An Efficient Algorithm for Mining Frequent Substructures from Graph Data

Akihiro Inokuchi, Takashi Washio and Hiroshi Motoda

I.S.I.R., Osaka University
8-1, Mihogaoka, Ibarakishi, Osaka, 567-0047, Japan
Phone: +81-6-6879-8541 Fax: +81-6-6879-8544
{inokuchi, washio, motoda}@sanken.osaka-u.ac.jp

Abstract. Graph structured data are abound in many practical fields such as molecular structures of chemical compounds and information flow patterns in the internet. In many cases, their intrinsic regularities are not captured by the features of individual vertices and edges but by only the features of substructures and their relations embedded in the graphs. Mining such regularities has been tackled by some others. However, previous approaches have not been efficient enough to handle real-world problems such as carcinogenesis predictions for chemical compounds, and thus the obtained knowledge is limited for the cases of quite simple regularities. This paper proposes a novel approach to more efficiently mine the association rules among the frequently appearing substructures in a given graph data set. A transaction is a graph involving the topological information among vertices and edges, and is represented by an adjacency matrix. The algorithm of the basket analysis has been extended to handle the adjacency matrices (that include items as a special case). Novel principles are introduced to uniquely order the adjacency matrices and to efficiently find isomorphism among them. The proposed approach finds association rules representing the co-occurrence of substructures in the transactions. Its performance has been evaluated for the artificial simulation data and the carcinogenesis data of Oxford University and NTP. Its high efficiency has been confirmed for the size of a real-world problem.

1 Introduction

Mining knowledge from structured data is a major research topic in recent data mining study. The approach proposed by Agrawal and Srikant for mining sequential patterns was one of the initiating works in this field [1]. Since then several approaches have been proposed from different angles for sequential or structural data. Mannila et al. proposed an approach to mine frequent episodes from sequences [2]. Shinatani and Kitsuregawa devised a fast mining algorithm for sequential data using parallel processing [3]. Srikant et al. used taxonomy hierarchy as background knowledge to mine association rules [4].

Sequence, association and taxonomy are typical data structures that appear in real-world domains, but "graph structure" also frequently appears in real-world data such as web links and chemical compounds structures. In the field of chemistry, CASE and MultiCASE systems have been often used to discover characteristic substructures of chemical compounds [5], [6]. Though these systems can efficiently find the substructures, the class of the substructures is limited to the no-branching atom sequences. Wang and Liu proposed the mining of wider class of substructures which are subtrees called schemas [7]. Though the proposed algorithm is very efficient to mine frequent schemas from massive data, the mining patterns are still limited to acyclic graphs. To mine characteristic patterns having general graph structures, the propositional classification techniques, e.g., C4.5, the regression tree techniques, e.g., M5, and the inductive logic programming (ILP) techniques have been applied in the carcinogenesis predictions of chemical compounds [8], [9]. However, these approaches can discover only limited types of characteristic substructures, because the graph structures must be pre-characterized by some specific features and/or ground instances of predicates (*e.g.*, a benzene ring is involved in the compound). This data preprocessing is inevitable for the propositional classification and regression tree techniques, since they can handle only feature tables. This preprocessing is also necessary for ILP techniques to reduce the computational complexity in the mining process.

Recently, a technique to mine the frequent substructures characterizing the carcinogenesis of chemical compounds has been proposed without requiring any conversion of substructures to specific features by Dehaspe et al. [10]. They used the ILP framework combined with levelwise search to minimize the access frequency to the database [11]. Since the efficiency achieved by this approach is much better than the former ILP approaches, some new discovery of substructures characterizing carcinogenesis was expected. However, the full search space was still so large that the search had to be limited within the 6th level where the substructures are represented with 6 predicates at maximum, and they reported that significant substructures have not been obtained within the search level. Some other researches have also developed the techniques to mine the frequent substructures in graph data. The graph-based induction (GBI) is an approach to seek the frequent patterns by iteratively chunking the vertex pairs that frequently appear [12], [13]. SUBDUE is another approach to seek the characteristic graph patterns to efficiently compress the original graph in terms of MDL principle [14]. These approach does not face the severe compu-

tational complexity. However, it may miss some significant patterns, since the search strategy is greedy. The use of the standard basket analysis has also been proposed [15]. Each graph is transformed into an itemset consisting of labeled vertices and edges, then the basket analysis is applied. Though this approach efficiently finds all frequent substructures and the association rules among them, it works for only a limited case where all vertices in a graph are distinct, *i.e.*, all vertices have different labels, which is not the case for chemical compounds analysis.

Though the task tackled by these works involves the problem of deciding graph isomorphism which is known to be NP-complete, each work mines some characteristic graph substructures by introducing the limitations on the search space and/or the class of substructures. The objective of this paper is to propose a novel approach to mine the frequent substructures and the association rules from the general class of graph structured data inmore efficient manner than the preceding work. Moreover, we assess the performance of the approach for the artificially simulated data and also for the carcinogenesis data of Oxford University and National Toxicological Program (NTP) [16].

2 Principle of Mining Graph Substructures

In the field of mathematical graph theory, the problem to check the isomorphism between two given graphs has been extensively studied [17], [18]. Two main methods have been used to solve this problem. The first method is called the "brute force" in which a direct correspondence between the two graphs is searched. It repeats to take vertices from each of the two graphs and check if the adjacency between the vertices matches. If all adjacencies are matched, the two graphs are known to be isomorphic. However, the computational complexity required for this algorithm is non-polynomial order of the vertex number of the graphs. The other method is to compute a unique canonical code for the two graphs and compare them. If they are isomorphic, the codes for the two graphs match each other. This approach can efficiently reduce the computational complexity, if a simple method to determine the canonical code of a graph can be obtained.

