# State of the Art of Graph-based Data Mining

Takashi Washio
The Institute of Scienti£c and Industrial Research, Osaka University
8-1, Mihogaoka, Ibaraki-shi
Osaka, Japan
washio@ar.sanken.osaka-u.ac.jp

Hiroshi Motoda
The Institute of Scienti£c and Industrial Research, Osaka University
8-1, Mihogaoka, Ibaraki-shi
Osaka, Japan
motoda@ar.sanken.osaka-u.ac.jp

## ABSTRACT

The need for mining structured data has increased in the past few years. One of the best studied data structures in computer science and discrete mathematics are graphs. It can therefore be no surprise that graph based data mining has become quite popular in the last few years.

This article introduces the theoretical basis of graph based data mining and surveys the state of the art of graph-based data mining. Brief descriptions of some representative approaches are provided as well.

## General Terms

Graph-based Data Mining

## Keywords

graph, tree, path, structured data, data mining

## 1. INTRODUCTION

During the past decade, the field of data mining has emerged as a novel field of research, investigating interesting research issues and developing challenging real-life applications. The objective data formats in the beginning of the field were limited to relational tables and transactions where each instance is represented by one row in a table or one transaction represented as a set. However, the studies within the last several years began to extend the classes of considered data to semi-structured data such as HTML and XML texts, symbolic sequences, ordered trees and relations represented by advanced logics. Many papers on data mining of semi-structured data have been presented in major international journals and conferences [19]. The research on data mining and machine learning of symbolic sequences also became active in the last several years. Many papers appeared in ILP, ALT and DS conferences [17]. Furthermore, in the last few years, many researchers started working on mining ordered tree structures [26]. One of the most recent research topics associated with structured data is multi-relational data mining whose main scope is to find patterns in expressive logical and relational languages from complex, multi-relational and structured data [1]. The main aim of mining semi-structured data, symbolic sequences and ordered trees is to extract patterns from structured data. Within this framework, the patterns mined are characterized by some measures such as frequency and information entropy are mined. The classes of the patterns handled in the multi-relational data mining are more expressive than the aforementioned data structures.

Recently, a novel field of data mining emerged from a topological view of the data structure. In mathematics, one of the most generic topological structures are graphs. Semi-structure represented by text tags, symbolic sequence and tree including ordered and unordered trees are subclasses of general graphs. The earliest studies to find subgraph patterns characterized by some measures from massive graph data were conducted by Cook and Holder (SUBDUE) [3] and Yoshida and Motoda (GBI) [25] in the middle of the 1990's. Their approaches used greedy search to avoid high complexity of the graph isomorphism problem, which resulted in an incomplete set of characteristic subgraphs. In 1998, Dehaspe and Toivonen proposed an ILP-based algorithm, WARMR, enabling a complete search for frequent subgraphs from graph data [6]. Subsequent work done by Nijssen and Kok proposed a faster algorithm, FARMER [21]. In 2000, Inokuchi et al. proposed an approach called AGM to combine Apriori algorithm and mathematical graph theory [11]. In 2001, De Raedt and Kramer proposed the version space based approach called MolFea to find characteristic paths from the graph data [4].

Based on these pioneering studies, the number of papers on graph mining is rapidly increasing. The total number of papers related to graph and tree mining in SIGMOD, SIGKDD, IJCAI/AAAI, ICML, ECML/PKDD and IEEE ICDM was 10 in 2001, whereas the number increased to 18 in 2002 in our count. In addition, a considerable number of papers on this topic appeared in other international conferences and workshops. Thus, the research field of graph mining just started to emerge. Since graph topology is one of the most fundamental structures studied in mathematics, and has a strong relation with logical languages, graph mining is expected to contribute to the development of new principles in data mining and machine learning. Furthermore, graph mining has a high potential to provide practical applications because the graph structured data widely occurs in various practical fields including biology, chemistry, material science and communication networking.

Graph mining has a strong relation with the aforementioned multi-relational data mining. However, the main objective of graph mining is to provide new principles and efficient algorithms to mine topological substructures embedded in graph data, while the main objective of multi-relational data mining is to provide principles to mine and/or learn the relational patterns represented by the expressive logical lan-

guages. The former is more geometry oriented and the latter more logic and relation oriented.

In this review article, the theoretical basis of graph-based data mining is explained in the following section. Second, the approaches to graph-based data mining are reviewed, and some representative approaches are briefly described.

## 2. THEORETICAL BASES

The theoretical basis of graph-based data mining is not limited to one principle although the history of this research field is still young. This is because research on graphs has a long history in mathematics. In this section, the five theoretical bases of graph-based data mining approaches are reviewed. They are subgraph categories, subgraph isomorphism, graph invariants, mining measures and solution methods. The subgraphs are categorized into various classes, and the approaches of graph-based data mining strongly depend on the targeted class. Subgraph isomorphism is the mathematical basis of substructure matching and/or counting in graph-based data mining. Graph invariants provide an important mathematical criterion to efficiently reduce the search space of the targeted graph structures in some approaches. Furthermore, the mining measures define the characteristics of the patterns to be mined similarly to conventional data mining. In this paper, the theoretical basis is explained for only undirected graphs without labels but with/without cyclic edges and parallel edges due to space limitations. But, an almost identical discussion applies to directed graphs and/or labeled graphs. Most of the search algorithms used in graph-based data mining come from artificial intelligence, but some extra search algorithms founded in mathematics are also used.

### 2.1 Subgraph Categories

Various classes of substructures are targeted in graph-based data mining. This is because the graph is one of the most generic data structures and includes characteristic substructures in various views.

Mathematically, a graph $G$ is represented as $G(V, E, f)$ where $V$ is a set of vertices, $E$ a set of edges connecting some vertex pairs in $V$, $f$ a mapping $f : E \rightarrow V \times V$. In the graph of Fig. 1 (a), $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}$. Each edge $e_h$ in $E$ has a relation represented as $f(e_h) = (v_i, v_j)$ in which $v_i$ and $v_j$ are in $V$, for example, $f(e_1) = (v_1, v_2)$, $f(e_2) = (v_1, v_2)$, $f(e_4) = (v_1, v_4)$ and $f(e_7) = (v_4, v_4)$ in Fig. 1 (a). The most generic class of the substructure of $G$ is a "*general subgraph*" where $V_s \subset V$, $E_s \subset E$ and $v_i, v_j \in V_s$ for all edges $f(e_h) = (v_i, v_j) \in E_s$. Fig. 1 (b) is an example of the general subgraph in which a vertex $v_5$ and edges $e_4, e_6, e_7, e_8, e_9$ are missed. Another important and generic class of the substructure is an "*induced subgraph*" where $V_s \subset V$, $E_s \subset E$ and $\forall v_i, v_j \in V_s$, $e_h = (v_i, v_j) \in E_s \Leftrightarrow f(e_h) = (v_i, v_j) \in E$. An induced subgraph $G_s^i$ of a graph $G$ has a subset of the vertices of $G$ and the same edges between pairs of vertices as in $G$. Fig. 1 (c) is an example of the induced subgraph in which a vertex $v_5$ is missed. In this case, only the edges $e_8$ and $e_9$ are also missed, and $e_4, e_6, e_7$ are retained since they exist among $v_1, v_3$ and $v_4$ in the original $G$. The third important and generic class of the substructure is a "*connected subgraph*" where $V_s \subset V$, $E_s \subset E$ and all vertices in $V_s$ are mutually reachable through some edges in $E_s$. Fig. 1 (d) is an example where $v_6$ is further missed from (c). Moreover, (d) is an example of an "*induced and connected subgraph*" since it satisfies both conditions of the induced subgraph and the connected subgraph.

