Pattern Discovery from Graph-Structured Data - A Data Mining Perspective -

Hiroshi Motoda

Asian Office of Aerospace Research & Development
Air Force Office of Scientific Research, Tokyo, JAPAN
hiroshi.motoda@aoard.af.mil

Abstract. Mining from graph-structured data has its root in concept formation. Recent advancement of data mining techniques has broadened its applicability. Graph mining faces with subgraph isomorphism which is known to be NP-complete. Two contrasting approaches of our work on extracting frequent subgraphs are revisited, one using complete search (AGM) and the other using heuristic search (GBI). Both use canonical labelling to deal with subgraph isomorphism. AGM represents a graph by its adjacency matrix and employs an Apriori-like bottom up search algorithm using anti-monotonicity of frequency. It can handle both connected and dis-connected graphs, and has been extended to handle a tree data and a sequential data by incorporating a different bias to each in joining operators. It has also been extended to incorporate taxonomy in labels to extract generalized subgraphs. GBI employs a notion of chunking, which recursively chunks two adjoining nodes, thus generating fairly large subgraphs at an early stage of search. The recent improved version extends it to employ pseudo-chunking which is called chunkingless chunking, enabling to extract overlapping subgraphs. It can impose two kinds of constraints to accelerate search, one to include one or more of the designated subgraphs and the other to exclude all of the designated subgraphs. It has been extended to extract paths and trees from a graph data by placing a restriction on pseudo-chunking operations. GBI can further be used as a feature constructor in decision tree building. The paper explains how both GBI and AGM with their extended versions can be applied to solve various data mining problems which are difficult to solve by other methods.

1 Introduction

Recent advancement of data mining techniques has made it possible to mine from complex structured data. Since structure is represented by proper relations and a graph can easily represent relations, knowledge discovery from graph-structured data (graph mining) poses a general problem for mining from structured data. Some examples amenable to graph mining are finding functional components from their behavior, finding typical web browsing patterns, identifying typical substructures of chemical compounds, finding subsequences of DNA typical to some functions and discovering diagnostic rules from patient history records.

The first example above is also called concept formation and has been a subarea of artificial intelligence research since many years ago. One such work

is by Yoshida and Motoda[11], where a concept is defined as something that minimizes inference load and the problem is finding a mapping that satisfies this requirement. This can be amenable to graph mining problem. The results of qualitative simulation of a digital circuit was mapped to a set of directed graph from which hierarchical concepts such as "pull down transistor", "exclusive OR" and "carry chain" were extracted. The idea behind the concept formation from a graph-structure data and its application to a digital circuit is schematically shown in Figs. 1 and 2. Similar work is reported in [1].

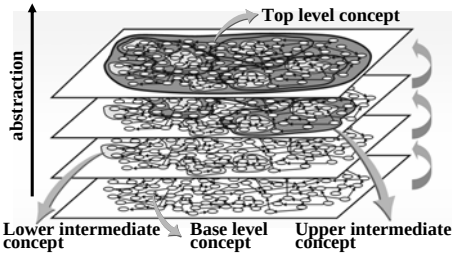

Fig. 1. Hierarchical functional abstraction of a complex object

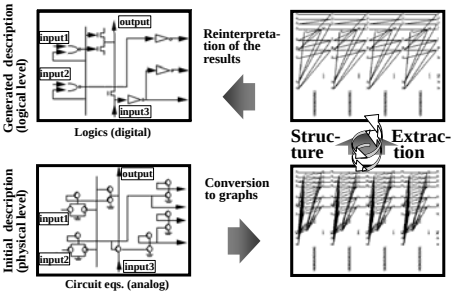

Fig. 2. Concept formation for a digital circuit

Graph mining is based on finding some typicality from a vast amount of graph-structured data. What makes it typical depends on each domain and each task. Most often frequency which has a good property of anti-monotonicity is used to discover typical patterns. Some measure such as information gain or χ^2 are also used but since they are not monotonic with respect to graph subsumption, special care must be taken. Graph mining inevitably faces with subgraph isomorphism which is known to be NP-complete. For example, three subgraphs in Fig. 3 are all isomorphic and it is not easy to find all of them in a huge set of graphs.