However, the methods studied in the mathematical graph isomorphism problem are not directly applicable to our case, because the mathematical techniques adopted there are only to check if the two given graphs are isomorphic. To search all frequent subgraphs in a given graph data set, the isomorphism among all subgraphs of all data must be checked in a naive framework. Since the selection of the subgraphs is combinatorial, we face the computational explosion even if we use the efficient isomorphism checking method for each subgraph pair. To overcome this difficulty, we have proposed an idea to introduce the mathematical graph representation of "adjacency matrix" having binary values of the elements and to combine it with an efficient levelwise search of the frequent canonical matrix code [19]. The levelwise search is based on the extension of the Apriori algorithm of the basket analysis [20]. However, this approach can not directly handle the

graph data in which the edges have label information. In this paper, the representation of the adjacency matrix is further extended to describe more general class of the graphs by introducing multiple values in the matrix elements. Both vertices and edges can have labels. The edges can be either directed or undirected. The graph can have loops (including self-loops), and can consist of unconnected multiple subgraphs. A novel technique to reduce the search space for finding an isomorphic subgraph in a large graph is also introduced. We call our proposing approach as "Apriori-based Graph Mining", *ABG* for short.

2.1 Representation of Graph Structures

A graph in which the vertices and edges have labels is mathematically defined as follows.

Definition 1 (Graph having Labels) *Given a set of vertices* $V(G) = \{v_1, v_2, ..., v_k\}$, *a set of edges connecting some vertex pairs in* $V(G)$; $E(G) = \{e_h = (v_i, v_j) | v_i, v_j \in V(G)\}$, *a set of vertex labels* $L(V(G)) = \{lb(v_i) | \forall v_i \in V(G)\}$ *and a set of edge labels* $L(E(G)) = \{lb(e_h) | \forall e_h \in E(G)\}$, *then a graph* G *is represented as*

$$G = (V(G), E(G), L(V(G)), L(E(G))).$$

This graph is represented by an adjacency matrix which is a very well known representation in mathematical graph theory [17].

Definition 2 (Adjacency Matrix) *Given a graph* $G = (V(G), E(G), L(V(G)), L(E(G)))$, *the adjacency matrix* X *has the following* (i, j)*-element,* x_{ij},

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

where $num(lb)$ *is an integer arbitrarily assigned to a label value* lb.

This transformation from G to X does not require much computational effort.

Definition 3 (Size of a Graph) *The "size" of a graph* G *is the number of vertices in* $V(G)$, *i.e.,* k *in Definition 1.*

The adjacency matrix of a graph whose size is k is noted as X_k, and the graph as $G(X_k)$.

Definition 4 (Graph Transaction and Graph Data) *A graph* $G = (V(G), E(G), L(V(G)), L(E(G)))$ *is a transaction, and graph data* GD *is a set of the transactions, where* $GD = \{G_1, G_2, ..., G_n\}$.

The labels of the edges and the vertices of an example graph transaction depicted in Figure 1 (a) are defined in the following manner.

$$lb(v_1) = V_1, lb(v_2) = V_2, lb(v_3) = V_1$$
$$lb(e_1) = E_1, lb(e_2) = E_1, lb(e_3) = E_2$$

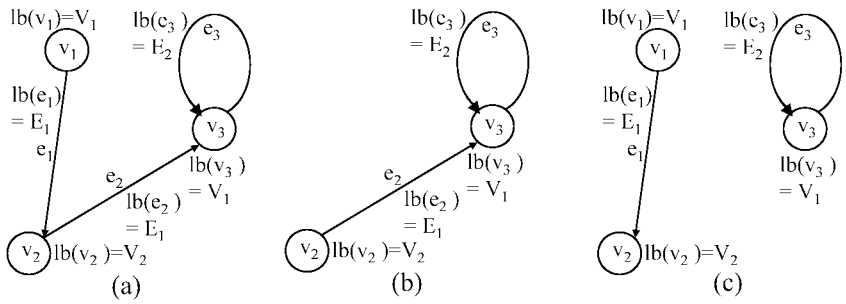


Fig. 1. A graph example.

Given the integer assignment to label values of the edges as $num(lb(e_1)) = num(E_1) = 1$ and $num(lb(e_2)) = num(E_2) = 2$, its adjacency matrix is represented as follows.

$$\begin{array}{c} \\ V_1 \\ V_2 \\ V_1 \end{array} \begin{array}{c} \begin{array}{ccc} V_1 & V_2 & V_1 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 2 \end{pmatrix} \end{array}. \tag{1}$$

The representation of the adjacency matrix depends on the assignment of each vertex to the i-th row (i-th column). To reduce the variants of the representations and increase the efficiency of the code matching described later, the vertices are sorted according to the numbers of their labels.

Definition 5 (Vertex-sorted Adjacency Matrix) *The adjacency matrix X_k of the graph $G(X_k)$ is vertex-sorted if*

$$num(lb(v_i)) \leq num(lb(v_{i+1})) \text{ for } i = 1, 2, ..., k-1.$$

The vertices v_2 and v_3 of Eq.(1) are permuted in this ordering, and the vertex-sorted adjacency matrix is represented by

$$\begin{array}{c} \\ V_1 \\ V_1 \\ V_2 \end{array} \begin{array}{c} \begin{array}{ccc} V_1 & V_1 & V_2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 1 & 0 & 0 \end{pmatrix} \end{array}.$$

The above explanation is for directed graphs, but the similar method is applied to undirected graphs. The difference from directed graphs is that the adjacency matrix becomes symmetric. Assuming that the edges of the graph in Figure 1 do not have a direction, the vertex-sorted adjacency matrix becomes as

$$\begin{array}{c} \\ V_1 \\ V_1 \\ V_2 \end{array} \begin{array}{c} \begin{array}{ccc} V_1 & V_1 & V_2 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 \\ 0 & 2 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array}.$$

In the standard basket analysis, items within an itemset are kept in lexicographic order [20]. This enables an efficient control of the generation of candidate itemsets. However, the vertex-sorted adjacency matrices do not have such lexicographic order. Thus, a coding method of the adjacency matrices need to be introduced.

Definition 6 (Code of Adjacency Matrix) *In case of an undirected graph, the code* $code(X_k)$ *of a vertex-sorted adjacency matrix* X_k*;*

$$X_k = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,k} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,k} \\ x_{3,1} & x_{3,2} & x_{3,3} & \cdots & x_{3,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{k,1} & x_{k,2} & x_{k,3} & \cdots & x_{k,k} \end{pmatrix},$$

is defined as

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

where the digits are obtained by scanning the elements along the columns at the upper triangular part of X_k*. In case of a directed graph, it is defined as*

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

where the digits are obtained similarly to the undirected case, but the diagonally symmetric element x_{ji} *is added after each* x_{ij} *when* $i \neq j$*.*

The upper left elements of the matrix locate at higher digits in both coding methods. The ordering of the adjacency matrices based on this code efficiently reduces the search space in the levelwise search as shown later.

