Extracting Discriminative Patterns from Graph Structured Data using Constrained Search

Kiyoto Takabayashi[1], Phu Chien Nguyen[1], Kouzou Ohara[1], Hiroshi Motoda[2], and Takashi Washio[1]

[1] I.S.I.R., Osaka University,
8-1, Mihogaoka, Ibaraki, Osaka, 567-0047, Japan
{kiyoto_ra,chien,ohara,washio}@ar.sanken.osaka-u.ac.jp
[2] AFOSR/AOARD
7-23-17, Roppongi, Minato-ku, Tokyo 106-0032, Japan
hiroshi.motoda@aoard.af.mil

Abstract. A graph mining method, Chunkingless Graph-Based Induction (Cl-GBI), finds typical patterns that appear in a graph structured data by the operation called chunkingless pairwise expansion which generates pseudo-nodes from selected pairs of nodes in the data. Cl-GBI enables to extract overlapping subgraphs, while its time and space complexities could be extremely high. Thus, it happens that Cl-GBI cannot extract patterns that need be large enough to describe characteristics of data within a limited time and a given computational resource. In such a case, extracted patterns may not be so much of interest for domain experts. To mine more discriminative patterns which cannot be extracted by the current Cl-GBI, we introduce a search algorithm in which patterns to be searched are guided by domain knowledge or interests of domain experts. We further experimentally show that the proposed method can efficiently extract more discriminative patterns using a real world dataset.

1 Introduction

Over the last decade, there has been much research work on data mining which intend to find useful and interesting knowledge from massive data on computers. Especially on mining frequent patterns from graph structured data, or simply graph mining, a number of studies have been made in recent years because of the high expressive power of graph representation [1, 13, 6, 12, 4, 5].

Chunkingless Graph Based Induction (Cl-GBI) [8] is one of the latest algorithms in graph mining and an extension of Graph Based Induction (GBI) [13] that can extract typical patterns from graph structured data by stepwise pair expansion, i.e., by recursively chunking two adjoining nodes. Similarly to GBI and its another extension, Beam-wise (B-GBI) [6], Cl-GBI extracts typical patterns based on the stepwise pair expansion principle, but never chunks adjoining nodes. Instead, Cl-GBI regards a pair of adjoining nodes as a *pseudo node* and assigns a new label to it. This operation is called *pseudo-chunking* and can fully solve the problems caused by chunking, i.e., ambiguity in selecting nodes to chunk and incompleteness of the search. This is because every node is available to make a new pseudo node at any time in Cl-GBI. However Cl-GBI sacrificed its time and space complexities in gaining the ability of extracting overlapping patterns. Thus, it happens that Cl-GBI cannot extract patterns that need be large enough to describe characteristics of data within a limited time and a given computational resource. In such a case, extracted patterns may not be so much of interest for domain experts.

From this background, in this paper, we propose a method of guiding the search of Cl-GBI using domain knowledge or interests of domain experts. The basic idea is adopting patterns representing knowledge or interests of domain experts as constraints on the search, in order to effectively restrict the search space and extract more discriminative or interesting patterns than those which can be extracted by the current Cl-GBI. For that purpose, we use two types of patterns as the constraints: one is the pattern that should be included in the extracted ones, and the other is the pattern that should not be included in them. These patterns allow us to specify patterns of interest and patterns trivial for domain experts, respectively. We also experimentally show the effectiveness of the proposed search method by applying the constrained Cl-GBI to the hepatitis dataset which is a real world dataset.

In this paper, we deal with only connected labeled graphs, and use information gain [10] as the discriminativity criterion. In addition, "a pair" denotes a pair of adjoining nodes in a graph.