An acyclic subgraph is called a "*tree*". Though the labels of vertices and edges are not considered in the aforementioned graph formulation, if we introduce the labels of edges in the tree, and if they are ordered in a way that the label of an edge is always younger than the labels of its lower (upper) and right (left) edges, the tree is defined as an "*ordered tree*". If the edge is not ordered or does not have labels, the tree is called an "*unordered tree*". Fig. 1 (e) is an example of the ordered tree in terms of the edge indices which has a root node $v_1$, and is included as a substructure in the original graph (a). The graph (f) is an example of the unordered tree. If the substructure does not include any branches, it is called a "*path*" of the original $G$. The graph (g) is an example path.

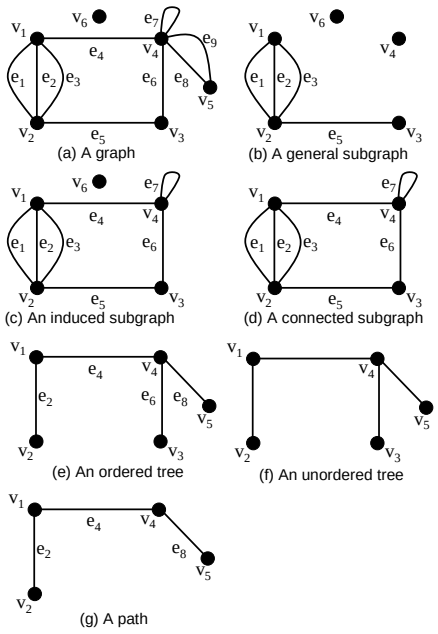


Figure 1: Representative subgraphs

### 2.2 Subgraph Isomorphism

Given two graphs $G_x(V_x, E_x, f_x)$ and $G_y(V_y, E_y, f_y)$, the "*subgraph isomorphism*" problem is to find the subgraphs $G_{sx}(V_{sx}, E_{sx}, f_x)$, $G_{sy}(V_{sy}, E_{sy}, f_y)$ and a bijection mapping $g_{xy}$ between the vertices in $V_{sx}$ and the vertices in $V_{sy}$ such that $G_{sx}$ and $G_{sy}$ are identical, *i.e.*, $f_x(e_{xh}) = (v_{xi}, v_{xj}) \in E_{sx}$ iff $f_y(e_{yh}) = (v_{yi}, v_{yj}) \in E_{sy}$ where $v_{yi} = g_{xy}(v_{xi})$ and $v_{yj} = g_{xy}(v_{xj})$. The existence of $g_{xy}$ ensures the topological identity between $G_{sx}$ and $G_{sy}$. For example, the graphs (b) and (d) in Fig. 1 commonly share the subgraph consisting of the vertices $\{v_1, v_2, v_3\}$ and the edges $\{e_1, e_2, e_3, e_5\}$ under the bijection mapping of $v_i = g_{bd}(v_i), i = 1, 2, 3$. Thus, this mapping is a "*subgraph isomorphism*" between the graphs (b) and (d).

In graph-based data mining, the subgraph isomorphism problem is further extended to cover multiple graphs. Given a set of graphs $\{G_k(V_k, E_k, f_k)|k = 1, ..., n\}$, the problem is to find the subgraph $G_s(V_s, E_s, f_s)$, a set of subgraphs $\{G_{sk}(V_{sk}, E_{sk}, f_k)|k = 1, ..., n\}$ and a bijection mapping $f_s$ between the vertices of $G_s$ and every $G_{sk}$ for all $k = 1, ..., n$. When $f_s$ satisfying this condition exists, $G_s(V_s, E_s)$ is a com-

mon subgraph of the given set of graphs. This definition of the subgraph isomorphism provides the basis for matching and counting of topologically identical parts of the given graphs.

The graph isomorphism problem (i.e. the problem of deciding whether two graphs have identical topological structure) has an unknown computational complexity. It is either NP-complete or polynomial and all attempts to classify it in one of these two categories have so far failed. On the other hand, the subgraph isomorphism problem (i.e. the problem of deciding whether one graph is a subgraph of another one) is known to be NP-complete.

## 2.3 Graph Invariants

Graph invariants are the quantities to characterize the topological structure of a graph. If two graphs are topologically identical, *i.e.*, isomorphic, they also have identical graph invariants, though the reverse property does not hold. Examples of graph invariants are the number of vertices, the degree of each vertex, *i.e.*, the number of edges connected to the vertex, and the number of cyclic loops. Isomorphic graphs always have identical values of all graph invariants, while the identical values of given graph invariants does not imply the isomorphism of the graphs. Accordingly the use of graph invariants is not equivalent to complete isomorphic subgraph matching and counting. However, graph invariants can be used to reduce the search space to solve the subgraph isomorphism problem. If any of the graph invariants show different values between two subgraphs, the subgraphs are not isomorphic.

In discrete mathematics, many studies to solve the graph isomorphism problem between two graphs by introducing the graph invariants have been made in the last decades. One of the representative works is the NAUTY algorithm which is known to be one of the fastest algorithms for graph isomorphism [18]. It focuses the graph invariants on each vertex in the graphs, e.g., its degree and the numbers of its adjacent vertices having certain degrees, to reduce the search space drastically. The mapping of vertices having different values of the graph invariants between the given two graphs never exist in the graph isomorphism problem, because such vertices locate at mutually different positions of the graphs in the sense of topology. Accordingly, it partitions the set of vertices $V$ of a given graph into its subsets where every vertex has mutually identical values of graph invariants. Then, it checks the graph isomorphism between the subsets having identical values of the graph invariants for the two graphs in a brute force manner. If all subsets are isomorphic between the two graphs, the isomorphism of the two graphs is concluded. This divide and conquer approach based on the graph invariants significantly enhances the computational efficiency in most of the practical problems.

More direct representation and handling of the graph structure can be made in an algebraic framework. The "*adjacency matrix*" is one such example [11]. The $i$-th row and the $i$-th column correspond to the $i$-th vertex $v_i$. The $i,j$-element of the matrix is the set of the edges $\{f(e_h) = (v_i, v_j)\}$ connecting the vertices. To be rigorous, it is not a matrix because its elements are not real numbers, but here we call it matrix for convenience. If no edge exists between the two vertices, the element is nil or 0. For example, the adjacency matrix of Fig. 1 (a) is represented as follows.

$$\begin{pmatrix} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \\ v_1 & 0 & \{e_2,e_3,e_4\} & 0 & \{e_1\} & 0 & 0 \\ v_2 & \{e_2,e_3,e_4\} & 0 & \{e_5\} & 0 & 0 & 0 \\ v_3 & 0 & \{e_5\} & 0 & \{e_6\} & 0 & 0 \\ v_4 & \{e_1\} & 0 & \{e_6\} & \{e_7\} & \{e_8,e_9\} & 0 \\ v_5 & 0 & 0 & 0 & \{e_8,e_9\} & 0 & 0 \\ v_6 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

All graph invariants are derived from this direct representation of a graph. A drawback is the complexity on the memory consumption and the processing time.