The method proposed in this paper discovers substructures frequently appearing in the graph transaction data GD. The rigorous definition of the substructure is given as follows.

Definition 7 (Induced Subraph) *Given a graph* $G = (V(G), E(G), L(V(G)), L(E(G)))$*, an induced subgraph of* G*,* $G_s = (V(G_s), E(G_s), L(V(G_s)), L(E(G_s)))$*, is a graph satisfying the following conditions.*

$$V(G_s) \subset V(G), E(G_s) \subset E(G),$$

$$\forall u, v \in V(G_s), (u, v) \in E(G_s) \Leftrightarrow (u, v) \in E(G).$$

When G_s *is an induced subgraph of* G*, it is denoted as* $G_s \subset G$*.*

For instance, the graph depicted in Figure 1 (b) is an induced subgraph of (a), but (c) is not, because the edge e_2 in (a) is not present in (c).

Furthermore, the two indecies which are identical to the definitions of "support" and "confidence" in the basket analysis are defined.

Definition 8 (Support and Confidence) *Given a graph G_s, the support of G_s is defined as*

$$sup(G_s) = \frac{number\ of\ graph\ transactions\ G\ where\ G_s \subset G \in GD}{total\ number\ of\ graph\ transactions\ G \in GD}.$$

Given two induced subgraphs G_b and G_h, the confidence of the association rule $G_b \Rightarrow G_h$ is defined as

$$conf(G_b \Rightarrow G_h) = \frac{number\ of\ graphs\ G\ where\ G_b \cup G_h \subset G \in GD}{number\ of\ graphs\ G\ where\ G_b \subset G \in GD}.$$

If the value of $sup(G_s)$ is more than a threshold value $minsup$, G_s is called as "frequent induced subgraph".

2.2 Algorithm

candidate generation Similarly to the Apriori algorithm, the candidate generation of the frequent induced subgraph is made by the levelwise search in terms of the size of the subgraph. Let X_k and Y_k be vertex-sorted adjacency matrices of two frequent induced graphs $G(X_k)$ and $G(Y_k)$ of size k, where the vertices have been lexicographically ordered according to their label values. If both $G(X_k)$ and $G(Y_k)$ have equal elements of the matrices except for the elements of the k-th row and the k-th column, then they are joined to generate Z_{k+1}.

$$X_k = \begin{pmatrix} X_{k-1} & \boldsymbol{x}_1 \\ \boldsymbol{x}_2^T & x_{kk} \end{pmatrix}, Y_k = \begin{pmatrix} X_{k-1} & \boldsymbol{y}_1 \\ \boldsymbol{y}_2^T & y_{kk} \end{pmatrix},$$

$$Z_{k+1} = \begin{pmatrix} X_{k-1} & \boldsymbol{x}_1 & \boldsymbol{y}_1 \\ \boldsymbol{x}_2^T & x_{kk} & z_{k,k+1} \\ \boldsymbol{y}_2^T & z_{k+1,k} & y_{kk} \end{pmatrix} = \left(\begin{array}{cc|c} & & \boldsymbol{y}_1 \\ \multicolumn{2}{c|}{X_k} & z_{k,k+1} \\ \hline \boldsymbol{y}_2^T & z_{k+1,k} & y_{kk} \end{array}\right), \tag{2}$$

where X_{k-1} is the adjacency matrix representing the graph whose size is $k-1$, $\boldsymbol{x}_i$ and $\boldsymbol{y}_i (i = 1, 2)$ are $(k-1) \times 1$ column vectors. X_k is called the "first matrix" and Y_k the "second matrix". The following relations hold among the vertex-sorted adjacency matrices X_k, Y_k and Z_{k+1}.

$$lb(v_i; v_i \in V(G(X_k)) = lb(v_i; v_i \in V(G(Y_k)) = lb(v_i; v_i \in V(G(Z_{k+1}))),$$

$$\begin{aligned} lb(v_i; v_i \in V(G(X_k)) &\leq lb(v_{i+1}; v_{i+1} \in V(G(X_k)), \\ lb(v_k; v_k \in V(G(X_k)) &= lb(v_k; v_k \in V(G(Z_{k+1})), \\ lb(v_k; v_k \in V(G(Y_k)) &= lb(v_{k+1}; v_{k+1} \in V(G(Z_{k+1})), \\ lb(v_k; v_k \in V(G(X_k)) &\leq lb(v_k; v_k \in V(G(Y_k)). \end{aligned} \tag{3}$$

Here, $i = 1, \cdots, k-1$. $z_{k,k+1}$ and $z_{k+1,k}$ are not determined by X_k and Y_k. Each can take every integer value $num(lb)$ corresponding to each edge label lb or 0 corresponding to the case that no edge exists between v_k and v_{k+1}. In case

of an undirected graph, $z_{k,k+1}$ and $z_{k+1,k}$ must have an identical value. Note that when the labels of the k-th vertices v_k of $G(X_k)$ and $G(Y_k)$ are the same, exchanging X_k and Y_k (*i.e.*, taking Y_k as the first matrix and X_k as the second matrix), produces redundant adjacent matrices. In order to avoid this redundant generation, the two adjacency matrices are joined only when Eq.(4) is satisfied. The vertex-sorted adjacency matrix generated under this condition is called a "normal form".

$$code(\text{the first matrix}) \leq code(\text{the second matrix}) \tag{4}$$

In the standard basket analysis, the $(k+1)$-itemset becomes a candidate frequent itemset only when all the k-sub-itemsets are confirmed to be frequent itemsets. Similarly, the graph G of size $k+1$ is a candidate of frequent induced subgraphs only when all adjacency matrices generated by removing from the graph G the i-th vertex v_i $(1 \leq i \leq k+1)$ and all its connected links are confirmed to be frequent induced subgraphs of the size k. As this algorithm generates only adjacency matrices of the normal form in the earlier (smaller) k-levels, if the adjacency matrix of the graph generated by removing the i-th vertex v_i is non-normal form, it must be transformed to a normal form to check if it matches one of the normal form matrices found earlier.