2 Chunkingless Graph-Based Induction(Cl-GBI)

Stepwise pair expansion is an essential operation in GBI and its variants, which recursively generates new nodes from pairs of two adjoining nodes and links between them. In GBI, a pair is selected according to a certain criterion based on frequency, and all of its occurrences in graphs are replaced with a node having a newly assigned label. Namely each graph is rewritten each time a pair is chunked, and never restored in any subsequent chunking. On one hand, this chunking mechanism is suitable for extraction of patterns from either a very large single graph or graph database because extracted patterns can rapidly grow. On the other hand, it involves ambiguity in selecting nodes to chunk, which causes a crucial problem, i.e., possibility of overlooking important subgraphs due to inappropriate chunking order. Beam search adopted by B-GBI can alleviate this problem by chunking the b (beam width) most frequent pairs and copying each graph into respective states, but not completely solve it because chunking process is still involved.

In contrast to GBI and B-GBI, Cl-GBI does not chunk a selected pair, but regards it as a *pseudo node* and assigns a new label to it. Thus, graphs are not "compressed" nor copied into respective states. Figure 1 illustrates examples of *pseudo-chunking* in Cl-GBI, in which a typical pattern consisting of nodes 1, 2, and 3 is extracted from the input graph. Cl-GBI first finds the pair $1 \rightarrow 3$ based

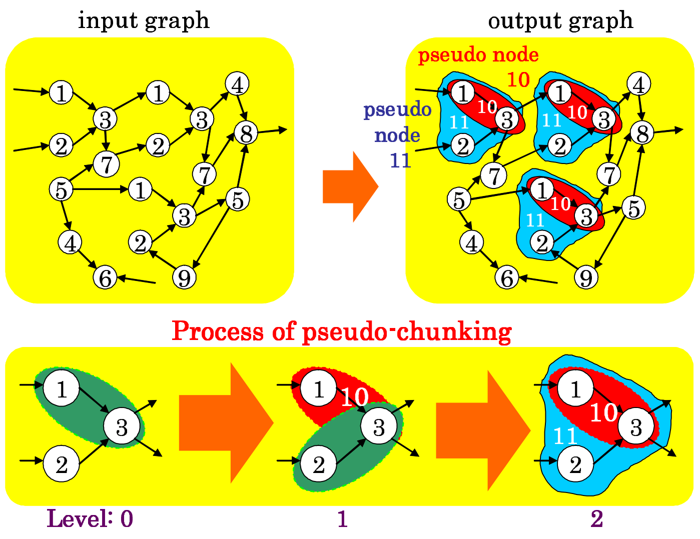

Fig. 1. An Example of Pseudo-chunking in Cl-GBI

on its frequency, and pseudo-chunks it, i.e., registers it as the pseudo node 10, but does not rewrite the graph. Then, in the next iteration, it finds the pair $2 \rightarrow 10$, and pseudo-chunks and registers it as the pseudo-node 11. As a result, the typical pattern is extracted. In the rest of the paper, we refer to each iteration in Cl-GBI as "level".

The algorithm of Cl-GBI is shown in Fig.2. The search of Cl-GBI is controlled by the following parameters: a beam width b, the maximal number of levels of pseudo-chunking N, and a frequency threshold θ. In other words, at each level, the b most frequent pairs are selected from a set of pairs whose frequencies are not less than θ, and are pseudo-chunked.

3 Constrained Search for Cl-GBI

3.1 Patterns Used as Constraints

The current Cl-GBI blindly extracts a huge number of frequent pairs without any clues other than frequency and selects pairs to pseudo-chunk from among them. However, if the goal is finding patterns which are either discriminative or of interest for domain experts, the current method of extracting pairs is too naive and inefficient in both time and space complexities. This is because such patterns are not always frequent in a database. If discriminative patterns are not so frequent in a database and there are a large number of patterns that are more frequent than them, it is difficult to extract such discriminative patterns within a limited time and a given computational resource.

Note that such a goal, i.e., finding discriminative/interesting patterns is not special. In DT-ClGBI [9] which constructs a decision tree from graph structured

Input. A graph database D, a beam width b, the maximal number of levels of pseudo-chunking N, a frequency threshold θ

Output. A set of typical patterns S

Step 1. Extract all the pairs consisting of two connected nodes in the graphs, register their positions using node id (identifier) sets. From the 2nd level on, extract all the pairs consisting of two connected nodes with at least one node being a new pseudo-node.

