Constructing Decision Trees for Graph-Structured Data by Chunkingless Graph-Based Induction

Phu Chien Nguyen, Kouzou Ohara, Akira Mogi, Hiroshi Motoda, Takashi Washio

Institute of Scientific and Industrial Research, Osaka University
8-1 Mihogaoka, Ibaraki, Osaka, 567-0047, Japan
{chien,ohara,mogi,motoda,washio}@ar.sanken.osaka-u.ac.jp

Abstract. A decision tree is an effective means of data classification from which one can obtain rules that are easy to understand. However, decision trees cannot be conventionally constructed for data which are not explicitly expressed with attribute-value pairs such as graph-structured data. We have proposed a novel algorithm, named Chunkingless Graph-Based Induction (Cl-GBI), for extracting typical patterns from graph-structured data. Cl-GBI is an improved version of Graph-Based Induction (GBI) which employs stepwise pair expansion (pairwise chunking) to extract typical patterns from graph-structured data, and can find overlapping patterns that cannot not be found by GBI. In this paper, we further propose an algorithm for constructing decision trees for graph-structured data using Cl-GBI. This decision tree construction algorithm, now called Decision Tree Chunkingless Graph-Based Induction (DT-ClGBI), can construct a decision tree from a graph-structured dataset while simultaneously constructing attributes useful for classification using Cl-GBI internally. Since patterns (subgraphs) extracted by Cl-GBI are considered as attributes of a graph, and their existence/non-existence are used as attribute values in DT-ClGBI, DT-ClGBI can be conceived as a tree generator equipped with feature construction capability. Experiments were conducted on both synthetic and real-world graph-structured datasets showing the usefulness and effectiveness of the algorithm.

1 Introduction

Over the last few years there has been much research work on data mining in seeking for better performance. Better performance includes mining from structured data, which is a new challenge. Since structure is represented by proper relations and a graph can easily represent relations, knowledge discovery from graph-structured data poses a general problem for mining from structured data.

On one hand, from this background, discovering frequent patterns of graph-structured data, i.e., frequent subgraph mining or simply graph mining, has attracted much research interest in recent years. AGM [6] and a number of other methods including AcGM [7], FSG [8], gSpan [16], etc. have been developed for the purpose of enumerating all frequent subgraphs of a graph database. However, the computation time increases exponentially with input graph size and minimum support. This is because the kernel of frequent subgraph mining is subgraph isomorphism, which is known to be NP-complete [3].

To avoid the complex subgraph isomorphism problem, heuristic algorithms, which are not guaranteed to find the complete set of frequent subgraphs, such as SUBDUE [2] and GBI (Graph-Based Induction) [17] have also been proposed. They tend to find an extremely small number of patterns based on greedy search. GBI extracts typical patterns from graph-structured data by recursively chunking two adjoining nodes. Later an improved version called B-GBI (Beam-wise Graph-Based Induction) [10] adopting the beam search was proposed to

increase the search space, thus extracting more discriminative patterns while keeping the computational complexity within a tolerant level. Since the search in GBI is greedy and no backtracking is made, which patterns are extracted by GBI depends on which pairs are selected for chunking. This means that patterns that partially overlap can no longer been extracted, and thus there can be many patterns which are not extracted by GBI. B-GBI can help alleviate this problem, but cannot solve it completely because the chunking process is still involved.

To overcome the problem of overlapping patterns incurred by GBI and B-GBI, we have proposed an algorithm to extract typical patterns from graph-structured data, called Chunkingless Graph-Based Induction (Cl-GBI)[12]. Although Cl-GBI is an improved version of B-GBI, it does not employ the pair-wise chunking strategy. Instead, the most frequent pairs are regarded as new nodes and given new node labels in the subsequent steps but none of them is chunked. In other words, they are used as pseudo nodes, thus allowing extraction of overlapping subgraphs. It was experimentally shown that Cl-GBI can extract more typical substructures than B-GBI [12].

On the other hand, a majority of methods widely used for data mining are for data that do not have structure and that are represented by attribute-value pairs. Decision trees [13, 14], and induction rules [11, 1] relate attribute values to target classes. Association rules often used in data mining also use this attribute-value pair representation. These methods can induce rules such that they are easy to understand. However, the attribute-value pair representation is not suitable to represent a more general data structure such as graph-structured data. This means that most of useful methods in data mining are not directly applicable to graph-structured data.