One of the most generic and important graph invariants is "*canonical label*" and "*canonical form*" [11; 4]. A graph can be represented by multiple forms. For example, all adjacency matrices obtained from an adjacency matrix through permutations of rows and columns represent an identical graph. This ambiguity is commonly seen in various graph representations, and induces a combinatorial increase of the search space in the graph isomorphism problem. The canonical label is the most effective remedy for this issue. Various definitions of the canonical label are possible, but it has to uniquely represent a graph. For example, the $n \times n$ adjacency matrix can be labeled by a code generated from the matrix elements in the following order.

$$x_{1,1}x_{1,2}x_{2,2}x_{1,3}x_{2,3}x_{3,3}...x_{n-2,n}x_{n-1,n}x_{n,n},$$

where only the upper right columns are used because of the diagonal symmetry of the matrix for the undirected graph. Similar coding is applicable to the case of directed graphs. We can uniquely define the canonical label as the lexicographically minimum (or maximum) code and the canonical form of the adjacency matrix as the matrix corresponding to the canonical label. The introduction of the canonical label and the canonical form significantly reduces the graph representation ambiguity and the search space.

Recently, a new research direction to use graph invariants for the construction of a high dimensional feature space characterizing a graph has been proposed [13]. Various machine learning, data mining and statistical approaches can be applied if the graph is transformed into a feature vector. The newly emerging approach collects many graph invariants on a graph $G$, and forms a feature vector $X_G$ consisting of the graph invariants. When the graph $G$ is very complex, the dimension of $X_G$ required to approximate the graph topology closely can be very large. This causes computational problems for many mining approach. To alleviate these issues, the new approach introduces a kernel function $K(X_{G_x}, X_{G_y})$ and a mapping $\phi$:$X_G \rightarrow H$ enabling the representation of $K$ by the inner product $< \phi(X_{Gx}), \phi(X_{Gy}) >$. $K$ represents a similarity between two graphs $G_x$ and $G_y$, and $H$ is usually the Hilbert space. Because the value of the inner product can be derived without directly computing the vectors in $H$, this approach can avoid the tractability issue. Moreover, the issue of the sparse distribution is also alleviated because the similarity $K$ is given by a scalar value. A drawback of this approach is that these kernels cannot be computed efficiently. Some appropriate alternatives and similarity measures have been proposed in recent works [7; 13].

## 2.4 Mining Measures

Various measures to mine substructures of graphs are used similarly to conventional data mining. The selection of the

measures depends on the objective and the constraints of the mining approach. The most popular measure in the graph-based data mining is the following "*support*" whose definition is identical with that of Basket Analysis [2]. Given a graph data set $D$, the support of the subgraph $G_s$, $sup(G_s)$, is defined as

$$sup(G_s) = \frac{\text{number of graphs including } G_s \text{ in } D}{\text{total number of graphs in } D}.$$

This measure has an anti-monotonic property that $sup(G_{sx}) \leq sup(G_{sy})$ if $G_{sy}$ is a subgraph of $G_{sx}$. By specifying a "*minimum support*" value *minsup*, subgraphs $\{G_s\}$ whose support values are more than the *minsup* are mined in some approaches.

The anti-monotonicity of the support is insufficient for some mining objectives. For example, an analysis to find subgraphs $G_s$s which appear more than a minimum support *minsup* but also less than a maximum support *maxsup* ($minsup < maxsup$) may be needed in some application domains. The former constraint $c_1$ is anti-monotonic, *i.e.*,

$$(G_s < G_{c_1}) \wedge (G_{c_1} \in sol(c_1)) \rightarrow (G_s \in sol(c_1)),$$

where $G_s < G_{c_1}$ means that $G_s$ is more general than $G_{c_1}$ ($G_s$ is a subgraph of $G_{c_1}$) and $sol(c_1)$ a solution set of $c_1$, *i.e.*, a set of all subgraphs satisfying $c_1$ in $D$. The latter constraint $c_2$ is monotonic, *i.e.*,

$$(G_s < G_{c_2}) \wedge (G_s \in sol(c_2)) \rightarrow G_{c_2} \in sol(c_2)),$$

where $sol(c_2)$ a solution set of $c_2$. The negation of a monotonic constraint is anti-monotonic, and the negation of an anti-monotonic constraint is monotonic. The version space algorithm of graph-based data mining can handle these measures as described later [4].

Many other mining measures which are very commonly used in the machine learning field are also used in some graph-based data mining approaches, especially, information entropy, information gain, gini-index and minimum description length (MDL) [25; 3].

## 2.5 Solution Methods

The aforementioned subgraph isomorphism problem among many graphs must be solved by using efficient search methods in graph-based data mining. Roughly speaking, five types of search methods are currently used. These are categorized into heuristic search methods and complete search methods in terms of the completeness of search. They are also categorized into direct and indirect matching methods from the view point of the subgraph isomorphism matching problem. The indirect matching does not solve the subgraph isomorphism problem but subgraph similarity problem under some similarity measure.

The first type of the search method is the conventional greedy search which has been used in the initial works in graph-based data mining [25; 3]. This type belongs to heuristic search and direct matching. The greedy search is further categorized into depth-first search (DFS) and breadth-first search (BFS). DFS was used in the early studies since it can save memory consumption. Initially the mapping $f_{s1}$ from a vertex in a candidate subgraph to a vertex in the graphs of a given data set is searched under a mining measure. Then another adjacent vertex is added to the vertex mapped by $f_{s1}$, and the extended mapping $f_{s2}$ to map these two vertices to two vertices in the graphs of a given data set is searched under the mining measure. This process is repeated until no more extension of the mapping $f_{sn}$ is available where $n$ is the maximal depth of the search of a DFS branch. A drawback of this DFS approach is that only an arbitrary part of the isomorphic subgraphs can be found when the search must be stopped due to the search time constraints if the search space is very large. Because of the recent progress of the computer hardware, more memory became available in the search. Accordingly, the recent approaches to graph-based data mining are using BFS. An advantage of BFS is that it can ensure derivation of all isomorphic subgraphs within a specified size of the subgraphs even under greedy search scheme. However, the search space is so large in many applications that it often does not fit in memory. To alleviate this difficulty, beam search method is used in the recent greedy search based approach [3; 25] where the maximum number of the BFS branches is set, and the search proceeds downward by pruning the branches which do not fit the maximum branch number. Since this method prunes the search paths, the search of the isomorphic subgraphs finishes within tractable time while the completeness of the search is lost.