An adjacency matrix of a non-normal form is transformed into a normal form by reconstructing the matrix structure in a bottom up manner. The procedure is depicted in the example of Figure 2. It shows how the non-normal adjacency matrix X_4 is transformed to a normal form. The numbers below the adjacency matrices in this figure show the corresponding codes. It starts with the adjacency matrices representing the induced subgraphs of X_4 consisting of one vertex shown at the level of (A) in Figure 2. Which matrices to join are limited to the pairs of the matrix having the minimum code and the others according to the constraint Eq.(4) of the normal form. In this case, we arbitrary choose a matrix pair of v_1 and the others as shown at (B) in the figure, since all matrices have an identical code. The values which can not be determined when joining, for example (1,2)-element and (2,1)-element of the matrix consisting of v_1 and v_2, are taken from x_{12} and x_{21} of X_4 as indicated at (C) to maintain the isomorphism with the original X_4. Next, matrices whose graphs have two nodes are joined as shown at (D). Because the leftmost matrix is one of the matrices having the minimum code, this becomes the first matrix, and the others become the second matrices as shown at (E). After joining these matrices, (1,3)-element and (3,1)-element are determined by X_4, and we obtain two 3×3 matrices which is shown at (F). Because the code 000010 of the right matrix is less than that of the left, the right matrix becomes the first matrix and the other the second matrix. This process continues until a normal form having the same size with the original X_4 is found. As each level of the reconstruction follows the constraints of Eq.(2),(3) and (4), the normal form of the original matrix is obtained. Because this reconstruction is the set of permutations of the raws and the columns of the original matrix X_k, the normal form X'_k has the realation of

$$X'_k = (T_k)^T X_k T_k$$

where T_k is the transformation matrix.

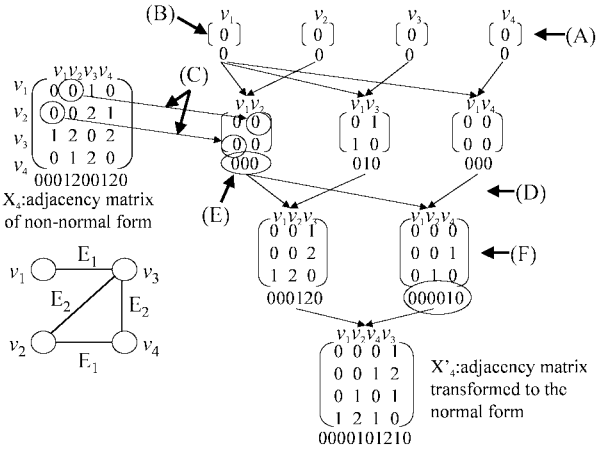


Fig. 2. Transformation into normal form.

canonical from After all candidate induced subgraphs are derived, the support value of each candidate is counted in the database. However, the normal form representation is in general not unique for a graph. For instance, the following two matrices which are both normal forms represent an identical graph.

$$X_3 = \begin{pmatrix} 0\,0\,1 \\ 0\,0\,1 \\ 1\,1\,0 \end{pmatrix}, Y_3 = \begin{pmatrix} 0\,1\,1 \\ 1\,0\,0 \\ 1\,0\,0 \end{pmatrix}.$$

If the support value is counted for each representation independently, it has to be summed up to obtain the correct support value for the correspondig graph. To perform this summation efficiently, all normal forms for an identical induced subgraph must be indexed.

For this purpose, canonical form is defined for normal adjacency matrices representing an identical induced subgraph, and an efficient method to index each normal form to its canonical form is introduced.

Definition 9 (Canonical Form) *Given a set $NF(G)$ of all normal forms of adjacency matrices representing an identical graph G, its canonical form X_c is defined as X having the minimum code number in $NF(G)$, i.e.,*

$$X_c = arg \min_{X \in NF(G)} code(X).$$

Figure 3 shows the algorithm to transform a given normal form to its canonical form with a transformation matrix.

1) **forall** X_k in a set of candidate frequent graphs
2) $X'_k = X_k$
3) **for**$(m = 1; m \leq k; m++)$ **do begin**
4) **if**($lb(v_k$; vertex of k-th raw (column) of $X_k) =$
$lb(v_m$; vertex of m-th raw (column) of $X_k))$ **then do begin**
5) **if**($code(X'_k) > code((T^m_k S^m_k)^T X_k (T^m_k S^m_k))$ **then do begin**
6) $X'_k = (T^m_k S^m_k)^T X_k (T^m_k S^m_k)$;
7) **if**(the canonical form of X'_k is known) **then do begin**
8) $X'_k = S'^T_k X'_k S'_k$; //where S'_k is the matrix to transform X'_k
in r.h.s. to its canonical form
9) **break**;
10) **end**
11) **end**
12) **end**
13) **end**
14) Canonical form of X_k is X'_k;
15) **end**

Fig. 3. An algorithm to derive canonical form

We assume that all the transformation matrices S_{k-1} to the canonical form from the normal forms of every frequent induced subgraph of size k-1 are known. Let X^m_{k-1} be the matrix obtained by removing the m-th vertex v_m $(1 \leq m \leq k)$ from $G(X_k)$. X^m_{k-1} is transformed to one of its normal forms, X'^m_{k-1}, by the procedure explained in Figure 2, and thus its transformation matrix T^m_{k-1} is known. Furthermore, let S_{k-1} of X'^m_{k-1} be S^m_{k-1}, then the transformed canonical form is represented by $(T^m_{k-1} S^m_{k-1})^T X^m_{k-1} T^m_{k-1} S^m_{k-1}$. The canonical form X_{ck} of X_k and the matrices S^m_k, T^m_k to transform X_k to X_{ck} are obtained from S^m_{k-1}, T^m_{k-1} by the following expressions. The detailed proof of this transformation can be seen in [21].