In this paper, we propose an algorithm to construct decision trees for graph structured data using Cl-GBI. This decision tree construction algorithm, called Decision Tree Chunkingless Graph-Based Induction (DT-ClGBI), is a revised version of our previous algorithm called Decision Tree Graph-Based Induction (DT-GBI) [4, 5], and can construct a decision tree for a graph-structured dataset while simultaneously constructing substructures used as attributes for the classification task by means of Cl-GBI instead of B-GBI adopted in DT-GBI. In this context, substructures means subgraphs or patterns that appear in a given graph database. Patterns extracted by Cl-GBI are regarded as attributes of graphs and their existence/non-existence are used as attribute values. Namely, DT-ClGBI does not require the user to define available substructures in advance. Since attributes (features) are constructed while a classifier is being constructed, DT-ClGBI can be conceived as a method for feature construction. Using synthetic and real-world graph-structured datasets, we experimentally show DT-ClGBI can construct decision trees from graph-structured data that achieve reasonably good predictive accuracy.

This paper is organized as follows: Section 2 briefly describes the framework of GBI, the problem caused by the nature of chunking in GBI, and the summary of the Cl-GBI algorithm. Section 3 explains DT-ClGBI and its working mechanism of how a decision tree is constructed using a simple example. The performance of DT-ClGBI is experimentally evaluated and reported in Section 4. Finally, Section 5 concludes the paper.

2 Graph-Based Induction Revisited

2.1 Graph-Based Induction (GBI)

GBI contracts the graph by chunking the most frequent patterns into single nodes. However, this process is not executed in one step, rather it follows a pairwise chunking (stepwise pair expansion) strategy. In the original GBI, an assumption is made that typical patterns represent

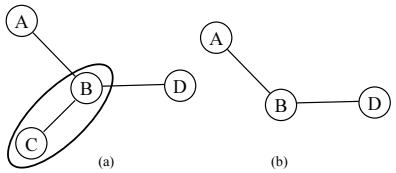

Fig. 1. Missing patterns due to chunking order

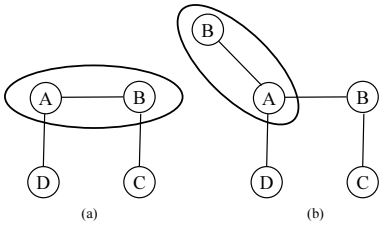

Fig. 2. A pattern is found in one input graph but not in the other

some concepts/substructures and "typicality" is characterized by the pattern's frequency or the value of some evaluation function of its frequency. We can use statistical indices as an evaluation function, such as frequency itself, Information Gain [13], Gain Ratio [14], all of which are based on frequency. It is of interest to note that GBI was improved later to use two criteria, one based on frequency measures for chunking and the other for finding discriminative patterns after chunking. The reader is referred to [17, 10] for a detailed mathematical treatment of GBI.

It is possible to extract typical patterns of various sizes by repeating this chunking process. Since the search in GBI is greedy and no backtracking is allowed, which patterns (subgraphs) are extracted depends on which pair is selected for chunking. This is because in enumerating pairs no pattern which has been chunked into one node is restored to the original pattern. Therefore, all the "typical patterns" that exist in the input graph are not necessarily extracted and patterns that partially overlap are never generated, i.e., any two patterns are either disjoint or perfect inclusion. The problem of extracting all the isomorphic subgraphs is known to be NP-complete. Thus, GBI aims at extracting only meaningful typical patterns of a certain size. Its objective is not finding all the typical patterns nor finding all the frequent patterns.

Fig. 1 shows an example of incomplete search in GBI. Here if the pair B–C is selected for chunking beforehand, there is no way to extract the substructure A–B–D even if it is a typical pattern. Moreover, any subgraph that GBI can find is along the way in the chunking process. Thus, it happens that a pattern found in one input graph is unable to be found in the other input graph even if it does exist in the graph. An example is shown in Fig. 2, where even if the pair A – B is selected for chunking and the substructure D – A – B – C exists in the input graphs, we may not find that substructure because an unexpected pair A – B is chunked (see Fig. 2(b)). This causes a serious problem in counting the frequency of a pattern.

An improved version of GBI, called Beam-wise Graph-Based Induction (B-GBI), adopting a beam search was proposed to relax the problem of overlapping subgraphs described above [10]. Though the beam search helps increase the search space, thus resulting in more discriminative patterns extracted by B-GBI than GBI, it cannot help solve this problem completely because the chunking process is still involved.