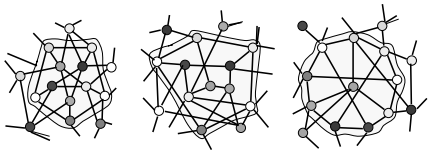

Fig. 3. Three isomorphic graphs

Two contrasting approaches have been taken to handle this problem, one searching all possible space efficiently devising a good data structure with an appropriate indexing and the other avoiding exhaustive search using a greedy algorithm with good heuristics. In this paper, our own work on these two approaches is revisited, AGM family for the former and GBI family for the latter, and how these are applied to solve difficult problems which are not easily solved my other approaches is explained. Both approaches use canonical labeling to handle subgraph isomorphism.

2 AGM family - algorithm and its applications -

AGM [3, 4] represents a graph by its adjacency matrix. Labels of nodes and edges are assigned natural numbers. The adjacency matrix is defined as follows. First

node labels are ordered according to their values and grouped according to this order, and a square matrix is formed. An element of the matrix is the edge label if there is an edge between the corresponding node pair and 0 if there is no edge between them. Since the same graph can be represented by multiple adjacency matrices because any permutation of the raw and the column within the same node label represents the same graph, each matrix is given a code and a code which gives a minimum (maximum) is defined as the canonical label.

A code consists of two parts, the first part is the code representing the node label ordering and the second part is the code consisting of the vertically scanned matrix elements. Thus, if the canonical label of two graphs are the same, they are identified as isomorphic. An example is given in Fig. 4. The graph has 5 nodes with 3 node labels and no edge labels. The node labels are ordered as grey (=1), dark-grey (=2) and black (=3). The first part is 11233 and the second part is 1011100011. The four adjacency matrices below represent the same graph. These are different only in the node numbering within the same label. The canonical label is 112331010011101 if the minimum is chosen.

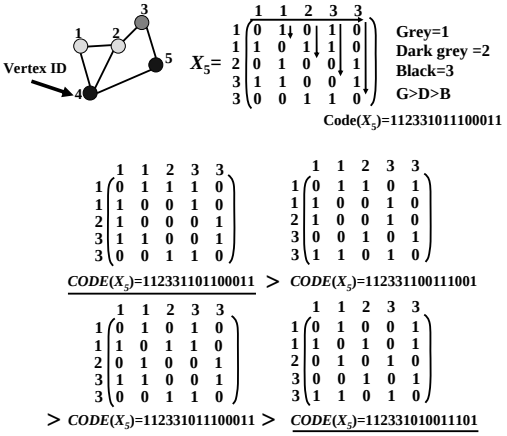

Fig. 4. Adjacency matrix and canonical label

Once we have defined a canonical label, candidate subgraph can be generated and searched using an Apriori-like bottom up search algorithm. In Apriori an itemset P_k of size k can be generated by 2 frequent itemsets of size $k-1$ that share the same $k-2$ items. This is called a join operation. Then it verifies that all the subsets of size $k-1$ in P_k are frequent or not, and if not it discards P_k. Finally the support of P_k is checked to see if it satisfies the min. support condition. The similar procedure can be applied to graphs. Two graphs of size $k-1$ can be joined to generate a graph of size k. Figure 5 shows an example when $k = 5$. Here, the first matrix is always a canonical form (matrix with canonical label). What is different is that the generated matrix has missing elements. All the possible alternatives must be considered. If we have β labels for edges, there are $\beta+1$ adjacency matrices for undirected graph and its square for directed graph. Thus, the number of subgraphs rapidly increases with size. AGM can handle general graphs, both connected and disconnected, both directed and undirected, and both labeled and unlabeled.