The second type of search method is to apply the framework of "*inductive logic programming (ILP)*" [20]. The "*induction*" is known to be the combination of the "*abduction*" to select some hypotheses and the "*justification*" to seek the hypotheses to justify the observed facts. Its main advantage is the abilities to introduce background knowledge associated with the subgraph isomorphism and the objective of the graph-based data mining. It can also derive knowledge represented by "*first order predicate logic*" from a given set of data under the background knowledge. The general graph is known to be represented by "*first order predicate logic*". The first order predicate logic is so generic that generalized patterns of graph structures to include variables on the labels of vertices and edges are represented. ILP is formalized as follows [20]. Given the background knowledge $B$ and the evidence (the observed data) $E$ where $E$ consists of the positive evidence $E^+$ and the negative evidence $E^-$, ILP finds a hypothesis $H$ such that the following "*normal semantics*" conditions hold.

1. Posterior Satisfiability: $B \wedge H \wedge E^- \not\models \Box$,
2. Posterior Sufficiency: $B \wedge H \models E^+$,

where $\Box$ is "*false*", and hence $\not\models \Box$ means that the theory is satisfiable. In case of ILP, intentional definitions are derived from the given data represented by instantiated first order predicates, *i.e.*, extensional definitions. For ILP, the advantage is not limited to the knowledge to be discovered but the ability to use the positive and the negative examples in the induction of the knowledge. A disadvantage is the size of the search space which is very huge in general and computational intractability. The ILP method can be any of heuristic, complete, direct and indirect search according to the background knowledge used to control the search process. When control knowledge is used to prune some search paths having low possibility to find isomorphic subgraphs under a given mining measure, the method is heuristic. Otherwise, it is complete. When some knowledge on predetermined subgraph patterns are introduced to match subgraph structures, the method is indirect since only the

subgraph patterns including the predetermined patterns or being similar to the predetermined patterns are mined. In this case the subgraph isomorphism is not strictly solved.

The third type is to use "*inductive database*" [10]. Given a data set, a mining approach such as inductive decision tree learning, basket analysis [2] and ILP is applied to the data to pregenerate inductive rules, relations or patterns. The induced results are stored in a database. The database is queried by using a query language designed to concisely express query conditions on the forms of the pregenerated results in the database. This framework is applicable to graph-based mining. Subgraphs and/or relations among subgraphs are pregenerated by using a graph-based mining approach, and stored in an inductive database. A query on the subgraphs and/or the relations is made by using a query language dedicated to the database. An advantage of this method is the fast operation of the graph mining, because the basic patterns of the subgraphs and/or the relations have already been pregenerated. Potential drawback is large amount of computation and memory to pregenerate and store the induced patterns. This search method is used in conjunction with the following complete level-wise search method in some works [4].

The fourth type is to apply "*complete level-wise search*" which is popularly used in the basket analysis and "*inductive database*". These are complete search and direct methods. In case of Apriori algorithm which is the most representative for the basket analysis [2], all frequent items which appear more than a specified minimum support "*minsup*" in the transaction data are enumerated as frequent itemsets of size 1. This task is easily conducted by scanning the given transaction data once. Subsequently, the frequent itemsets are joined into the candidate frequent itemsets of size 2, and their support values are checked in the data. Only the candidates having the support higher than *minsup* are retained as the frequent itemsets of size 2. This process to extend the search level in terms of the size of the frequent itemsets is repeated until no more frequent itemsets are found. This search is complete since the algorithm exhaustively searches the complete set of frequent item sets in a level-wise manner. In case of the graph-based data mining, the data are not the transactions, *i.e.*, sets of items, but graphs, *i.e.*, combinations of a vertex set $V(G)$ and an edge set $E(G)$ which include topological information. Accordingly, the above level-wise search is extended to handle the connections of vertices and edges [11; 4]. Similarly to the Apriori algorithm, the search in a given graph data starts from the frequent graphs of size 1 where each consists of only a single vertex. Subsequently, the candidate frequent graphs of size 2 are enumerated by combining two frequent vertices. Then the support of each candidate is counted in the graph data, and only the graphs having higher support than the *minsup* are retained. In this counting stage, the edge information is used. If the existence and the label of the edge between the two vertices do not match, the graph of size 2 is not counted as an identical graph. This process is further repeated to incrementally extend the size of the frequent graphs in a level wise manner, and finishes when the frequent graphs are exhaustively searched. In inductive database and constraint-based mining, the algorithm can be extended to introduce monotonic measures such as "*maxsup*" [4].

The fifth type is "*Support Vector Machine (SVM)*" [23]. This is a heuristic search and indirect method in terms of the subgraph isomorphism problem and used in the graph classification problem. It is not dedicated to graph data but to feature vector data. Given feature and class vectors

$$(x_1, y_1), ..., (x_L, y_L) \quad x_i \in Z, y_i \in \{+1, -1\},$$

where $L$ is the total number of data, $i = 1, ..., L$, $Z$ a set of vectors and $y_i$ a binary class labels, each sample feature vector $x_1$ in the data is classified by

$$y = sgn(\sum_{j=1}^{n} y_j \alpha_j \phi(x_i) \bullet \phi(x) + b).$$

Here $\phi : Z \rightarrow H$ where $H$ is the Hilbert space, $\alpha_i, b \in R$ and $\alpha_i$ positive finite. By extending the feature space to far higher dimension space via $\phi$, SVM can properly classify the samples by a linear hyper plane even under complex nonlinear distributions of the samples in terms of the class in $Z$. The product $\phi(x_i) \bullet \phi(x)$ can be represented by the aforementioned kernel function $K(X_{G_x}, X_{G_y})$ for graphs where $X_{G_x} = x_i$ and $X_{G_y} = x$. Accordingly, SVM can provide an efficient classifier based on the set of graph invariants.

# 3. APPROACHES OF GRAPH MINING

The approaches to graph-based data mining are categorized into five groups. They are greedy search based approach, inductive logic programming (ILP) based approach, inductive database based approach, mathematical graph theory based approach and kernel function based approach. In this section, some major studies in each category are described, and representative methods among the studies are sketched.

## 3.1 Greedy Search Based Approach

Two pioneering works appeared in around 1994, both of which were in the framework of greedy search based graph mining. Interestingly both were originated to discover concepts from graph representations of some structure, e.g. a conceptual graph similar to semantic network and a physical system such as electric circuits.

One is called "SUBDUE" [3]. SUBDUE deals with conceptual graphs which belong to a class of connected graph. The vertex set $V(G)$ is $R \cup C$ where $R$ and $C$ are the sets of labeled vertices representing relations and concepts respectively. The edge set $E(G)$ is $U$ which is a set of labeled edges. Though the original SUBDUE targeted the discovery of repeatedly appearing connected subgraphs in this specific type of graph data, *i.e.*, concept graph data, the principle can be applied to generic connected graphs.