2.2 Chunkingless Graph-Based Induction (Cl-GBI)

We have introduced an algorithm, named Chunkingless Graph-Based Induction (Cl-GBI) [12], to cope with the aforementioned problem of overlapping subgraphs incurred by both GBI and B-GBI. Cl-GBI employs a "chunkingless chunking" strategy, where frequent pairs are never chunked but used as pseudo nodes in the subsequent steps, thus allowing extraction of overlapping subgraphs. As in B-GBI, the Cl-GBI approach can handle both directed and undirected graphs as well as both general and induced subgraphs. It can also extract typical

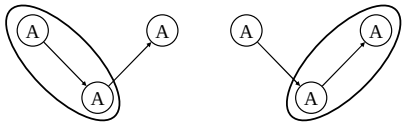

Fig. 3. Two different pairs representing identical patterns

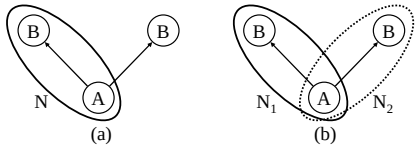

Fig. 4. Example of frequency counting

patterns in either a single large graph or a graph database. The algorithm of Cl-GBI is briefly described as follows.

Given a graph database, two natural numbers b (beam width) and N_e, and a frequency threshold θ, the new "chunkingless chunking" strategy repeats the following three steps N_e times, each of which is referred to as a level. N_e is thus the number of levels.

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

Step 2 Select the b most frequent pairs from among the pairs extracted at Step 1 (from the 2^{nd} 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 nodes but pseudo nodes, they are restored to the original patterns before registration.

Step 3 Assign a new label to each pair selected at Step 2 but do not rewrite the graphs. Go back to Step 1.

All the pairs extracted at Step 1 in all the levels (i.e. level 1 to level N_e), including those that are not used as pseudo nodes, are ranked based on a typicality criterion using a discriminative function such as Information Gain or Gain Ratio. Those pairs that have frequency count below θ are eliminated, which means that there are three parameters b, N_e, θ to control the search in Cl-GBI.

GBI assigns a new label to each newly chunked pair. Because it recursively chunks pairs, it happens that the new pairs that have different labels happen to be the same pattern as illustrated in Fig. 3. B-GBI identifies if different pairs represent the same pattern using their canonical labels [3]. Two pairs are considered to be identical only when their labels are the same. Unlike B-GBI, to count the number of occurrences of a pattern in a graph transaction, not only the canonical labeling but also the node id set is employed in Cl-GBI. If both the canonical label and the node id set are identical for two subgraphs, we regard that they are the same and count once. Without information on the node id set, it happens that the substructure N $\rightarrow$ B in Fig. 4(a) is incorrectly counted twice as shown in Fig. 4(b) due to the presence of two pseudo nodes N_1 and N_2.

The output of Cl-GBI algorithm is a set of ranked typical patterns, each of which comes together with the positions of every occurrence of the pattern in each transaction of the graph database (given by the node id sets) as well as the number of occurrences.

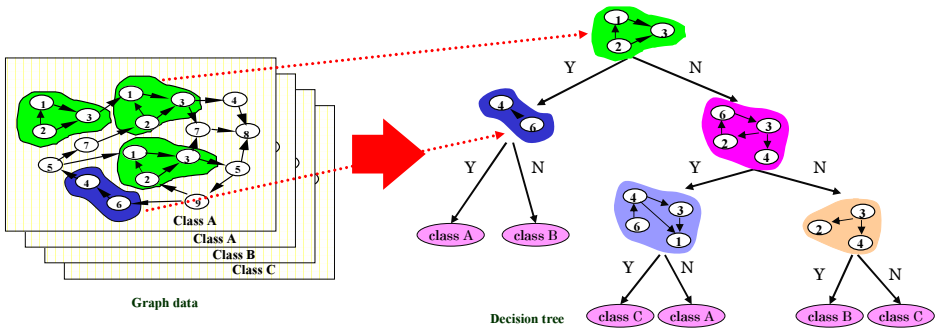

Fig. 5. Decision tree for classifying graph-structured data

3 Decision Tree Chunkingless Graph-Based Induction (DT-ClGBI)

3.1 Decision Tree for Graph-Structured Data

As mentioned in Section 1, the attribute-value pair representation is not suitable for graph-structured data, although both attributes and their values are essential for a classification or prediction task because a class is related to some attribute values in most cases. In a decision tree, each node and a branch connecting the node to its child node correspond to an attribute and one of its attribute values, respectively. Thus, to formulate the construction of a decision tree for a graph-structured dataset, we define attributes and their values as follows:

- attribute: a pattern/subgraph in graph-structured data,
- value of an attribute: existence/non-existence of the pattern in each graph.

Since the value of an attribute is either yes (the pattern corresponding to the attribute exists in the graph) or no (the pattern does not exist), the resulting decision tree is represented as a binary tree. Namely, data (graphs) are divided into two groups: one consists of graphs with the pattern, and the other consists of graphs without it. Fig. 5 illustrates the decision tree constructed based on this approach. One remaining question is how to determine patterns which are used as attributes for graph-structured data. Our approach to this question is described in the next subsection.