Step 2. Count frequencies of extracted pairs and eliminate pairs whose frequencies count below θ.

Step 3. Select the b most frequent pairs from among the remaining pairs at Step 2 (from the 2nd level on, from among the unselected pairs in the previous levels and the newly extracted pairs). Each of the b selected pairs is registered as a new node. If either or both nodes of the selected pair are not original but pseudo-nodes, they are restored to the original patterns before registration.

Step 4. Assign a new label to each pair selected at Step 3 but do not rewrite the graphs. Go back to Step 1.

Fig. 2. Algorithm of Cl-GBI

data, Cl-GBI is adopted to extract discriminative patterns used as test nodes in a tree. When we analyzed the hepatitis dataset [11] provided by Chiba University Hospital with Cl-GBI, domain experts (medical doctors) expected that patterns that are interesting for them were extracted, but in fact we could not find satisfactory ones with the current Cl-GBI.

Therefore, in this paper, we introduce domain knowledge or interests of domain experts and impose them as constraints on patterns that are extracted in Cl-GBI in order to efficiently extract discriminative/interesting patterns which the current Cl-GBI could not extract within a limited time and a given computational resource. We represent such domain knowledge and interests of domain experts as graphs, or patterns, and call them the *constraint patterns.*

Although various types of constraint patterns can be considered, in this paper, we focus on the following two types of patterns: one is the pattern that should be included in extracted patterns, and the other is the pattern that should not be included in them. We refer to the former type as the *INpattern*, and the latter as the *EXpattern.* Thus, the constraints we introduce in this paper are defined as follows:

Constraint 1. Extracted patterns must include INpatterns.
Constraint 2. Extracted patterns must not include EXpatterns.

Figures 3 (a) and (b) show the examples of Constraints 1 and 2, respectively. Note that in case of Constraint 1, not only patterns including the given INpatterns, but also patterns including at least one proper subgraph of the INpatterns should be extracted in order to extract patterns satisfying the imposed constraints based on the stepwise pair expansion principle. The case is illustrated

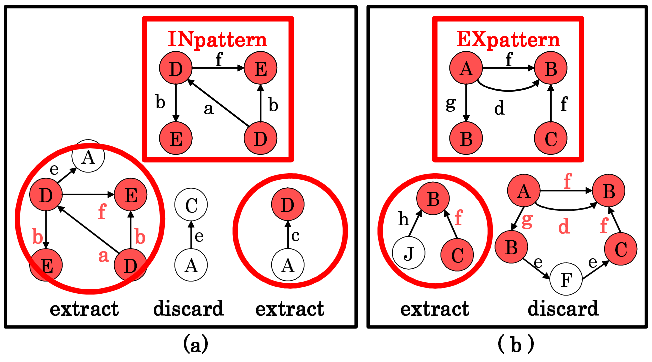

Fig. 3. Examples of INpatterns and EXpatterns

at the bottom right in Fig.3 (a). We call such a pattern including only proper subgraphs of the given INpatterns a *neighborhood pattern*. In other words, an INpattern does not necessarily have to be identical to a pattern representing domain knowledge or interests of domain experts, but need merely include at least one of its proper subgraphs. Namely, Constraint 1 is useful to specify patterns of interest for domain experts and to aggressively search around them, while Constraint 2 is useful to avoid extracting trivial or boring patterns for domain experts.