SUBDUE starts looking for a subgraph which can best compress an input graph $G$ based on Minimum Description Length (MDL) principle. The found subgraph can be considered a concept. This algorithm is based on a computationally-constrained beam search. It begins with a subgraph comprising only a single vertex in the input graph $G$, and grows it incrementally expanding a node in it. At each expansion it evaluates the total description length (DL), $I(G_s)+I(G|G_s)$, of the input graph $G$ which is defined as the sum of the two: DL of the subgraph, $I(G_s)$, and DL of the input graph, $I(G|G_s)$, in which all the instances of the subgraph are replaced by single nodes. It stops when the subgraph that minimizes the total description length is found. The search

is completely greedy, and it never backtracks. Since the maximum width of the beam is predetermined, it may miss an optimum $G_s$. One of the good features of SUBDUE is that it can perform approximate matching to allow slight variations of subgraphs. It can also embed background knowledge in the form of predefined subgraphs. After the best substructure is found and the input graph is rewritten, the next iteration starts using the rewritten graph as a new input. This way, SUBDUE finds a more abstract concept at each round of iteration. As is clear, the algorithm can find only one substructure at each iteration. Furthermore, it does not maintain strictly the original input graph structure after compression because its aim is to facilitate the global understanding of the complex database by forming hierarchical concepts and using them to approximately describe the input data. Recently, a new technique to induce a graph grammar has been developed by the same group [12]. The other one is called "*Graph Based Induction*(GBI)" [25]. GBI was originally intended to find interesting concepts from inference patterns by extracting frequently appearing patterns in the inference trace. GBI was formulated to derive a graph having a minimal size similarly to SUBDUE by replacing each found subgraph with one vertex that it repeatedly compress the graph. It used an empirical graph size definition that reflected the sizes of extracted patterns as well as the size of compressed graph. This prevented the algorithm from continually compressing, which meant the graph never became a single vertex. GBI can handle both directed and undirected labeled graph with closed paths (including closed edges). An opportunistic beam search similar to genetic algorithm was used to arrive at local minimum solutions. In this algorithm, the primitive operation at each step in the search was to find a good set of linked pairs of vertices by an edge to chunk, *i.e.*, pairwise chunking. The idea of pairwise chunking in case of a directed graph is given in Fig. 2. GBI chunks the triplet $(A_k, f_i, B_j)$ which minimizes the above graph size where $A_k$ and $B_j$ are the vertices, and $f_i$ is a link directing from $B_j$ to $A_k$. This chunking is repeated until the size of the graph reaches a local minimum. It is possible that either one or both of the paired nodes have already been chunked. Chunking can be nested. GBI remembers the link information and can reconstruct the original graph at any time of the search.

Later an improvement was made to use other measures than frequency. Because the search is local and stepwise, an indirect measure rather than a direct estimate of the graph size is adopted to find the most promising pairs. Such measures include information gain, information gain ratio and gini index, all of which can be evaluated from the frequency of the pair selected to chunk. By using such measures it is possible to extract substructures that are discriminating if a class label is attached to each input graph, and construct a classifier. This idea was applied to induce a classifier for

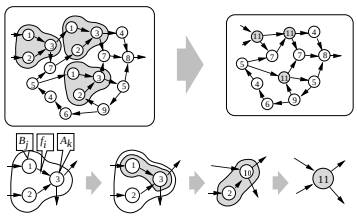

Figure 2: Graph compression by pairwise chunking.

command prediction. It is known that a simple command sequence is not enough to accurately predict the next command. The file I/O relations across the issued commands together with the command sequence information form a directed tree, and the prediction task is formulated to predict the root node of a tree from its subtree. Figure 3 depicts the process of the subgraph (subtree in this case) discovery in the commands and I/O history data indicated in the table. Unlabeled edges indicate command sequences and labeled edges indicate file I/O relations. As shown in steps (A), (B) and (C), the file *paper.dvi* is processed by three different commands: *xtex*, *xdvi* and *dvi2ps*. The corresponding directed graphs that are inputs to GBI are also given as (A), (B) and (C) in the figure. The GBI algorithm first chooses the *dvi* edge ($f_i$) and its starting vertex *latex* ($B_j$) for testing, and chunks the triplets (*xdvi, dvi, latex*) in (B) and (*dvi2ps, dvi, latex*) in (C) in accordance with the information gain. Next, the algorithm chooses the unlabeled edge and its starting vertex *xdvi* for testing and chunks the triplet ((*dvi2ps, dvi, latex*) *, unlabel, xdvi*). This separates (C) from (B) and the induction stops. The parts surrounded by round rectangles are the typical (discriminating) patterns. It is straightforward to construct a decision tree from these patterns.

The most recent version of GBI can handle two different measures, one for chunking and the other for extracting patters. This is to overcome the problem caused by non monotonic nature of discriminating measure. The fact that a subgraph $A$ is a good discriminator does not necessarily mean that its subgraph $B$ is also a good discriminator. However, successive pairwise chunking requests that $B$ must be extracted in an earlier step in order to obtain $A$ later. Current GBI employs canonical labeling to see if two chunks obtained by different histories are isomorphic or not. Recently GBI is also being used as a feature constructor in a decision tree classifier for graph structured data [8].

| Step | Application | Input File |
|---|---|---|
| (A) | xtex | paper.dvi |
| | emacs | paper.tex |
| | latex | paper.tex |
| (B) | xdvi | paper.dvi |
| (C) | dvi2ps | paper.dvi |

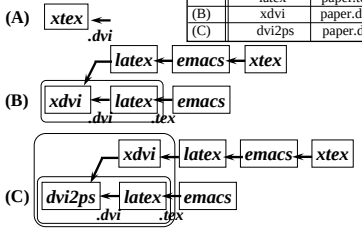


Figure 3: Subgraph discovery in commands-I/O history.

## 3.2 ILP Based Approach

To our knowledge, the first system to try complete search for the wider class of frequent substructure in graphs named WARMR was proposed in 1998 [6]. They combined ILP method with Apriori-like level wise search to a problem of carcinogenesis prediction of chemical compounds. The structures of chemical compounds are represented by the first order predicates such as $atomel(C, A1, c)$, $bond(C, A1, A2, BT)$, $aromatic\_ring(C, S1)$ and $alcohol(C, S2)$. The first two state that $A1$ which is a carbon atom bond to $A2$ where the bond type is $BT$ in a chemical compound $C$. The third represents that substructure $S1$ is an aromatic ring in a chemical compound $C$, and the last represents that $S2$ is an alcohol base in $C$. Because this approach allows variables to

be introduced in the arguments of the predicates, the class of structures which can be searched is more general than graphs. However, this approach easily faces the high computational complexity due to the equivalence checking under $\theta$-subsumption (an NP-complete operation) on clauses and the generality of the problem class to be solved. To alleviate this difficulty, a new system called FARMAR has recently been proposed [21]. It also uses the level wise search, but applied less strict equivalence relation under substitution to reduced atom sets. FARMAR runs two orders of magnitudes faster. However, its result includes some propositions having different forms but equivalent in the sense of the $\theta$-subsumption due to the weaker equivalence criterion. A major advantage of these two systems is that they can discover frequent structures in high level descriptions. These approaches are expected to address many problems, because many context dependent data in the real-world can be represented as a set of grounded first order predicates which is represented by graphs.