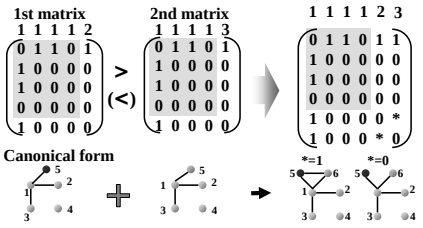

Fig. 5. Join operation of AGM

Since AGM is most generic, restricting the types of subgraphs is straightforward. It is easy to place a bias in join operation. B-AGM, biased AGM, can handle connected subgraphs, tree, ordered tree, path and sequence [5]. The conceptual scheme of the AGM family is depicted in Fig. 6.

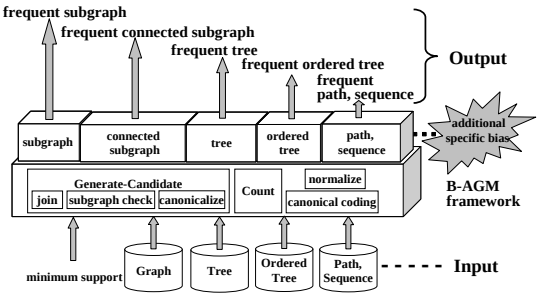

Fig. 6. Mining Various substructures by B-AGM Framework

Performance of AGM was evaluated using PTE dataset and is shown in Figure 7. This is a small dataset (the average graph size is 27). The connected version of B-AGM, AcGM favorably compares with the other later developed graph mining algorithms[10, 6]. All do the complete search. If we limit the search to induced subgraphs, B-AGM is most efficient.

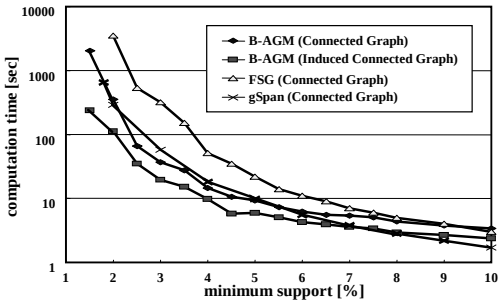

Fig. 7. Performance evaluation of AcGM (PTE data, Average size of graphs=27)

AGM family has been applied to many problems, mostly to chemical compound analysis such as mutagenicity, HIV and dopamine, and a few others such as Web browsing history analysis and consumer behavior analysis. In the analysis of mutagenicity of amino acid compounds, AGM was able to find that compounds that have hydrogen next to the nitro substituent in a benzene ring can be active. The results was obtained by using only the topological information of graphs. Later analysis of three dimensional structure of these compounds revealed that in case of hydrogen there is no steric hindrance that destroys the coplanarity to a benzene ring but a more complicated one does destroy the coplanarity. This explains the mining results. When applied to HIV data, AGM was able to find a subgraph which is very close to what is called azido-thymidine (AZT), a well known anti-HIV medicine. The found subgraph is shown in Fig. 8. This is a three class problem and the task is to find all the frequent subgraphs that are greater than the minimum support in the active compounds and less than the maximum support in the inactive compounds.

AGM was extended to handle taxonomy in node labels. Use of taxonomy makes it possible to find more abstract concepts even if there are not enough number of frequent subgraphs at the base level description. The problem is conceptually easy but overgeneralization must be prevented. It is possible to extract the least general subgraph by discarding more general subgraphs that

have the same total occurrence where multiple count is allowed if there are more than one occurrence of a subgraph in a graph. This technique was applied to PTE dataset and AGM found more discriminative subgraphs together with the taxonomy itself. AGM was further extended to handle three dimensional structure without modifying the algorithm (3D-AGM). Edge length was discretized into a finite number and was given different labels. With this approach, when applied to classification of dopamine antagonists, 3D-AGM was able to find more discriminative substructures than the standard AGM. An interesting application of AGM is consumer behavior analysis. Each time a person goes to a grocery store, items purchased together are recorded and this can be used for later analysis. The purchase history can be mapped to a directed graph. When applied to a data for a beer market, AGM discovered that a particular brand "Asahi super dry" is sold together with fresh fish and fruit. This hypothesis was tested in collaboration with the grocery store by rearranging the items so that these three are placed close. Interestingly the sales of beer went up more than double and fish sales went up also by 12% during the promotion [9].