3.2 Feature Construction by Cl-GBI

The algorithm we propose here, called Decision Tree Chunkingless Graph-Based Induction (DT-ClGBI), utilizes Cl-GBI to extract patterns from graph-structured data and use them as attributes for a classification task, whereas our previous algorithm, Decision Tree Graph-Based Induction (DT-GBI), adopted B-GBI to extract patterns. Namely, DT-ClGBI invokes Cl-GBI at each node of a decision tree, and selects the most discriminative pattern from those which were extracted by Cl-GBI. Then the data (graphs) are divided into two groups, i.e., one with the pattern and the other without the pattern as described above. For each group, the same process is recursively applied until the group contains graphs of a single class like the ordinary decision tree construction method such as C4.5 [14]. The algorithm of DT-ClGBI is summarized in Fig. 6.

```
DT-ClGBI(D)
  INPUT
  D: a graph database
  begin
    Create a node DT for D
    if termination condition reached
      return DT
    else
      P := Cl-GBI(D) (with b, N_e, and θ specified)
      Select the most discriminative pattern p from P
      Divide D into D_y (with p) and D_n (without p)
      for D_i := D_y, D_n
        DT_i := DT-ClGBI(D_i)
        Augment DT by attaching DT_i as its child
        along yes/no branch
    return DT
  end
```

Fig. 6. Algorithm of DT-ClGBI

In DT-ClGBI, each of the parameters of Cl-GBI, b, N_e, and θ, can be set to different values at different nodes in a decision tree. All patterns extracted at a node are inherited to its descendant nodes to prevent a pattern that has already been extracted in the node from being extracted again in its descendants. This means that, as the construction of a decision tree progresses, the number of patterns to be considered at a node progressively increases, and the size of a pattern newly extracted can be larger than existing patterns. Thus, although initial patterns at the start of search consist of two nodes and the link between them, attributes useful for the classification task can be gradually grown up into larger patterns (subgraphs) by applying Cl-GBI recursively. In this sense, DT-ClGBI can be conceived as a method for feature construction, since features, i.e., attributes (patterns) useful for classification task, are constructed during the application of DT-ClGBI.

However, recursive partitioning of data until each subset in the partition contains data of a single class often results in overfitting to the training data and thus degrades the predictive accuracy of resulting decision trees. To avoid overfitting, and improve predictive accuracy, DT-ClGBI incorporates "pessimistic pruning" used in C4.5 [14] that prunes an overfitted tree based on the confidence interval for binomial distribution. This pruning is a postprocess that follows the algorithm in Fig. 6.

Note that the criterion for selecting a pair that becomes a pseudo node in Cl-GBI and the criterion for selecting a discriminative pattern in DT-ClGBI can be different. In the following experiments, frequency of a pair is used as the former criterion, and information gain of a pattern is used as the latter criterion[1].

3.3 Working Example of DT-ClGBI

Suppose DT-ClGBI receives a set of 4 graphs in the upper left-hand side of Fig. 7. Both the beam width b and the number of levels N_e of Cl-GBI are set to 1 at every node of a decision tree to simplify the working of DT-ClGBI in this example, and the frequency threshold θ is set to 0%. Cl-GBI called inside of DT-ClGBI enumerates all the pairs in these graphs and extracts 11 kinds of pairs from the data. These pairs are: a→a, a→b, a→c, a→d, b→a, b→b, b→c,

[1] We did not use information gain ratio because DT-ClGBI constructs a binary tree.

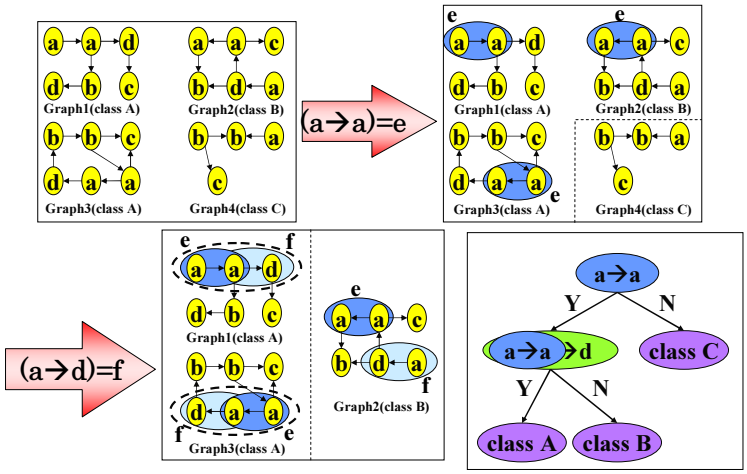

Fig. 7. Example of decision tree construction by DT-ClGBI

Graph	a→a	a→b	a→c	a→d	b→a	b→b	b→c	b→d	d→a	d→b	d→c
1 (class A)	1	1	0	1	0	0	0	1	0	0	1
2 (class B)	1	1	1	1	0	0	0	0	1	1	0
3 (class A)	1	0	1	1	1	1	1	0	0	1	0
4 (class C)	0	1	0	0	0	1	1	0	0	0	0