## 3.3 Inductive Database Based Approach

A work in the framework of inductive database having practical computational efficiency is MolFea system based on the level-wise version space algorithm [4]. This method performs the complete search of the paths embedded in a graph data set where the paths satisfy monotonic and anti-monotonic measures in the version space. The version space is a search subspace in a lattice structure. The monotonic and anti-monotonic mining measures described in subsection 2.4 define borders in the version space. To define the borders, the minimal and the maximal elements of a set in terms of generality are introduced. Let $F$ be a set of paths included in graph data, then define

$$max(F) = \{f \in F | \neg\exists q \in F : f \leq q\},$$

$$min(F) = \{f \in F | \neg\exists q \in F : q \leq f\}$$

where $f \leq q$ indicates that $q$ is more general than or equally general to $f$. Then, given a set $sol(c)$ of all paths satisfying a primitive constraint $c$ in a graph data set $D$, the borders $S(c)$ and $G(c)$ are defined as

$$S(c) = min(sol(c)) \text{ and } G(c) = max(sol(c)).$$

When $c$ is a proper anti-monotonic constraints, $G(c) = \{\top\}$ and $S(c) \neq \{\bot\}$ holds, and when $c$ is a proper monotonic constraints, $S(c) = \{\bot\}$ and $G(c) \neq \{\top\}$ holds. Under these definitions, the two borders in the version space have the following relation with $sol(c)$.

$$sol(c) = \{f \in F | \exists s \in S(c), \exists g \in G(c) : g \leq f \leq s\}.$$

The set of solutions $sol(c_i)$ to each primitive constraint $c_i$ is a version space defined by the borders $S(c_i)$ and $G(c_i)$. The computation of $S(c_i)$ and $G(c_i)$ for a constraint $c_i$ is conducted by the commonly used level-wise algorithm mentioned in subsection 2.5. Furthermore, $sol(c_1 \wedge ... \wedge c_n)$ to a conjunctive constraint $c_1 \wedge ... \wedge c_n$ is also characterized by $S(c_1 \wedge ... \wedge c_n)$ and $G(c_1 \wedge ... \wedge c_n)$. This $sol(c_1 \wedge ... \wedge c_n)$ is derived by the level wise version space algorithm [4].

Based on this framework, MolFea system can perform a complete search for the solutions, *w.r.t*, a conjunctive constraint consisting of monotonic and anti-monotonic primitive constraints. For example, given a set of molecule structure graph data of chemical compounds, the following constraint: $(c\text{-}o \leq f) \wedge \neg(f \leq c\text{-}o\text{-}s\text{-}c\text{-}o\text{-}s) \wedge sup(f) \geq 100$ queries for all paths $f$s embedded in the molecule structure that include (being more specific than or equal to) the subpath $c-o$ but do not include (not being more general than or not equal to) the subpath of *c-o-s-c-o-s* and have a frequency larger than 100. MolFea system has applied to the Predictive Toxicology Evaluation challenge data set [22]. This data set consists of over 300 compounds. The goal of the data mining here is to discover molecular fragments that are frequent in carcinogenetic compounds and infrequent in non-carcinogenetic compounds which are called "*structural alerts*" in toxicology. The frequency and the infrequency which are anti-monotonic and monotonic measures respectively are specified to certain numbers as indicated below and a solution for each pair of the frequency and the infrequency bounds are enumerated. Single bond and aromatic bond between two atoms are represented by $-$ and $\sim$ respectively.

$max:min$ $sup$ $sup$

6 : 0 : $G = \{c - c - c - c - o - c - c \sim c\}$
$S = \{c - c - c - c - o - c - c \sim c$
$\sim c \sim c \sim c \sim c\}$

10 : 2 : $G = \{c \sim c - c \sim c, b_r, c - o - c \sim c \sim c \sim c$
$\sim c \sim c - n, c - o - c \sim c - n\}$
$S = \{c \sim c \sim c \sim c \sim c \sim c - c \sim c \sim c \sim c$
$\sim c \sim c, b_r - c, c - o - c \sim c \sim c \sim c$
$\sim c \sim c - n, c - o - c \sim c - n\}$

16 : 5 : $G = S = \{c - c \sim c \sim c \sim c - n\}$

20 : 7 : $G = S = \{n - c \sim c \sim c \sim c \sim c \sim c - o,$
$c - c \sim c \sim c \sim c - n, n - c \sim c - o\}$

## 3.4 Mathematical Graph Theory Based Approach

The mathematical graph theory based approach mines a complete set of subgraphs under mainly support measure. The initial work is AGM (Apriori-based Graph Mining) system [11]. The basic principle of AGM is similar to the Apriori algorithm for basket analysis. Starting from frequent graphs where each graph is a single vertex, the frequent graphs having larger sizes are searched in bottom up manner by generating candidates having an extra vertex.

An edge should be added between the extra vertex and some of the vertices in the smaller frequent graph when searching for the connected graphs. One graph constitutes one transaction. The graph structured data is transformed without much computational effort into an adjacency matrix mentioned in subsection 2.3. Let the number of vertecies contained in a graph be its "*size*", an adjacency matrix of a graph whose size is $k$ be $X_k$, the $ij$-element of $X_k$, $x_{ij}$ and its graph, $G(X_k)$. AGM can handle the graphs consisting of labeled vertices and labeled edges. The vertex labels are defined as $N_p$ $(p = 1, \cdots, \alpha)$ and the edge labels, $L_q$ $(q = 1, \cdots, \beta)$. Labels of vertices and edges are indexed by natural numbers for computational efficiency. The AGM system can mine various types of subgraphs including general subgraph, induced subgraph, connected subgraph, ordered subtree, unordered subtree and subpath. Because of the space limitation, the case to mine induced subgraph is shown here. This algorithm generates association rules having support and confidence higher than user specified threshholds. In the actual implementation, the adjacency matrices

are represented by the codes defined as in subsection 2.3. According to the code, the canonical label and the canonical form are also introduced.

The candidate generation of frequent induced subgraph is done as follows. It is also outlined in Fig. 4. Two frequent graphs are joined only when the following conditions are satisfied to generate a candidate of frequent graph of size $k$+1. Let $X_k$ and $Y_k$ be adjacency matrices of two frequent graphs $G(X_k)$ and $G(Y_k)$ of size $k$. 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 $Z_{k+1}$ as follows

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

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

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. The elements $z_{k,k+1}$ and $z_{k+1,k}$ represent an edge label between $k$-th vertices of $X_k$ and $Y_k$. Their values are mutually identical because of the diagonal symmetry of the undirected graph. Here, the elements $z_{k,k+1}$ and $z_{k+1,k}$ of the adjacency matrix $Z_{k+1}$ are not determined by $X_k$ and $Y_k$. In case of an undirected graph, two possible cases are considered in which 1) there is an edge labeled $L_q$ between the $k$-th vertex and the $k+1$-th vertex of $G(Z_{k+1})$ and 2) there is no edge among them. This is indicated in Fig. 4. Accordingly $\beta+1$ adjacency matrices whose $(k, k+1)$-element and $(k+1, k)$-element are "0" and "$L_q$" are generated. $X_k$ and $Y_k$ are called the first matrix and the second matrix to generate $Z_{k+1}$ respectively. Because the labels of the $k$-th nodes of $X_k$ and $Y_k$ are the same, switching $X_k$ and $Y_k$, *i.e.*, taking $Y_k$ as the first matrix and $X_k$ as the second matrix, produces redundant adjacency matrices. In order to avoid this redundancy, the two adjacency matrices are joined only when the following condition is satisfied.