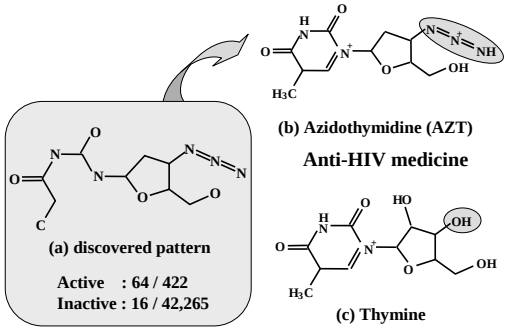

Fig. 8. Discovered subgraph by AcGM for HIV data

3 GBI family - algorithm and its applications -

GBI [7] employs a notion of chunking, which recursively chunks two adjoining nodes, thus generating fairly large subgraphs at an early stage of search. GBI also uses canonical labeling to solve the graph isomorphism problem. Due to the nature of chunking, GBI is able to handle only connected graphs. Instead, GBI can use any criterion that is based on the frequency of paired nodes. However, for finding a subgraph that is of interest any of its subgraphs must be of interest because of the nature of repeated chunking. The frequency measure satisfies this monotonicity property. However, if the criterion chosen does not satisfy this property, repeated chunking may not lead to finding good subgraphs even though the best pair based on the criterion is selected at each iteration. To resolve this issue GBI uses two criteria, one based on frequency measures for chunking and the other for finding discriminative subgraphs after chunking. The latter criterion does not necessarily exhibit the monotonicity property. Any function that is discriminative can be used, such as Information Gain, Gain Ratio, Gini Index and others.

The original GBI contracts graphs after chunking and thus the size of the graphs progressively becomes smaller as chunking proceeds and thus the computational complexity is almost quadratic to the graph size. The basic algorithm

is given in Fig. 9. However, the biggest problem with this approach is that it cannot find overlapping subgraphs. Later version introduced a beam search to alleviate this problem (B-GBI), but it was not enough. The recent improved version Cl-GBI extends it to employ pseudo-chunking which is called chunkingless chunking, enabling to extract overlapping subgraphs [8].

GBI(G)
  Enumerate all the pairs P_{all} in G
  Select a subset P of pairs from P_{all} (all the pairs in G) based on typicality criterion
  Select a pair from P_{all} based on chunking criterion
  Chunk the selected pair into one node c
  $G_c :=$ contracted graph of G
  while termination condition not reached
    $P := P \cup$ GBI(G_c)
  return P

Fig. 9. Algorithm of GBI

In Cl-GBI, the selected pairs are registered as new nodes and assigned new labels but are never chunked and the graphs are never "contracted" nor copied into respective states as in B-GBI. In the presence of the pseudo nodes (i.e., newly assigned-label nodes), the frequencies of pairs consisting of at least one new pseudo node are counted. The other is either one of the pseudo nodes including those already created in the previous steps or an original node. The most frequent pairs with the number equal to the beam width specified in advance are selected among the remaining pairs and the new pairs which have just been counted for their frequencies. These steps are repeated for a predetermined number of times, each of which is referred to as a level. Those pairs that satisfy a typicality criterion (e.g., pairs whose information gain exceeds a given threshold) among all the extracted pairs are the output of the algorithm. A frequency threshold is used to reduce the number of pairs being considered to be typical patterns. Another possible method to reduce the number of pairs is to eliminate those pairs whose typicality measure is low even if their frequency count is above the frequency threshold. The two parameters, beam width and number of levels, control the search space. The frequency threshold is another important parameter. The algorithm of Cl-GBI is depicted in Fig. 10. In contrast to GBI and B-GBI graph size remains the same due to pseudo-chunking, and thus, the number of pairs to pseudo-chunk progressively increases (computational complexity is now exponential to the size). It searches a much larger portion of subgraphs and in fact, search can be complete by setting the beam width and the level large enough. Like B-AGM, the subgraphs to be searched can be limited to paths, acyclic