Fig. 8. Attribute-value pairs at the first step

b→d, d→a, d→b, d→c. The existence/non-existence of the pairs in each graph is converted into the ordinary table representation of attribute-value pairs, as shown in Fig. 8. For instance, graph 1, graph 2 and graph 3 have the pair a→a but graph 4 does not have it. This is shown in the first column in Fig. 8.

Then Cl-GBI selects the most frequent pair "a→a", assigns new label "e" to it to generate a pseudo node as shown in the upper right-hand side of Fig. 7, and terminates. It is worth noting that Cl-GBI can calculate frequency of a pattern based on either the number of graphs that include the pattern (document frequency) or the total number of occurrences of the pattern in all the graphs (total frequency). In this example, document frequency is employed. Next, DT-ClGBI selects the discriminative pattern, i.e., the pattern (pair) with the highest evaluation for classification (i.e., information gain) from the enumerated pairs, and uses it to divide the data into two groups at the root node. In this example, the pair "a→a" is selected. As a result, the input data is divided into two groups: one consisting of graph 1, graph 2, and graph 3, and the other consisting of only graph 4.

The above process is recursively applied at each node to grow up the decision tree while constructing attributes (patterns) useful for the classification task at the same time. In this example, since the former group consists of graphs belonging to different classes, again Cl-GBI is applied to it, while the latter group is no longer divided because it contains a single graph of class C. For the former group, pairs in graph 1, graph 2 and graph 3 are enumerated and the attribute-value table is updated as shown in Fig. 9. Note that the pairs included in the table for the parent node are inherited. In this case, the pair "a→d" is selected by Cl-GBI as the most frequent pair to be a pseudo node "f", while the pair "e→d" is selected as the most discriminative pattern by DT-ClGBI. Consequently, the graphs are separated into two

Graph	a→b	a→c	a→d	b→a	b→b	b→c	b→d	d→a	d→b	d→c
1 (class A)	1	0	1	0	0	0	1	0	0	1
2 (class B)	1	1	1	0	0	0	0	1	1	0
3 (class A)	0	1	1	1	1	1	0	0	1	0

e→b	e→c	e→d	b→e	d→e
1	0	1	0	0
1	1	0	0	1
0	1	1	1	0

Fig. 9. Attribute-value pairs at the second step

partitions, each of which contains graphs of a single class as shown in the lower left-hand side of Fig. 7. The constructed decision tree is shown in the lower right-hand side of Fig. 7.

3.4 Classification using the Constructed Decision Tree

Unseen new graph data must be classified once the decision tree has been constructed. Here again, the problem of subgraph isomorphism arises to test if the input graph contains the pattern (subgraph) specified in the test node of the tree. To alleviate this problem, we utilize Cl-GBI again. Theoretically, if the test pattern actually exists in the input graph, Cl-GBI can find it by setting the beam width b and the number of levels N_e large enough and by setting the frequency threshold to 0. However, note that nodes and links that never appear in the test pattern are never used to form the test pattern in Cl-GBI. Therefore, we can remove such nodes and links from the input graph before applying Cl-GBI to reduce its running time. This approach is summarized as follows:

Step 1 Remove nodes and links that never appear in the test pattern from the input graph.

Step 2 Apply Cl-GBI to the resulting input graph setting the parameters b and N_e large enough, while setting the parameter θ to 0.

Step 3 Test if one of the canonical labels of extracted patterns with the same size as the test pattern is equal to the canonical label of the test pattern.

In general, Step 1 results in a small graph and Cl-GBI can run very quickly without any constraints on N and b. However, if we need to set these constraints, we may not be able to obtain the correct answer because we don't know how large these parameters should be. In that sense, this procedure can be regarded as an approximate solution to the subgraph isomorphism problem.

4 Experimental Evaluation of DT-ClGBI

To evaluate the performance of DT-ClGBI, we conducted some experiments on both synthetic and real-world datasets consisting of directed graphs.