$$s_{ij} = \begin{cases} s^m_{ij} & 0 \leq i \leq k-1 \text{ and } 0 \leq j \leq k-1, \\ 1 & i = k \text{ and } j = k, \\ 0 & \text{otherwise}, \end{cases}$$

$$t_{ij} = \begin{cases} t^m_{ij} & i < m \text{ and } j \neq k, \\ t^m_{i-1,j} & i > m \text{ and } j \neq k, \\ 1 & i = m \text{ and } j = k, \\ 0 & \text{otherwise}, \end{cases}$$

$$X_{ck} = arg \min_{m=1,\cdots,k} code((T^m_k S^m_k)^T X_k (T^m_k S^m_k)),$$

where s_{ij}, s^m_{ij}, t_{ij} and t^m_{ij} are the elements of matrix S^m_k, S^m_{k-1}, T^m_k and T^m_{k-1} respectively. $T^m_k S^m_k$ which minimize the code is S_k of X_k.

frequency calculation Frequency of each candidate induced subgraph is counted by scanning the database after generating all the candidates of frequent induced subgraphs and obtaining their canonical forms. Figure 4 shows an example to explain the support counting for a directed graph transaction $G(X)$ given in the database. We assume that all the subgraphs of size 2 except the one whose code is 0110 are known to be frequent induced subgraphs, and the set C_3 of the adjacency matrices of canonical form for the candidate frequent induced subgraphs of size 3 have already been obtained. The basic task is to enumerate the adjacency matrices of all subgraphs of size 3 embedded in X and to check if some of them are in C_3. Because the adjacency matrices of normal form cover all subgraphs in $G(X)$, the enumeration is limited to the normal form for efficiency by applying the procedure similar to Figure 2. The procedure starts from level 1 and generates the adjacency matrices of the normal form of size 1. In level 2, since the leftmost adjacency matrix having the code 0110 is not the frequent induced subgraph, this matrix is not further expanded. In addition, for two matrices to be joined, the submatrices obtained by removing the last raw and column respectively from them must be identical. Consequently, only two joins in this level is admitted to generate the adjacency matrices of size 3 of the normal form. Finally, if the subgraphs having the codes 000000100 and/or 020010100 are known to correspond to the canonical forms in C_3 by the aforementioned transformation matrices, the counters of the canonical forms are incremented by 1. Even when there are more than one isomorphic subgraph in the given graph G, the counter is incremented by 1 as the default setting in our program. This is based on the definition of support that is the fraction of the number of graph transactions containing the induced subgraph to all graph transactions in the database. Counting the number of occurences of the identical induced subgraph in the search tree is also straightforward and is optional.

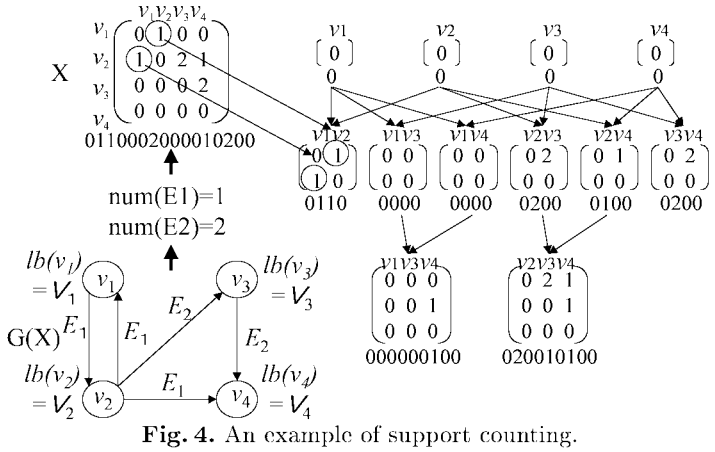


Fig. 4. An example of support counting.

2.3 Implementation

The algorithm explained in the previous subsections is implemented using a trie data structure as depicted in Figure 5. This is an example of undirected graphs that have two vertex labels V_1, V_2 and a unique edge label E_1. A node of the trie corresponds to a matrix of the normal form. The upper literal and the lower numeral in each node show respectively the vertex label sequence and the code of the adjacency matrix. The depth of the trie corresponds to the size of the subgraphs, and for each node in the trie its parent node is the first matrix to generate the matrix of the node. In this trie structure, the adjacency matrices are generated primarily in the lexicographic order of the literals and secondarily in the ascending order of the code numerals. This efficiently reduces the search space of the join, since the first matrix must be joined with only the second matrix locating in the right hand side of the same level.

First, the support values of the graphs of size 1 is calculated in the root, and these are joined using the method described in the previous subsection. Next, each vertex of the graph constructed by the join operation is removed in sequence, and the resulting graphs are checked to see if all of them are frequent induced subgraphs. When all of them are frequent induced subgraphs, the joined graph is considered to be a candidate frequent induced subgraph. Then the canonical form is obtained for each candidate and its support value is counted in the levelwise manner by accessing the database. If the value exceeds the threshold *minsup*, the subgraph is a frequent induced subgraph. The above process is repeated stepwisely from the root downward. Note that the candidate generation and the support calculation interleave, and thus the node whose support is below the support threshold is never expanded. Once all frequent induced subgraphs are found, the association rules among them whose confidence values are more than a given confidence threshold are enumerated by using the algorithm similar to the standard basket analysis.

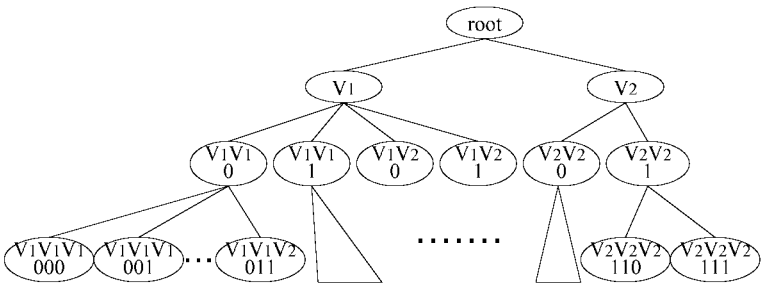


Fig. 5. Trie data structure.