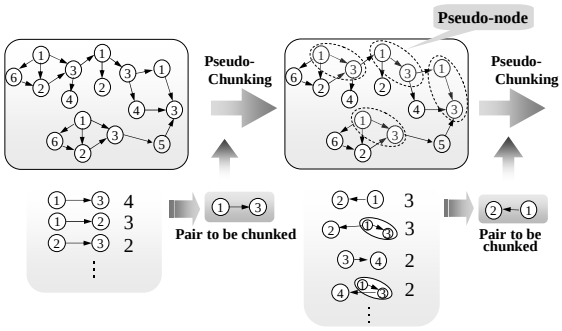

Fig. 10. Pseudo-chunking of Cl-GBI

subgraphs and subgraphs (both induced and general) by placing appropriate constraints when chunking the pairs.

Performance of Cl-GBI was evaluated using the same PTE dataset as in AGM. The results are shown in Table 1. The number of beam width and the number of levels to find all the subgraphs are shown. The subgraphs found are confirmed to be the same as what were found by AcGM.

Table 1. Performance evaluation of Cl-GBI (PTE Data)

General subgraphs

Frequency threshold (θ)	30%	20%	10%
No. of freq. patterns	68	190	844
Beam width (b)	10	10	10
No. of levels (N) needed	12	18	84

Induced subgraphs

Frequency threshold (θ)	30%	20%	10%
No. of freq. patterns	49	139	537
Beam width (b)	10	10	10
Number of levels (N) needed	4	7	18

GBI family does not aim to find all the possible subgraphs but rather it attempts to find reasonably good subgraphs at an early stage in the search without searching all the space. To accomplish this, various techniques are introduced to improve its search efficiency. Two of them are mentioned here. The first one is to impose constraints to restrict search. Two kinds of constraints can be conceived. One is to extract only patterns that are of interest to domain experts or related to domain knowledge, and the other is to exclude extracting patterns uninteresting to domain experts or unrelated to domain knowledge. Both are called INpattern constraint and EXpattern constraint. Their implementation is straightforward. However, subgraph isomorphism checking is needed to check these constraints, and thus the number of checking must be reduced as much as possible. For this, necessary conditions for two subgraphs to be isomorphic, e.g. degree of node, node and edge labels, etc. are used. If these conditions are not satisfied, there is no need to compare the two subgraphs. The second one is to use pruning. Frequency satisfies anti-monotonicity but information gain, and χ^2 do not satisfy this constraint. However, they have a nice property of being convex with respect to the arguments. In this case it is possible to set an upperbound to them. Suppose we have a subgraph P and want to chunk it with other subgraph, generating a larger subgraph Q. The information gain of Q is bounded by the maximum of these extreme cases where the input instances containing Q only consists of a single class. If the upperbound for Q is less than or equal to the so far found best information gain or χ^2, then there is no need to pseudo chunk P.

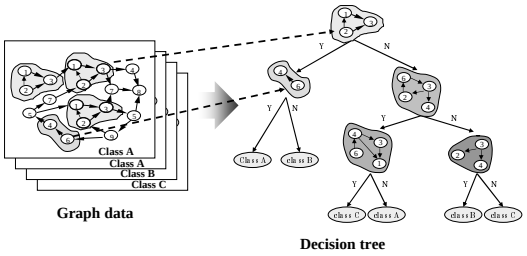

Fig. 11. Using Cl-GBI as a feature constructor in decision tree building

If the final task is classification, graph mining has to be combined with classifier construction. For this task, GBI family can be used as a feature constructor in decision tree building [2] (See Fig. 11). Decision tree constructed this way was named DT-GBI or DT-ClGBI. Each node has a subgraph for which an input data is tested. Cl-GBI is run recursively reusing whatever can be inherited from the previous runs. The beam width and depth level are parameters that can be set at each tree level. The algorithm of DT-ClGBI is given in Fig. 12

```
DT-GBI(D)
  Create a node DT for D
  if termination condition reached
    return DT
  else
    P := GBI(D) (with the number of
         chunking specified)
    Select a pair p from P
    Divide D into D_y (with p) and D_n (with-
     out p)
    Chunk the pair p into one node c
    D_yc := contracted data of D_y
    for D_i := D_yc, D_n
      DT_i := DT-GBI(D_i)
      Augment DT by attaching DT_i as its
       child along yes(no) branch
  return DT
```

Fig. 12. Algorithm of DT-ClGBI

The performance of DT-ClGBI was evaluated by a synthetic dataset. Directed graphs are randomly generated and equally divided into two classes, active and inactive. Four kinds of basic patterns were embedded in class active with equal probability. The average size of the graphs is changed from 30 to 50.