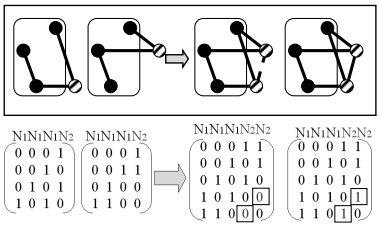


Figure 4: Candidate generation of frequent subgraph.

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

The adjacency matrix generated under these constraints is a "*normal form*". The graph $G$ of size $k$+1 is a candidate frequent graph only when adjacency matrices of all induced subgraphs whose size are $k$ are confirmed to be frequent graphs. If any of the induced subgraphs of $G(Z_{k+1})$ is not frequent, $Z_{k+1}$ is not a candidate frequent graph, because any induced subgraph of a frequent graph must be a frequent graph due to the anti-monotonicity of the support. This check to use only the former result of the frequent graph mining is done without accessing the graph data set. After the generation of candidate subgraphs, their support is counted by accessing the data set. To save computation for the counting, the graphs in the data set are represented in normal form matrices, and each subgraph matching is made between their normal forms. This technique significantly increases the matching efficiency. The process continues in level-wise manner until no new frequent induced subgraph is discovered.

This approach has been applied to the analysis of the association of the molecule substructures of chemical compounds with their mutagenesis activity. The data was drawn from a chemistry journal of A. K. Debnath et al. [5]. The data contains 230 aromatic and heteroaromatic nitro compounds. The percentages of the molecule having the classes of high, medium, low and inactive mutagenicity are 15.2%, 45.7%, 29.5% and 9.6% respectively. In the data preprocessing stage, numerical attributes of the chemical compounds of LogP and LUMO were discretized into symbolic magnitudes, converted to an isolated vertex, and added to each molecule structure.

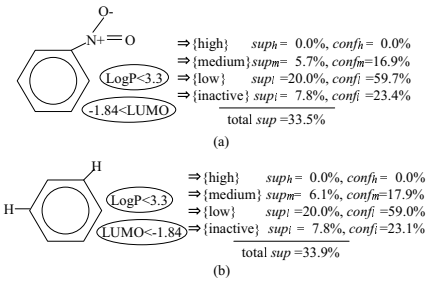


Figure 5: Examples of low activity substructures.

Figure 5 is the discovered molecular substructure indicating low activity, and Fig. 6 contains the ones indicating high activity. The deviations of the activity distributions from the original distribution are clear for both cases, and their significances has been confirmed via $\chi^2$-test. After the proposal of AGM, a family of graph-based data mining based on similar principles has been proposed. A work is FSG (Frequent SubGraph discovery) system [15] which also takes similar definition of canonical labeling of graphs based on the adjacency matrix. To increase the efficiency of deriving the canonical labels, the approach uses some graph vertex invariants such as the degree of each vertex in the graph. FSG also increases the efficiency of the candidate generation of frequent subgraphs by introducing the transaction ID (TID) method. Furthermore, FSG limits the class of the frequent subgraphs to connected graphs. Under this limitation, FSG introduces an efficient search algorithm using "*core*" which is a shared part of the size $k-1$ in the two frequent subgraphs of the size $k$. FSG increases the joining efficiency by limiting the common part of the two frequent graphs to the core. Once the candidate set is obtained, their frequency counting is conducted by checking the cardinality of the intersection of both TID lists. FSG runs fast due to the introduction of many techniques, but it consumes much memory space to store TID lists for massive graph data.

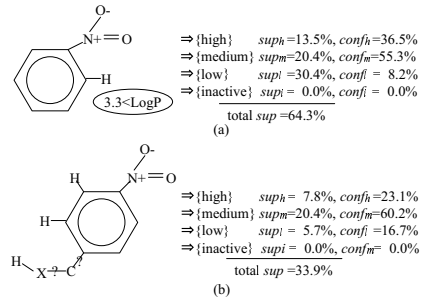


Figure 6: Examples of high activity substructures.

More recently, DFS based canonical labeling approach called gSpan (graph-based Substructure pattern mining) has been proposed [24]. This approach also uses the idea of canonical labeling which is derived from a coding scheme of a graph representation. The main difference of the coding from the other approach is that it uses a tree representation of each graph instead of the adjacency matrix to define the code of the graph as depicted in Fig. 7. Given a graph (a), various quasi-tree expressions of the graph exist depending on the way to take a root vertex among the vertices. (b), (c) and (d) are the examples where the least number of edges to remove all cyclic paths are represented by dashed lines. Upon this representation, starting from the root vertex, the code is generated by following the lexicographical order of the labels for the combinations of vertices and the edge bound by the vertices. For example, the combinations of $(v_0, v_1)$ with the label $(v_0, v_1, X, a, Y)$ comes first in the code because this is younger than dashed $(v_0, v_2)$ having $(v_0, v_2, X, a, X)$. Next, starting from $v_1$ which is the last vertex of the previous code element, the youngest code element $(v_1, v_2, Y, b, X)$ is chosen. Then from the last $v_2$, the element $(v_2, v_0, X, a, X)$ is chosen. When this trace returns to a vertex which is involved in the previous code element, the trace backtracks by one step, and the next younger element starting from $v_2$, *i.e.*, $(v_2, v_3, X, c, Z)$ is chosen. The subsequent elements $(v_3, v_1, Z, b, Y)$ and $(v_1, v_4, Y, d, Z)$ are traced in a recursive manner. The sequence of these elements is called a code of this quasi-tree expression. The other quasi-tree expressions including (c) and (d) have their own codes respectively. Among the codes, the quasi-tree expression having the minimum code in terms of the lexicographical order is the canonical form, and the corresponding code is the canonical label. Because the code is derived in the DFS algorithm, this code is called DFS code. Every graph in a data set is represented by the multiple codes in this manner. Then all codes are sorted according to ascending lexicographical order, and the matching of the codes starting from the first elements among the codes are conducted by using DFS in the sorted order. This means that the search trees of the DFS matching are ordered trees where the left branch is always younger than the right branch. Accordingly, when the branch representing a subgraph which is identical to the subgraph previously visited in the ordered tree search are found, the further DFS in the search tree can be pruned. By applying this DFS coding and DFS search, gSpan can derive complete set of frequent subgraphs over a given *minsup* in a very efficient manner in both computational time and memory consumption.