4.1 Synthetic Datasets

Data Preparation Synthetic datasets were artificially generated in a random manner. The number of nodes in a graph, is determined by the gaussian distribution having the average of

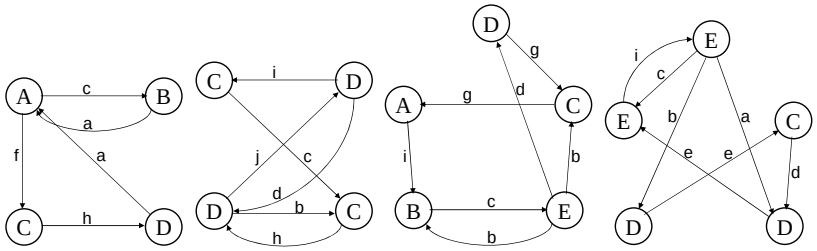

Fig. 10. Example of 4 basic subgraphs

T and the standard deviation of 1. The links are attached randomly with the probability of p. The node labels and link labels are randomly determined with equal probability. The number of node labels and the number of link labels are denoted as L_V and L_E, respectively. The total number of transactions is kept fixed as GD.

Two datasets of directed graphs having the average size of 30 and 40, each of which has $GD = 300$, $L_V = 5$, $L_E = 10$, $p = 20\%$, were generated and are represented as $T30$ and $T40$, respectively. Each dataset was equally divided into two classes, namely "active" and "inactive". Similarly, L basic patterns of connected subgraphs having the average size of I, where $L = 4$ and $I = 4$, were generated. The number of basic patterns to be embedded in a transaction G_t of the class "active", N_t, was randomly selected in the range between 1 and L. Each of these N_t basic patterns was in turn chosen from the set of L basic patterns by equal probability, i.e. $1/L$, and overlaid on that transaction. This means that each transaction of the class "active" includes from 1 to L basic subgraphs, some of them may happen to be the same. We also check if there is any basic subgraph included in a transaction of the class "inactive" by Cl-GBI as described in Section 3.4. If there is, the involved node and link labels are changed in a way that the basic pattern no longer exists in the transaction. In other words, basic subgraphs are those which discriminate the two classes. This does not necessarily mean that any subgraph of the basic patterns is not discrimitive enough. We have not checked that all the subgraphs of the basic patterns appear in both active and inactive data. Fig. 10 shows the 4 basic subgraphs which were embedded in the transactions of the class "active".

Experiments For these two synthetic datasets, the classification task is to classify two classes "active" and "inactive" using DT-ClGBI by a single run of 10-fold cross validation (CV). The final prediction error rate was evaluated by the average of 10 estimates of the prediction error (a total of 10 decision trees).

The first experiment was conducted to confirm that the most discriminative patterns can be extracted by Cl-GBI not only at the root node, but also at each internal node itself. To this end, we compared the predictive accuracy and the tree size obtained by two different settings for DT-ClGBI described as follows. In the first setting, i.e. setting 1, a decision tree is constructed by applying Cl-GBI at the root node only, with $N_e = 2$. At the other nodes, we simply recalculate information gain for those patterns that have already been discovered at the root node. In the other case, i.e., setting 2, Cl-GBI is invoked at the root node with $N_e = 2$ and other nodes with $N_e = 1$. In addition, the total number of levels of Cl-GBI in the second setting is limited to 6 to keep the computation time at a tolerant level. Whenever the total number of levels reaches this limitation, Cl-GBI is no longer used for extracting patterns.

Table 1. Comparisons of different settings for DT-ClGBI

Dataset	Setting 1			Setting 2		
	Training error	Test error	Average of tree sizes	Training error	Test error	Average of tree sizes
$T30$	0.22%	1.33%	17.2	0%	0%	9.2
$T40$	0.37%	5%	18	0.15%	3.33%	12.8

Instead, only the existing patterns are employed for constructing the decision tree thereafter. Note that beam width is set to 5 in both settings.

Results of the first experiment are summarized in Table 1, and it is shown that the second setting obtains higher predictive accuracy. Moreover, we observe that the decision trees constructed by the second setting have smaller sizes in most cycles of the 10-fold CV for both datasets. The result reveals that invoking of Cl-GBI at internal nodes is needed to improve the predictive accuracy of DT-ClGBI, as well as to reduce the tree size. Intuitively, the search space increases by applying Cl-GBI at the internal nodes in addition to the root node. As a result, more discriminative patterns which have not been discovered in the previous steps are discovered at these nodes. In other words, applying Cl-GBI at only the root node cannot help enumerate all the necessary patterns unless N_e and b are set large enough. For example, in the decision tree constructed by the first run of 10-fold CV on the dataset $T30$ using the second setting, the classifying pattern for a node at the third level was not found at the root node but its parent node. If N_e is set large enough, the necessary pattern should be able to be found at the root node. This pattern, if found at the root node, should give smaller information gain at the root node but Cl-GBI retains this and passes down to the lower node. The question is how to find this pattern where it is needed without running Cl-GBI using all the dataset.