In addition, in case of Constraint 1, we can discard a pair if it does not include any node/link labels appearing in the given INpatterns as shown in Fig.3 (a). This is because such a pair, or pattern is unnecessary to make a pattern that includes at least one of the given INpatterns. Even if in fact the discarded pattern is a subgraph of a pattern P satisfying all constraints and a given evaluation criterion such as the minimal frequency, P can be constructed from another pattern including at least one node/link label appearing in the given INpatterns. In other words, enumerating only the pairs with at least one node/link label in the given INpatterns as candidates to be pseudo-chunked allows us to effectively restrict the search space. However, it is noted that a label with high frequency does not effectively work as a constraint because it may appear also in many pairs which the user intends to exclude from the search space. Thus, if a set of node (link) labels contains such frequent ones, we do not use the set to restrict pairs in the enumeration process.

3.2 Design of Constrained Search

To guide the search process of Cl-GBI using either INpatterns or EXpatterns, we have to check if an enumerated pair includes them, which requires additional subgraph isomorphism checking. Since subgraph isomorphism checking is known to be NP-complete [2], it is obvious that a naive approach could be computationally expensive. To reduce the computational cost as much as possible, we should detect pairs that have no possibility of including constraint patterns before the checking. For that purpose, we focus on the quantities that characterize the topological structure of a graph, especially the number of node/link labels in patterns.

First of all, for two arbitrary pairs, or patterns x and y, we define a quantity T_{num} as follows:

$$T_{num}(x, y) = \sum_{L_k \in L(y)} f(x, L_k), \tag{1}$$

where $L(y)$ is a set of labels appearing in y, and $f(x, L_k)$ is the number of occurrences of the label $L_k \in L(y)$ in x. Note that if x is identical to y, $T_{num}(x, y)$ must be equal to $T_{num}(y, y)$. Similarly, $T_{num}(x, y)$ must be greater than $T_{num}(y, y)$ if y is a subgraph of x. Consequently, we can skip subgraph isomorphism checking for the pair of an enumerated pattern P_i and a constraint pattern T_j if $T_{num}(P_i, T_j) < T_{num}(T_j, T_j)$ because P_i never includes T_j.

Furthermore, we can prune more subgraph isomorphism checking even if $T_{num}(P_i, T_j) \geq T_{num}(T_j, T_j)$ because it does not guarantee that P_i necessarily includes T_j. In order for P_i to include T_j, for every label appearing in T_j, the number of its occurrences in P_i has to be greater than or equal to that in T_j. Namely, we can skip subgraph isomorphism checking for P_i if P_i does not satisfy this condition. To check this condition, we define the following boolean value P_{info} for two patterns x and y.

$$P_{info}(x, y) = \bigwedge_{L_k \in L(y)} p(x, y, L_k), \tag{2}$$

where

$$p(x, y, L_k) = \begin{cases} true & \text{if } f(x, L_k) \geq f(y, L_k), \\ false & \text{otherwise.} \end{cases}$$

If $P_{info}(P_i, T_j)$ is $true$, then subgraph isomorphism checking has to be done; otherwise it can be skipped.

These ideas discussed above are summarized in Fig.4 as the algorithm that extracts pairs, which is embedded, or invoked at **Step 1** in the algorithm shown in Fig.2, and provides a set of candidate pairs to be pseudo-chunked at each level. As input, a graph database D, a set of constraint patterns T consisting of either INpatterns or EXpatterns, a parameter L_v specifying the current level, and a list of pairs L consisting of patterns that have been extracted before are given. Then it outputs L adding newly extracted pairs. $PD(P_i, T_j)$ in this algorithm is the procedure for subgraph isomorphism checking, which returns true if P_i includes T_j; otherwise false. In fact, it applies Cl-GBI to a graph database consisting of only P_i and T_j after deleting all nodes and links that never appear in T_j from P_i. By running Cl-GBI without the constraint patterns until T_j is extracted and checking the occurrences of T_j in P_i, one can detect whether P_i includes T_j or not.

```
**ExtPair**($D$, $T$, $L$, $L_v$, $M$)
**Input**: a database $D$, a set of constraint patterns $T$, the current level $L_v$,
           a set of extracted pairs $L$ (initially empty),
           the constraint mode $M$ (either “INpattern” or “EXpattern”);
**Output**: a set of extracted pairs $L$ with newly extracted pairs;
begin
  if $L_v = 1$ then
    if $M =$ “INpattern” then
      Enumerate pairs in $D$, which consist of nodes or links
      appearing in $T$, and store them in $E$;
    else
      Enumerate all the pairs in $D$ and store them in $E$;
  else
    Enumerate pairs, which consist of one or both
    pseudo nodes in $L$, and store them in $E$;
  for each $P_i \in E$ begin
    if $P_i$ is marked then
      $L := L \cup \{P_i\}$; next;
    $register := 1$;
    for each $T_j \in T$ begin
      if $T\_num(P_i, T_j) \geq T\_num(T_j, T_j)$ then
        if $P_{info}(P_i, T_j) =$ true then
          if $M =$ “INpattern” then
            if $PD(P_i, T_j) =$ true then mark $P_i$;
          else
            if $PD(P_i, T_j) =$ true then
              discard $P_i$; $register := 0$; break;
    end
    if $register = 1$ then $L := L \cup \{P_i\}$;
  end
  return $L$;
end
```