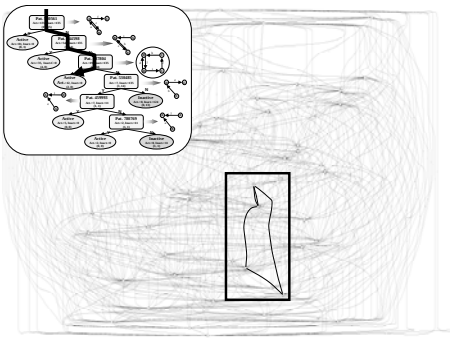

Fig. 13. An example of input graph classified as positive by DT-ClGBI

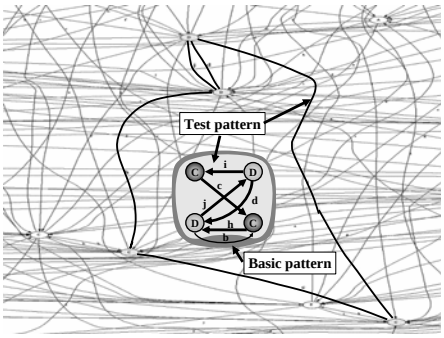

Fig. 14. Test pattern found in the input graph in Fig.13

The expectation was that these basic patterns appear in the test nodes of the decision tree, but the result was not exactly what was expected. The patterns chosen at each node were subgraphs of the basic patterns. Figure 13 shows one of the decision trees obtained and an input graph that failed in the first and the second tests but passed the third test and classified as positive. A graph of size 50 is not a large graph, but for humans it is complicated enough and not easy to check if it includes the subgraph used for the test.

Both DT-GBI and DT-ClGBI were applied to a medical dataset of chronic hepatitis B and C which has been collected at the medical department of Chiba University over 20 years from 1982 to 2001. It is a large uncleansed time series data and has inconsistent measurements and many missing values. It was hoped that graph mining can extract typical correlation among different tests across time. Figure 15 shows how the time series data of each patient is converted to a directed graph after steps of preprocessing.

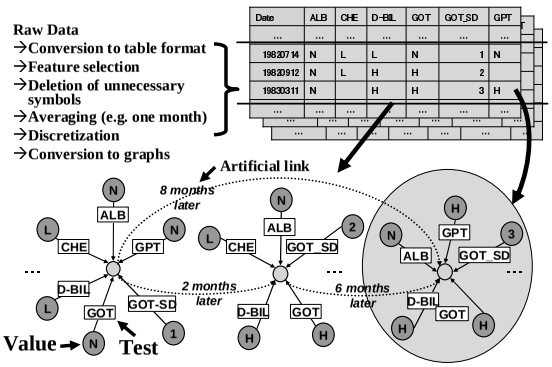

Fig. 15. Data preprocessing and conversion to a directed graph

The star shaped graph represents the various test data that are averaged over two months at a particular time point. These are connected sequentially and artificial links are added up to two years in future to represent possible direct effects of the past to the future. The main task is classifying patients with different fibrosis stages. The data used was 500 days before and after the first biopsy. Feature selection resulted in 32 tests (measurements) as the useful attributes. and the average size of the graphs was about 300. For each experiment 10 fold cross validation was run 10 times. Beam width was set to 15 and the number of levels to 20 at every node of the tree. The average error of distinguishing between F4 stage (liver cirrhosis) and F0+F1 stages (almost healthy) is 12.5% and its standard deviation was 2.12%. The average error of distinguishing between F4 stage and F3+F2 stages (not cirrhosis yet but close to it), which is a more difficult task, was 23.5% with the standard deviation of 2.39%. One of the decision trees is shown in Fig. 16. The patterns are fairly complex and these were exactly what we were expecting.

