make Computers Easier to Use. *Proc. of Int'l. Joint Conf. on Artificial Intelligence*, Vol. 2, pp. 1622–1631, 1997.
12. Yan, X. & Han, J. gSpan: Graph-Based Substructure Pattern Mining. *Proc. of the 3rd International Conference on Data Mining*, (ICDM-2002), pp.721–724, 2002.
13. Yoshida, K., & Motoda, H. CLIP: Concept Learning from Inference Patterns. *Artificial Intelligence*, Vol. 75, No. 1 pp. 63–92, 1995.
14. Zaki, M. Efficiently Mining Frequent Trees in a Forest. *Proc. of the 8th International Conference on Knowledge Discovery and Data Mining* , pp. 71–80, 2002.