Fig. 4. Algorithm of the constrained pattern extraction

4 Experimental Evaluation

4.1 Experimental settings

To evaluate the proposed method, we implemented Cl-GBI with the algorithm shown in Fig.4 on PC (CPU: Pentium 4 3.2GHz, Memory: 4GB, OS: Fedora Core release 3) in C++, and applied this *constrained Cl-GBI* to the hepatitis dataset. At preset, only either INpatterns or EXpatterns is available at a time. In this experiment, we used two classes, *Response* and *Non-Response*, in the dataset, denoted by R and N, respectively. R consists of patients to whom the interferon therapy was effective, while N consists of those to whom it was not effective. We used 24 inspection items as attributes, and converted the records

Table 1. Size of graphs of the hepatitis dataset

class	R	N
number of graphs	38	56
average number of nodes in a graph	104	112
maximal number of nodes in a graph	145	145
minimum number of nodes in a graph	24	20
total number of nodes	3,944	6,296
kinds of node labels	12	
average number of links in a graph	108	117
maximal number of links in a graph	154	154
minimum number of links in a graph	23	19
total number of links	4,090	6,577
kinds of link labels	30	

of each patient into a graph in the same way as [3]. The statistics on the size of resulting graphs are shown in Table. 1.

In this experiment, we used 4 sets of INpatterns shown in Fig.5, in which (a) to (c) are patterns reported in [7] and represent typical examination results for patients belonging to R, while (d) is the pattern with the highest information gain, or the most discriminative pattern among ones extracted by the current Cl-GBI. In the following, we refer to the most discriminative pattern as the *MDpattern*. The node with the label "d" in Fig.5 is a dummy node representing a certain point of time. For example, the leftmost pattern in Fig.5 (a) means that at a certain point of time, the value of GPT is High and the value of PLT is Low. Note that node labels in this dataset such as "d", "H", etc. are common and may appear with large frequency. Thus, we used only the link labels appearing in the INpatterns as constraints for the pair enumeration as discussed above. As for the parameters of Cl-GBI, we set them as follows: b = 10, N = 10, and $\theta = 0\%$.

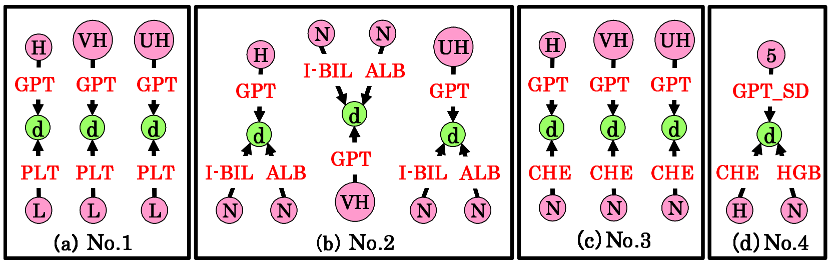

Fig. 5. INpatterns used in the experiments

Table 2. Experimental results

	time[sec]	max information gain
original	44,973 (—)	0.1139
No.1	9,355 (66,211)	0.1076
No.2	6,893 (31,527)	0.1698
No.3	20,495 (159,434)	0.1110
No.4	4,970 (14,923)	0.1297