3 Performance Evaluation

The performance of the method was examined using an artificially generated graph transaction data. The machine used is a PC with 400MHz CPU and 128MB main memory. Table 1 summarizes the default parameter values of the test data that are used in the experiments. The size of each transaction is determined by the gaussian distribution with the average $|T|$ and the standard deviation 1. The vertex labels are randomly determined with equal probability. The edges are attached randomly with the probability of p. L basic patterns of the average size $|I|$ are generated, and one of them is randomly overlaid on each transaction. The two groups of the test data, one for the directed graph and the other for the undirected graph, are prepared. The direction of the edges are given randomly in the former group.

Table 1. Default specification of parameter setting in test data.

Parameter	Definition	Base value
D	Nuber of graphs	10,000
$\|T\|$	Average graph size	10
L	Number of basic subgraphs	10
$\|I\|$	Average basic subgraph size	4
N_v	Number of vertex labels	5
N_e	Number of edge labels	1
p	Edge existence probability	50%
$minsup$	Minimum support	10%

Figures 6, 7, 8 and 9 show the results of computation time for different number of transactions, number of vertex labels, minimum support threshold and average transaction size for both directed and undirected graphs, respectively. In every parameter setting, the required computational time and the number of the discovered frequent induced subgraphs are less in the case of directed graph. Because the number of possible subgraph patterns is larger due to existence of edge direction, the frequency of each subgraph pattern is smaller. This also reduces the required computation for search. Figure 6 shows that the number of discovered frequent induced subgraphs remain nearly constant, and the computation time is proportional to the number of graph data. Computation time rapidly decreases as the number of vertex labels increases as shown in Figure 7 because the increase in the vertex labels also increases the number of possible patterns. However, the number of the frequent induced subgraphs does not decrease rapidly. This is because the basic subgraphs are artificially embedded in these simulation data, and thus, the number of the frequent induced subgraphs does not decrease below a certain level. Computation time and the number of frequent induced subgraphs increase when the minimum support is reduced as shown in Figure 8. However, their increase becomes insignificant at very small values of the minimum support, because there become no more undiscovered frequent induced

subgraphs at the samll support values. Though we have not checked due to the limitation of the tractable computation time in the experiments, if the minimum support is far too small, the computation time and the number of frequent induced subgraphs may increase considerablely again because the algorithm will pick up the random noise patterns. Figure 9 indicates the exponential increase in computation time with the increase in the average size of the graph transactions in the data. This is because the size of the graph transaction directly affects the combinatorial complexity of the subgraphs. In short summary, the proposed algorithm does not show intractable computational complexity except the cases for graphs of large size in the database. This character is considered to be suitable for the chemical compound structure analysis such as the carcinogenesis prediction that only requires the handling of graph transactions of moderate size.

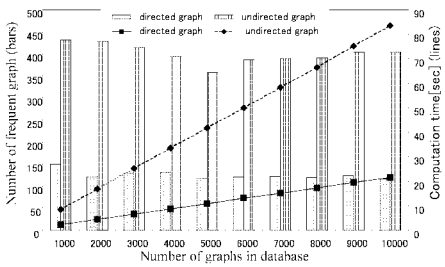


Figure 6: Complexity v.s. number of transactions

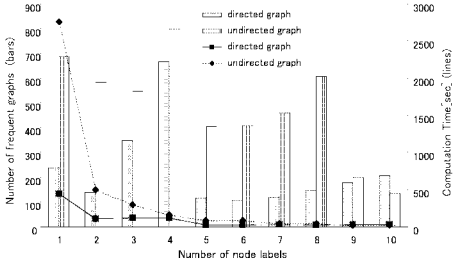


Figure 7: Complexity v.s. number of vertex labels

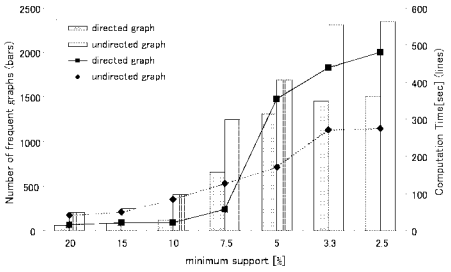


Figure 8: Complexity v.s. minimum support

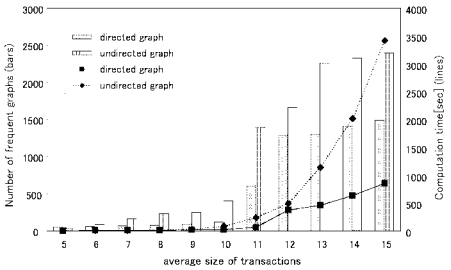


Figure 9: Complexity v.s. transaction size

4 Application to Chemical Analysis

Many new chemical compounds are being synthesized in seeking for better life and health, but their toxicity is also causing problems. The need for effectively evaluating their effects on living bodies and environments such as toxicity, resolvability and condensationability is called for. However, the experiments on living bodies and environments are quite expensive and very time consuming, and thus it is sometimes prohibitive to rely solely on experiments from both

economical and efficiency point of view. It will be extremely useful if some of these properties can be shown predictive by the structure of the chemical substances before being actually synthesized. The proposed approach was applied to chemical carcinogenesis analysis which is a challenge topic proposed in IJCAI-97 by Srinisavan et al. [16]. The task is to find structures typical to carcinogen of organic chlorides.

Though many studies have been reported, most of the works were to find carcinogenesis conditions that are characterized by man-made features on the structure such as existence of benzene ring and chlorine [8], [9]. This approach may miss sensitive structures which are unknown by human experts. In contrast, Dehaspe et al. tried to mine carcinogenic substructures from the direct representation of the chemical structures in form of the first order predicates [10]. They used ILP in conjunction with a levelwise search to improve the search efficiency. However, the search space is still large that only the substructure descriptions of short length consisting of six predicates at maximum were found within a tractable computation time even by the improved search technique. These descriptions represent small molecular substructures consisting of only 3 atoms or so. In this section, we assess the power of our proposed method in terms of the computational efficiency and the size of the discovered chemical substructures which indicate positive or negative carcinogenesis.