The second experiment focused on the comparisons between DT-ClGBI and DT-GBI [4, 5], also in terms of the predictive accuracy and the tree size. Here beam width is also set to 5 in both cases. For DT-GBI, the number of levels of B-GBI at any node of a decision tree is kept fixed as 4. It should be noted that, whenever being invoked for constructing a decision tree by DT-GBI, B-GBI starts extracting typical patterns from the beginning, i.e. no inheritance is employed, because the graphs that pass down to the yes branch have been chunked by the test pattern. On the other hand, the number of levels of Cl-GBI is 4 at the root node and 1 at the other nodes of a decision tree in the case of DT-ClGBI. In addition, the total number of levels of Cl-GBI is limited to 8, which means that the number of levels performed by the feature construction tool in DT-ClGBI is much less than that in DT-GBI. As mentioned earlier, this limitation helps reduce the computation time required for constructing the decision trees by DT-ClGBI, however, at the expenses of the decrease of the search space in Cl-GBI.

Table 2 shows the results of the second experiment. It can be seen that DT-ClGBI achieves lower prediction error for both datasets. We also observe that, for each dataset, the decision trees constructed by DT-ClGBI have smaller sizes in most cycles of the 10-fold CV. The higher predictive accuracy of DT-ClGBI and the simpler decision trees obtained by this method can be explained by the improvement of Cl-GBI over B-GBI, and the inheritance of previously extracted patterns at an internal node (in a decision tree) from its predecessors. It is known that Cl-GBI resolves the problem of overlapping patterns incurred by B-GBI, thus resulting in more typical patterns extracted by Cl-GBI.

Additionally, it should be noted that the size of the embedded graphs in these two datasets is 4 or 5. Setting $N_e = 2$ as in the first experiment means that the maximum size of patterns we can get at the root node is 4. Considering the beam width, it is unlikely that the embedded patterns

Table 2. Comparisons of DT-ClGBI and DT-GBI

Dataset	DT-GBI			DT-ClGBI		
	Training error	Test error	Average of tree sizes	Training error	Test error	Average of tree sizes
$T30$	1.41%	7.67%	24	0%	0%	9
$T40$	3.15%	7.67%	18.2	0%	0.67%	9

are found at the root node. Even $N_e = 4$ as in the second experiment, these basic patterns cannot be found. However, the substructures of the embedded graphs are discriminative enough as shown in the two aforementioned experiments. Due to the downward closure property, these substructures were embedded in the transactions of the class "active".

4.2 Real-world Datasets

Finally, we verified if DT-ClGBI can construct decision trees that achieve reasonably good predictive accuracy also on a real world dataset. For that purpose, we used the hepatitis dataset as in [5]. The classification task here is to classify patients into two classes, "LC" (Liver Cirrhosis) and "nonLC" (non Liver Cirrhosis) based on their fibrosis stages, which are categorized into five stages in the dataset: F0 (normal), F1, F2, F3, and F4 (severe = Liver Cirrhosis). All 43 patients at F4 stage were used as the class "LC", while all 4 patients at F0 stage and 61 patients at F1 stage were used as the class "nonLC". This ratio of "LC" to "nonLC" was determined based on [15]. The records for each patient were converted into a directed graph as described in [5]. The resulting graph database has 108 graph transactions. The average size of a graph transaction in this database is 316.2 in terms of the number of nodes, and 386.4 in terms of the number of links.

Through some preliminary experiments on this database using DT-ClGBI, we found that existence of some graphs often makes the resulting decision tree too complicated and worsen the predictive accuracy. This has led us to adopt a two step approach, first to divide the patients into "typical" and "non-typical", and second to construct a decision tree for each group of the patients. To divide the patients in the first step, we ran 10-fold cross validation of DT-ClGBI on this database, varying its parameters b and N_e in the ranges of $\{5, 6, 8, 10\}$ and $\{6, 8, 10, 12\}$, respectively. Note that these values are only for the root node and we did not run Cl-GBI at the succeeding nodes. The frequency threshold θ was fixed to 10%. Namely, we conducted 10-fold cross validation 16 times with different combinations of these parameters, and obtained totally 160 decision trees in this step. Then we classified graphs whose average error rate is 0% into "typical", and the others into "non-typical". As a result, for the class "LC", 28 graphs are classified into the subset "typical" and the other 15 graphs into "non-typical", while for the class "nonLC", 48 graphs are classified into "typical" and 17 graphs into "non-typical".