4.2 Experimental Results

We gave each set of patterns shown in Fig.5 as INpatterns to the constrained Cl-GBI, and observed the computation time, the MDpattern, and its information gain in each case. The results regarding the computation time and information gain of the MDpatterns are shown in Table. 2, in which the row named "original" contains the results by the current Cl-GBI with the same parameter settings. Namely, the MDpattern shown in Fig.5 (d) corresponds to the MDpattern in the case of "original", and its information gain is 0.1139. The resulting MDpatterns obtained by the constrained Cl-GBI are illustrated in Fig. 6.

First, focusing on the column of "max information gain" in Table 2, it is found that the MDpatterns extracted by the constrained Cl-GBI are more discriminative than the MDpattern by the current Cl-GBI in the cases of No.2 and No.4. In the cases of No.1 and No.3, the values of information gain of the MDpatterns are compatible with that of the MDpattern extracted by the current Cl-GBI. In addition, the computation times in all 4 cases using INpatterns are much less than in the case of the current Cl-GBI. Note that the values in parentheses in the column "time" of Table 2 are corresponding computation times by the constrained Cl-GBI without pruning subgraph isomorphism checking. Comparing with computation times in the same row, it could be said that checking if pairs include constraint patterns based on the number of nodes/links could reduce the computation time significantly. From these results, we can say that given appropriate constraints, the constrained Cl-GBI could efficiently extract

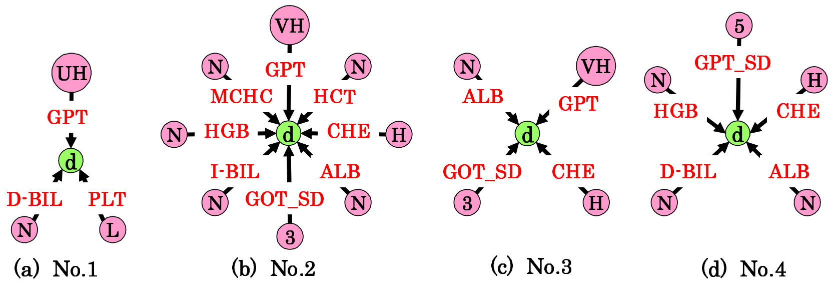

Fig. 6. MDpatterns Extracted by the constrained Cl-GBI

patterns which are more discriminative than those which are extracted by the current Cl-GBI.

Next, as shown in Fig.6, except in the case of No.3, the MDpatterns extracted by the constrained Cl-GBI include one of the given INpatterns completely. In the case of No.3, the MDpattern includes a subgraph of one of the INpatterns. Thus, it is expected that the proposed constrained search method may work well even if it is not sure that given INpatterns are genuinely appropriate ones. This is because it can extract not only patterns completely including them, but also the neighborhood patterns.

In addition, note that the INpattern used in the case of No.4 is the MDpattern obtained by the current Cl-GBI with the same parameter settings, and works as a good constraint, succeeding in extracting more discriminative patterns. From this result, it is expected that MDpatterns obtained before may work as good constraints to guide the search, which would be desirable if no domain knowledge is available to restrict the search space: in such a case, instead of running the current Cl-GBI only once setting the maximal level L to a large value, repeatedly running the constrained Cl-GBI setting L to a smaller value and using the MDpattern extracted by the previous run as the new INpattern might allow us to extract patterns that are more discriminative in a less computation time. Verifying this expectation is one of our future work.