The objective data were obtained from the website of National Toxicology Program (NTP). Many chemical compounds do not have the description of carcinogenesis classes of positive and negative in the data set, thus we have added the class information obtained from the website of Oxford University. Totally, the 300 compounds whose classes are known were selected for the analysis from the 515 compounds in the original data, of which 185 compounds have positive carcinogenesis and the rests are negative. Thus, the fraction of the carcinogenic compounds is 61.7%. The types of atoms involved in the compounds are C, H, O, Cl, F, S and some cations, and the types of bonds are single, double, aromatic and cation bonds. Each transaction data were preprocessed to add artificial edges from each vertex to every other vertex that is within the distance of 6 edges as depicted in Figure 10. In the figure, only the edges added from an oxygen to the other atoms are shown. Each added edge has a label to indicate the distance between the two vertices that are connected by the edge. This enables to mine the frequent cooccurrence of some specific structures at a specific distance within 6. The distance limit of 6 was determined based on the chemical insights that the influence of an atom does not usually propagate along the path more than 6 bonds in molecules of moderate sizes. Furthermore, an isolated vertex labeled by the carcinogenesis class of the compound, *i.e.*, "class vertex", is added to each chemical structure graph. For instance, the class vertex labeled as negative is added in Figure 10, since its carcinogenesis is negative.

The analysis was made on the same PC described in the previous section. Table 2 shows the number of the candidate induced subgraphs (NOC) and that of the discovered frequent induced subgraphs (NOF) for each level of the search, *i.e.*, the size of the induced subgraphs. The threshold value *minsup* was set to

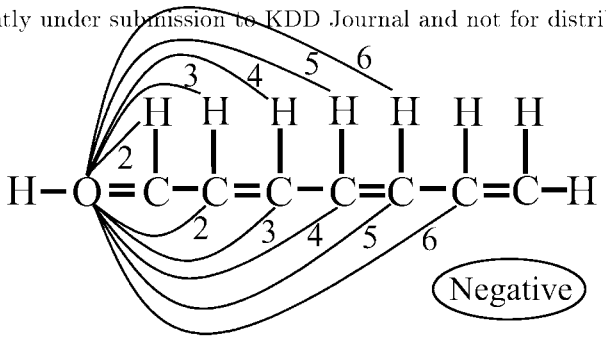


Fig. 10. Addition of artificial edges.

be 10%, 15% and 20%. In each *minsup* case, all frequent induced subgraphs were exhaustively discovered. The computation time required to complete the search was far longer for the *minsup* of smaller value, and was almost 8 days for 10%, while it was only about 40 minutes for 20%. The size of the largest frequent induced subgraph discovered in the case of 10% was 13. This is almost equivalent to the description length of 26 predicates in PROGOL [10]. The proposed approach is very powerful to exhaustively mine quite large and complex substructures.

Table 2. Results for three *minsup* values.

	minsup = 20%		*minsup* = 15%		*minsup* = 10%	
L	NOC	NOFS	NOC	NOFS	NOC	NOFS
1	24	7	24	8	24	10
2	280	62	360	67	550	108
3	2277	477	2525	640	4558	964
4	6223	2178	9709	3333	18268	5912
5	9767	4806	18740	9372	40744	19568
6	6899	4726	19813	13479	56179	37219
7	2655	2179	11989	9499	52082	41639
8	668	655	4347	4019	33208	29817
9	118	118	1212	1199	15618	15242
10	7	7	220	220	5739	5725
11	-	-	21	21	1455	1455
12	-	-	1	1	23	23
13	-	-	-	-	15	15
Total	28918	15215	68961	41858	228663	157897

L:level(number of vertices included in frequent subgraph)
NOC:number of candidates
NOFS:number of frequent graphs

Figure 11 shows the relation between the confidence deviation Δ and the cover rate CR of the discovered association rules whose heads contain the class

vertices. Δ of an association rule $G_b \Rightarrow G_h$ is given as follows.

$$\Delta = \begin{cases} conf(G_b \Rightarrow G_h) - fr_p & \text{if } G_h \text{ contains a positive class vertex.} \\ conf(G_b \Rightarrow G_h) - fr_n & \text{if } G_h \text{ contains a negative class vertex.} \end{cases}$$

Here, fr_p is the fraction of positive compounds in the data, *i.e.*, 61.7% in this case, and fr_n is that of negative compounds, i.e., 38.3% (=100%-61.7%). CR of a set of association rules is the fraction of graphs (chemical compounds) whose classes are derived by applying the rule set to the data. Given a value of Δ_{th}, a set of association rules each having Δ more than the Δ_{th} is defined, and CR of the rule set is calculated. As shown in Figure 11, the set of all discovered association rules covers all data in each case of the support threshold. However, the rule sets obtained for the higher support thresholds do not contain rules having significantly high or low confidence. In contrast, the rule set derived for the 10% threshold involves some rules having significant confidence. Accordingly, the exhaustive search for low support threshold is considered to be very effective to mine valuable rules.

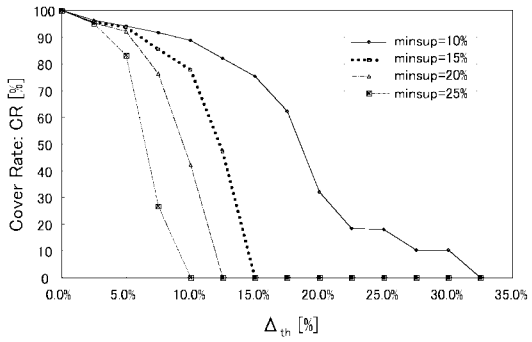
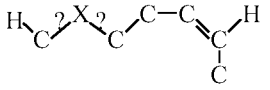


Fig. 11. Relation between Δ_{th} and cover rate CR.

Figure 12 shows some association rules obtained for the carcinogenesis class. The first rule is the most significant association rule having $\Delta = 32.2\%$ obtained for the support threshold 10%. This is very simple, but indicates that a sulfur atom plays an important role to suppress the carcinogenesis. The second rule shows a quite complex substructure of positive carcinogenesis having moderately high significance. The symbol X of a vertex and ? of an edge indicate that their labels are arbitrary, in other words, a specific type of an atom and a bond at these locations do not influence the carcinogenesis, but some atom and bond must be there. The third is an example of a less significant but more complex substructure involving a benzene ring. Many other rules also involve benzene rings. This is consistent with the chemical knowledge that benzene rings frequently have a positive effect on the carcinogenesis.