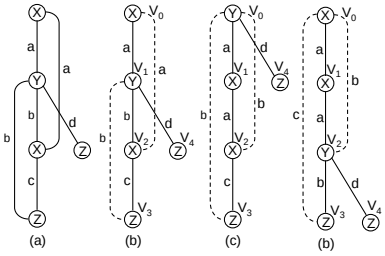

Figure 7: DFS coding of a graph.

## 3.5 Kernel Function Based Approach

As explained in the last paragraph in subsection 2.3, a kernel function $K$ defines a similarity between two graphs $G_x$ and $G_y$. For the application to graph-based data mining, the key issue is to find the good combinations of the feature vector $X_G$ and the mapping $\phi : X_G \rightarrow H$ to define appropriate similarity under abstracted inner product $< \phi(X_{Gx}), \phi(X_{Gy}) >$. A recent study proposed a composition of a kernel function characterizing the similarity between two graphs $G_x$ and $G_y$ based on the feature vectors consisting of graph invariants of vertex labels and edge labels in the certain neighbor area of each vertex [13]. This is used to classify the graphs into binary classes by SVM mentioned in subsection 2.5. Given training data consisting of graphs having binary class, the SVM is trained to classify each graph. Though the similarity is not complete and sound in terms of the graph isomorphism, the graphs are classified properly based on the similarity defined by the kernel function.

Another framework of kernel function related with graph structures is called "*diffusion kernel*" [14]. Though this is not dedicated to graph-based data mining, each instance is assigned to a vertex in a graph structure, and the similarity between instances is evaluated under the diffusion process along the edges of the graph. Some experiments report that the similarity evaluation in the structure characterizing the relations among the instances provides better performance in classification and clustering tasks than the distance based similarity evaluation. This type of work is supposed to have some theoretical relation with graph-based data mining [7].

# 4. DISCUSSION AND SUMMARY

There are many other studies related to graph mining. Geibel and Wysotzki proposed a method to derive induced subgraphs of graph data and to use the induced subgraphs as attributes on decision tree approaches [9]. Their method can be used to find frequent induced subgraphs in the set of graph data. However, the upper limit number of the vertices to be included in the subgraph must be initially specified to avoid the exponential explosion of the computational time, and thus the search is not complete. Liquiere and Sallantin proposed a method to completely search homomorphically equivalent subgraphs which are the least general over a given set of graphs and do not include any identical triplet of the labels of two vertices and the edge direction between the vertices within each subgraph [16]. They show that the computational complexity to find this class of subgraphs is polynomial for 1/2 locally injective graphs where the labels of any two vertices pointing to another common node or pointed from another common vertex are not identical. However, many graphs appearing in real-world problems such as chemical compound analysis are more general, and hence the polynomial characteristics of this approach do not apply in real cases. In addition, this approach may miss many interesting and/or useful subgraph patterns since the homomorphically equivalent subgraph is a small subclass of the general subgraph.

In this review article, the theoretical basis of the graph-based data mining was explained from multiple point of views such as subgraph types, subgraph isomorphism problem, graph invariants, mining measures and search algorithms. Then, representative graph-based data mining approaches were shown in the latter half of this article. Even from theoretical perspective, many open questions on the graph characteristics and the isomorphism complexity remain. This research field provides many attractive topics in both theory and application, and is expected to be one of the key fields in data mining research.

## 5. ACKNOWLEDGEMENTS

The authors express our deep appreciation to the reviewers and editors of this article.

## 6. REFERENCES

[1] MRDM'01: Workshop multi-relational data mining. In conjunction with PKDD'01 and ECML'01, 2002. http://www.kiminkii.com/mrdm/.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB'94: Twentyth Very Large Dada Base Conference*, pages 487–499, 1994.

[3] J. Cook and L. Holder. Substructure discovery using minimum description length and background knowledge. *J. Artificial Intel. Research*, 1:231–255, 1994.

[4] L. De Raedt and S. Kramer. The levelwise version space algorithm and its application to molecular fragment finding. In *IJCAI'01: Seventeenth International Joint Conference on Artificial Intelligence*, volume 2, pages 853–859, 2001.

[5] A. Debnath, R. De Compadre, G. Debnath, A. Schusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *J. Medicinal Chemistry*, 34, 1991.

[6] L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, 1999.

[7] T. Gaertner. A survey of kernels for structured data. *SIGKDD Explorations*, 5(1), 2003.

[8] W. Geamsakul, T. Matsuda, T. Yoshida, H. Motoda, and T. Washio. Classifier construction by graph-based induction for graph-structured data. In *PAKDD'03: Proc. of 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining, LNAI2637*, pages 52–62, 2003.

[9] P. Geibel and F. Wysotzki. Learning relational concepts with decision trees. In *ICML'96: 13th Int. Conf. Machine Learning*, pages 166–174, 1996.

[10] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58–64, 1996.

[11] A. Inokuchi, T. Washio, and H. Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50:321–354, 2003.

[12] I. Jonyer, L. Holder, and D. Cook. Concept formation using graph grammars. In *Workshop Notes: MRDM 2002 Workshop on Multi-Relational Data Mining*, pages 71–792, 2002.

[13] H. Kashima and A. Inokuchi. Kernels for graph classification. In *AM2002: Proc. of Int. Workshop on Active Mining*, pages 31–35, 2002.

[14] R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input space. In *ICML'02: Nineteenth International Joint Conference on Machine Learning*, pages 315–322, 2002.

[15] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *ICDM'01: 1st IEEE Conf. Data Mining*, pages 313–320, 2001.

[16] M. Liquiere and J. Sallantin. Structural machine learning with galois lattice and graphs. In *ICML'98: 15th Int. Conf. Machine Learning*, pages 305–313, 1998.

[17] H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. In *2nd Intl. Conf. Knowledge Discovery and Data Mining*, pages 146–151, 1996.

[18] B. Mckay. Nauty users guide (version 1.5). Technical Report Technical Report, TR-CS-90-02, Department of computer Science, Australian National University, 1990.

[19] A. Mendelzon, A. Mihaila, and T. Milo. Querying the world wide web. *Int. J. Digit. Libr.*, 1:54–67, 1997.

[20] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *J. Logic Programming*, 19(20):629–679, 1994.

[21] S. Nijssen and J. Kok. Faster association rules for multiple relations. In *IJCAI'01: Seventeenth International Joint Conference on Artificial Intelligence*, volume 2, pages 891–896, 2001.

[22] A. Srinivasan, R. King, and D. Bristol. An assessment of submissions made to the predictive toxicology evaluation challenge. In *IJCAI'99: Proc. of 16th International Joint Conference on Artificial Intelligence*, pages 270–275, 1999.

[23] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, New York., 1995.

[24] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *ICDM'02: 2nd IEEE Conf. Data Mining*, pages 721–724, 2002.

[25] K. Yoshida, H. Motoda, and N. Indurkhya. Graph-based induction as a unified learning framework. *J. of Applied Intel.*, 4:297–328, 1994.

[26] M. Zaki. Efficiently mining frequent trees in a forest. In *8th Intl. Conf. Knowledge Discovery and Data Mining*, pages 71–80, 2002.