In the second step, we applied DT-ClGBI to each subset again adopting the best parameter setting in the first step with respect to the predictive accuracy, where $b = 8$ and $N_e = 10$. The predictive accuracy (average of 10-CV) for the subset "typical" is 97.4%, and that for "non-typical" is 78.1%. The overall accuracy is 91.7%, which is much better than the accuracy obtained by applying DT-ClGBI to the original dataset with $b = 8$ and $N_e = 10$, i.e., 83.4%. We can find typical features for a patient with Liver Cirrhosis in the extracted patterns such as "GOT is High" or "PLT is Low". From these results, we can say that DT-ClGBI can achieve reasonably good predictive accuracy on a real world dataset and extract discriminative features embedded in the dataset as subpatterns.

5 Conclusions

In this paper, we have proposed an algorithm called DT-ClGBI, which can construct decision trees for graph-structured data using Cl-GBI. In DT-ClGBI, substructures, or patterns useful for a classification task are constructed on the fly by means of Cl-GBI during the construction process of a decision tree. The experimental results using synthetic datasets showed that decision trees constructed by DT-ClGBI achieve good predictive accuracy for graph-structured data. The good predictive accuracy of DT-ClGBI is mainly attributed to the fact that the Cl-GBI method can give the correct number of occurrences of a pattern in each transaction of the graph database due to its capability of extracting overlapping patterns, which is very useful for algorithms such as DT-ClGBI that need correct counting. Also, the inheritance of previously extracted patterns at an internal node from its predecessors is shown helpful.

For future work, we plan to employ some heuristics to speed up the Cl-GBI algorithm to extract larger typical subgraphs at an early stage in the search process. This could also improve the performance of DT-ClGBI.

References

1. Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. 1989. The CN2 Induction Algorithm, *Machine Learning*, **3**: 261–283.
2. Cook, D. J. and Holder, L. B. 1994. Substructure Discovery Using Minimum Description Length and Background Knowledge, *Artificial Intelligence Ressearch*, **1**: 231–255.
3. Fortin, S. 1996. The Graph Isomorphism Problem, *Technical Report TR96-20*, Dept. of Computer Science, University of Alberta, Edmonton, Canada.
4. Geamsakul, W., Matsuda, T., Yoshida, T., Motoda, M., and Washio, T. 2003. Classifier Construction by Graph-Based Induction for Graph-Structured Data, In *Proc. PAKDD 2003*, pp. 52–62.
5. Geamsakul, W., Yoshida, T., Ohara, K., Motoda, H., Washio, T., Takabayashi, K., Yokoi, H. 2005. Constructing a Decision Tree for Graph-Structured Data and Its Applications. *Fundamenta Informaticae*, **66**(1-2): 131–160.
6. Inokuchi, A., Washio, T., and Motoda, H. 2003. Complete Mining of Frequent Patterns from Graphs: Mining Graph Data, *Machine Learning*, **50**(3): 321–354.
7. Inokuchi, A., Washio, T., Nishimura, K., and Motoda, H. 2002. A Fast Algorithm for Mining Frequent Connected Subgraphs, *IBM Research Report RT0448*, Tokyo Research Laboratory.
8. Kuramochi, M. and Karypis, G. 2004. An Efficient Algorithm for Discovering Frequent Subgraphs, *IEEE Trans. Knowledge and Data Engineering*, **16**(9): 1038-1051.
9. Kuramochi, M. and Karypis, G. 2004. GREW–A Scalable Frequent Subgraph Discovery Algorithm, In *Proc. ICDM 2004*, pp. 439–442.
10. Matsuda, T., Motoda, H., Yoshida, T., and Washio, T. 2002. Mining Patterns from Structured Data by Beam-wise Graph-Based Induction, In *Proc. DS 2002*, pp. 422–429.
11. Michalski, R.S. 1990. Learning Flexible Concepts: Fundamental Ideas and a Method Based on Two-Tiered Representation, In *Machine Learning: An Artificial Intelligence Approach*, **3**: 63–102.
12. Nguyen, P.C., Ohara, K., Motoda, H., and Washio, T. 2005. Cl-GBI: A Novel Approach for Extracting Typical Patterns from Graph-Structured Data, In *Proc. PAKDD 2005*, pp. 639–649.
13. Quinlan, J.R. 1986. Induction of Decision Trees, *Machine Learning*, **1**: 81–106.
14. Quinlan, J.R. 1993. *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers.
15. Yamada, Y., Suzuki, E., Yokoi, H., and Takabayashi, K. 2003. Decision-tree Induction from Time-series Data Based on a Standard-example Split Test, In *Proc. ICML 2003*, pp. 840–847.
16. Yan, X. and Han, J. 2002. gSpan: Graph-Based Structure Pattern Mining, In *Proc. ICDM 2002*, pp. 721–724.
17. Yoshida, K. and Motoda, M. 1995. CLIP: Concept Learning from Inference Patterns, *Artificial Intelligence*, **75**(1): 63–92.