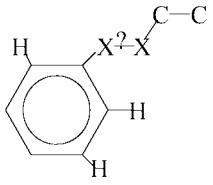
S=> Negative

sup=10.3%, conf=70.5%,Δ=32.2%



=> Positive

sup=10.3%, conf=77.5%, Δ=15.8%



=> Positive

sup=10.3%. conf=73.8%, Δ=12.1%

Fig. 12. Examples of discovered association rules.

5 Discussion

Dehaspe et al. claimed that the causation of chemical carcinogenesis is highly complex with many separate mechanisms involved [10]. In this study, the association between the chemical substructures and the carcinogenesis was investigated exhaustively, but the strong significance of the association was not observed except the case of the sulfur. This fact strengthens their claim. However, the resultant association rules obtained by the proposed approach indicate that quite complex chemical substructures have contributions to some degree to the positive and negative carcinogenesis. We also applied our approach to the chemical compound data of mutagenesis [22]. The chemical mechanism to cause the mutagenesis is considered among chemists to be more direct and simpler. In fact, our approach found quite significant and complex substructures in this case [23].

The efficiency of the proposed algorithm is higher than the past approaches which try to mine the complete set of frequent subgraphs. However, the required computation time easily becomes intractable when our approach is applied to the graphs having large sizes. A major portion of the required computation time is caused by the search for the parts in the given graphs which are isomorphic with the candidate frequent induced subgraphs. In the field of mathematical graph theory, several methods to enhance the efficiency of the isomorphism checking between two given graphs have been proposed. These introduced the use of the invariance of the graphs [18], [24]. Though the search of the isomorphic part in a large graph is different from the isomorphism checking between two graphs, the use of the graph invariance has a possibility to improve the search efficiency.

6 Conclusion

A novel approach was proposed that can efficiently mine frequently appearing induced subgraphs in a given graph data set and the association rules among the frequent induced subgraphs. In this approach, a graph is represented in form of an adjacency matrix. The matrices are uniquely ordered, and isomorphism among the subgraphs of the given graph transactions are efficiently found. Its performance has been evaluated for both the artificial simulation data and the real world chemical carcinogenesis data. The powerful performance of this approach has been confirmed through these evaluation.

Acknowledgement

The authors wish to thank Prof. Takashi Okada in Center for Information providing us with the chemical compound data and his expertise in the chemistry domain.

References

1. Agrawal, R. and Srikant, R. 1995. Mining sequential patterns. In Proceedings of the Eleventh International Conference on Data Engineering (ICDE'95), pp.3–14.
2. Mannila, H., Toivonen, H. and Verkamo, A.I. 1997. Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery Vol.1, No.3, pp.259 289.
3. Sintani, T. and Kituregawa, M. 1998. Mining algorithms for sequential patterns in parallel: Hash based approach, In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), pp.283 294.
4. Srikant, R., Vu, Q. and Agrawal, R. 1997. Mining Association Rules with Item Constraints. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), pp.67 73.
5. Klopman, G. 1984. Artificial intelligence approach to structure activity studies. J. Amer. Chem. Soc., Vol.106, pp.7315-7321.
6. Klopman, G. 1992. MultiCASE 1. A hierarchical computer automated structure evaluation program, QSAR, Vol.11, pp.176 184.
7. Wang, K. and Liu, H. 1997. Schema discovery for semistructured data. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), pp.271–274.
8. Kramer, S., Pfahringer, B. and Helma, C. 1997. Mining for causes of cancer: Machine learning experiments at various levels of detail. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), pp.223 226.
9. King, R., Muggleton, S., Srinivasan, A. and Sternberg, M. 1996. Structure-activity relationships derived by machine learning; The use of atoms and their bond connectives to predict mutagenicity by inductive logic programming. In Proceedings of the National Academy of Sciences, Vol.93, pp.438-442.
10. Dehaspe, L., Toivonen, H. and King, R.D. 1998. Finding frequent substructures in chemical compounds. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), pp.30 36.

11. Mannila, H. and Toivonen, H. 1997. Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery, Vol.1, No.3, pp.241-258.
12. Yoshida, K. and Motoda, H. 1995. Clip: Concept Learning from Inference Pattern. Artificial Intelligence, Vol.75, No.1, pp.63-92.
13. Matsuda, T., Horiuchi, T., Motoda, H. and Washio, T. 2000. Extension of Graph-Based Induction for General Graph Structured Data. In Proceedings of the Fourth Pacific-Asia Conference of Knowledge Discovery and Data Mining (PAKDD2000), pp.420–431.
14. Cook, D.J. and Holder, L.B. 1994. Substructure Discovery Using Minimumn Description Length and Background Knowledge, Journal of Artificial Intelligence Research, Vol.1, pp.231-255.
15. Inokuchi, A., Washio, T. and Motoda, H. 1999. Basket analysis for graph structured data. In Proceedings of PAKDD99: Methodologies for Knowledge Discovery and Data Mining, pp.420–431.
16. Srinisavan, A., King, R.D., Muggleton, S.H. and Sternberg, M.J.E. 1997. The predictive toxicology evaluation challenge. In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97), pp.4–9.
17. Biggs, N., 1973. *Algebraic Graph Theory.* Cambridge Univ. Press.
18. Fortin, S. 1996. The graph isomorphism problem. Technical Report 96-20, University of Alberta, Edomonton, Alberta, Canada.
19. Inokuchi, A., Washio, T. and Motoda, H. 1999. Derivation of the topology structure from massive graph data. Discovery Science: Proceedings of the Second International Conference, DS'99, pp.330–332.
20. Agrawal, R. and Srikant, R. 1994. Fast algorithms for mining association rules. In Proc. of the 20th VLDB Conference, pp.487–499.
21. Inokuchi, A. 2000. The study on a fast mining method from massive graph structure data. Master thesis (in Japanese), I.S.I.R., Osaka Univ.
22. Debnath, A.K. et al. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. J. Med. Chem. Vol.34, pp.786–797.
23. Inokuchi, A. et al. 2000. Application of Frequent Substructure Mining to Mutagenesis Data Analysis. To appear in International Workshop of KDD Challenge on Real-world Data, PAKDD2000.
24. Read, R. and Corneil, D. 1977. The graph isomorphism disease. Journal of Graph Theory, Vol.1, pp.339–363.