5 Conclusion

In this paper, we proposed a constrained search method that effectively restricts the search space of Cl-GBI by imposing domain knowledge or interests of domain experts as constraints on patterns to be searched, and embedded it in Cl-GBI, resulting in the constrained Cl-GBI. The proposed method avoids conducting subgraph isomorphism checking as much as possible based on the number of node/link labels in patterns because it is computationally expensive. Experimental results showed that if given constraints are appropriate, the constrained Cl-GBI can extract more discriminative patterns in a less computation time than the current Cl-GBI. In addition, the results also showed the possibility that discriminative patterns extracted before may work as good constraints in the constrained Cl-GBI and contribute to extracting more discriminative patterns. It is worth saying that the basic idea of imposing constraints on extracting patterns in graph mining does not rely on how to construct candidate patterns. Namely, this approach could apply to any grpah mining method if the patterns were constructed step by step.

As future work, we plan to provide more flexible ways to give constraints such as a combination of INpatterns and EXpatterns, and further examine the effect of reusing the MDpatterns extracted before as new INpatterns. We will also evaluate the effectiveness of using EXpatterns, as well as the usefulness of the constrained Cl-GBI as a feature construction method in DT-ClGBI.

References

1. Cook, D. J. and Holder, L. B.: Substructure Discovery Using Minimum Description Length and Background Knowledge. Artificial Intelligence Research, Vol. 1, pp. 231–255, (1994).
2. Fortin, S.: The Graph Isomorphism Problem. Technical Report TR96-20, Department of Computer Science, University of Alberta, (1996).
3. Geamsakul, W., Yoshida, T., Ohara, K., Motoda, H., Yokoi, H., and Takabayashi, K.: Constructing a Decision Tree for Graph-Structured Data and its Applications. Fundamenta Informaticae Vol. 66, No.1-2, pp. 131–160, (2005).
4. Inokuchi, A., Washio, T., and Motoda, H.: Complete Mining of Frequent Patterns from Graphs: Mining Graph Data. Machine Learning, Vol. 50, No. 3, pp. 321–354, (2003).
5. Kuramochi, M. and Karypis, G.: An Efficient Algorithm for Discovering Frequent Subgraphs. IEEE Trans. Knowledge and Data Engineering, Vol. 16, No. 9, pp. 1038–1051, (2004).
6. Matsuda, T., Motoda, H., Yoshida, T., and Washio, T.: Mining Patterns from Structured Data by Beam-wise Graph-Based Induction. Proc. of DS 2002, pp. 422–429, (2002).
7. Motoyama, S., Ichise, R., and Numao, M.: Knowledge Discovery from Inconstant Time Series Data. JSAI Technical Report, SIG-KBS-A405, pp. 27–32, in Japanese, (2005).
8. Nguyen, P. C., Ohara, K., Motoda, H., and Washio, T.: Cl-GBI: A Novel Approach for Extracting Typical Patterns from Graph-Structured Data. Proc. of PAKDD 2005, pp. 639–649, (2005).
9. Nguyen, P. C., Ohara, K., Mogi, A., Motoda, H., and Washio, T.: Constructing Decision Trees for Graph-Structured Data by Chunkingless Graph-Based Induction. Proc. of PAKDD 2006, pp. 390–399, (2006).
10. Quinlan, J. R.: Induction of decision trees. Machine Learning, Vol. 1, pp. 81–106, (1986).
11. Sato, Y., Hatazawa, M., Ohsaki, M., Yokoi, H., and Yamaguchi, T.: A Rule Discovery Support System in Chronic Hepatitis Datasets. First International Conference on Global Research and Education (Inter Academia 2002), pp. 140–143, (2002).
12. Yan, X. and Han, J.: gSpan: Graph-Based Structure Pattern Mining. Proc. of the 2nd IEEE International Conference on Data Mining (ICDM 2002), pp. 721–724, (2002).
13. Yoshida, K. and Motoda, H.: CLIP: Concept Learning from Inference Patterns. Artificial Intelligence, Vol. 75, No. 1, pp. 63–92, (1995).