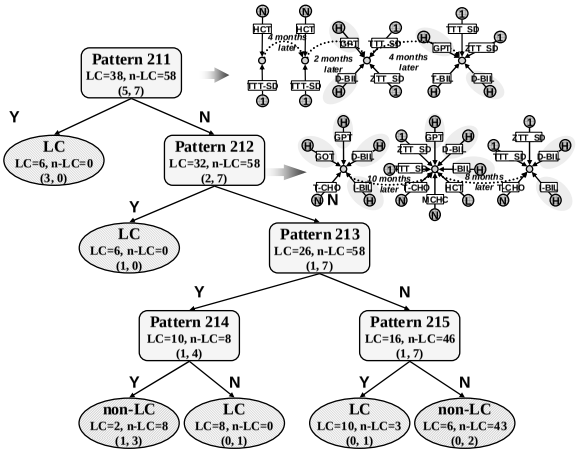

Fig. 16. Optimal decision tree to classify fibrosis progress

However, unfortunately the medical doctors could not interpret these patterns. They thought the techniques interesting but did not accept the results blindly.

Detailed analysis revealed that there are groups of patients whose data behave strangely. This lead to a two stage analysis in which first these abnormal patients were separated from the rest and then a classifier was constructed for each group. The weighted average error of normal and abnormal patients reduced considerably to 7.3% and the tree became very simple and easily interpretable.

4 Conclusion

The paper discussed the use of graph-structured data in data mining perspective. Graph mining is an important area for extracting useful knowledge from structured data. Many interesting and difficult problems can be solved with this approach. However, graph mining is a computationally heavy task. Good algorithm and good data structure are needed. However, what is more important is a right problem setting to produce human understandable solutions. Human's cognitive capability is limited.

References

1. D. J. Cook and L. B. Holder. Graph-based data mining. *IEEE Intelligent Systems*, 15(2):32–41, 2000.
2. W. Geamsakul, T. Yoshida, K. Ohara, H. Motoda, H. Yokoi, and K Takabayashi. Constructing a decision tree for graph-structured data and its applications. *Journal of Fundamenta Informatiae, Special issue on Advances in Mining Graphs, Trees and Sequence*, 66(1–2):131–160, 2005.
3. A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Proc. of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 13–23, 2000.
4. A. Inokuchi, T. Washio, and H. Motoda. Complete mining of frequent patterns from graphs: Mining graph data. *Machine Learning*, 50(3):321–354, 2003.
5. A Inokuchi, T. Washio, and H. Motoda. General framework for mining frequent subgraphs from labeled graphs. *Journal of Fundamenta Informatiae, Special issue on Advances in Mining Graphs, Trees and Sequence*, 66(1–2):53–82, 2005.
6. M. Kuramochi and G. Karypis. An efficient algorithm for discovering frequent subgraphs. *IEEE Trans. Knowledge and Data Engineering*, 16(9):1038–1051, 2004.
7. T. Matsuda, H. Motoda, and T. Washio. Graph-based induction and its applications. *Advanced Engineering Informatics*, 16(2):135–143, 2002.
8. P. C. Nguyen, K. Ohara, H. Motoda, and T. Washio. Cl-gbi: A novel approach for extracting typical patterns from graph-structured data. In *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2005.
9. K. Yada, H. Motoda, T. Washio, and A. Miyawaki. Consumer behavior analysis by graph mining technique. *New Mathematics and Natural Computation*, 2(1):59–68, 2005.
10. X. Yan and J. Han. gspan: Graph-based structure pattern mining. In *Proc. of the 2nd IEEE International Conference on Data Mining*, pages 721–724, 2002.
11. K. Yoshida and H. Motoda. Clip : Concept learning from inference pattern. *Journal of Artificial Intelligence*, 75(1):63–92, 1995.