Cl-GBI: A Novel Strategy to Extract Typical Patterns from Graph Data

Phu Chien Nguyen, Kouzou Ohara, Hiroshi Motoda
and Takashi Washio†

A machine learning technique called Graph-Based Induction (GBI) extracts typical patterns from graph data by stepwise pair expansion (pair-wise chunking). Because of its greedy search strategy, it is very efficient but suffers from incompleteness of search. Also, it cannot give the correct number of occurrences as well as the positions of patterns in each transaction of the graph data. Improvement is made on its search capability by using a new search strategy, where frequent pairs are never chunked but used as pseud-nodes in the subsequent steps, thus allowing extraction of overlapping subgraphs. This new algorithm, called Cl-GBI (Chunkingless Graph-Based Induction), was tested against two datasets, the promoter dataset from UCI repository and the hepatitis dataset provided by Chiba University, and shown successful in extracting more typical substructures.

1. Introduction

In recent years, discovering frequent patterns of graph data, i.e., frequent subgraph mining or simply graph mining, has attracted much research interest because of its broad application areas such as bioinformatics[6], cheminformatics[7),13)], etc. Moreover, since these patterns can be used as input to other data mining tasks (e.g., clustering and classification[5]), the graph mining algorithms play an important role in further expanding the use of data mining techniques to graph-based datasets.

AGM[7] and a number of other methods (AcGM[8], gSpan[13], FFSM[6], 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[4].

On the other hand, existing heuristic algorithms, which are not guaranteed to find the complete set of subgraphs, such as SUBDUE[3] and GBI (Graph-Based Induction)[14], tend to find an extremely small number of patterns. Both the two methods use greedy search to avoid high complexity of the subgraph isomorphism problem. GBI extracts typical patterns from graph 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 depend on which pairs are selected for chunking. 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.

In this paper we propose a novel algorithm for extracting typical patterns from graph data, which does not employ the pair-wise chunking strategy. Instead, the most b (beam width) 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 pseud-nodes. This algorithm, now called Chunkingless Graph-Based Induction (or Cl-GBI for short), was evaluated on two datasets, the promoter dataset from UCI repository and the hepatitis dataset provided by Chiba University, and shown successful in extracting more typical substructures compared to the B-GBI algorithm.

2. Graph-Based Induction Revisited

2.1 Principle of GBI

GBI employs the idea of extracting typical

† Institute of Scientific and Industrial Research, Osaka University, 8-1 Mihogaoka, Ibaraki, Osaka 567-0047, Japan.

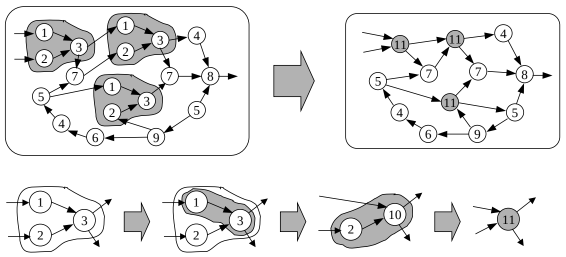

Fig. 1 Principle of GBI.

patterns by stepwise pair expansion as shown in **Fig. 1**. In the original GBI an assumption is made that typical patterns represent some concepts/substructure 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[11)], Gain Ratio[12)] and Gini Index[2)], all of which are based on frequency. In Fig. 1 the shaded pattern consisting of nodes 1, 2, and 3 is thought typical because it occurs three times in the graph. GBI first finds the 1→3 pairs based on its frequency, chunks them into a new node 10, then in the next iteration finds the 2→10 pairs, chunks them into a new node 11. The resulting node represents the shaded pattern.

It is possible to extract typical patterns of various sizes by repeating the above three steps. Note that the search is greedy. No backtracking is made. This means that in enumerating pairs no pattern which has been chunked into one node is restored to the original pattern. Because of this, all the "typical patterns" that exist in the input graph are not necessarily extracted. 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.

As described earlier, GBI can use any criterion that is based on the frequency of paired nodes. However, for finding a pattern that is of interest any of its subpatterns must be of interest because of the nature of repeated chunking. In Fig. 1 the pattern 1→3 must be typical for the pattern 2→10 to be typical. Said differently, unless pattern 1→3 is chunked, there is no way of finding the pattern 2→10. Frequency measure satisfies this monotonicity. However, if the criterion chosen does not satisfy this monotonicity, repeated chunking may not find good patterns even though the best pair based on the criterion is selected at each iteration. To resolve this issue GBI was improved to use two criteria, one for frequency measure for chunking and the other for finding discriminative patterns after chunking. The latter criterion does not necessarily hold monotonicity property. Any function that is discriminative can be used, such as Information Gain[11)], Gain Ratio[12)] and Gini Index[2)], and some others.

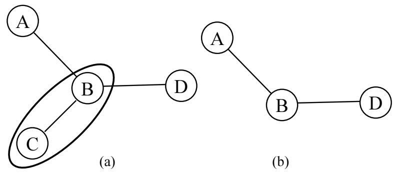

Fig. 2 Missing patterns due to chunking order.

2.2 Beam-wise Graph-Based Induction (B-GBI)

Since the search in GBI is greedy and no backtracking is made, which patterns are extracted by GBI depends on which pair is selected for chunking. There can be many patterns which are not extracted by GBI. In **Fig. 2**, if B–C is selected for chunking beforehand, there is no way to extract the structure A–B–D even if it is a typical pattern.

A beam search is incorporated to GBI, B-GBI[10)], within the framework of greedy search in order to relax this problem, increase the search space, and extract more discriminative patterns while still keeping the computational complexity within a tolerant level. A certain fixed number of pairs ranked from the top are selected to be chunked individually in parallel. To prevent each branch growing exponentially, the total number of pairs to chunk (the beam width) is fixed at every time of chunking. Thus, at any iteration step, there is always a fixed number of chunking performed in parallel.

Figure 3 shows how search is conducted in B-GBI when beam width is set to five. First, five frequent pairs are selected from the graphs at the starting state in search (*cs* in Fig. 3). Graphs in cs are then copied into the five states ($c11 \sim c15$), and each of five pairs is chunked

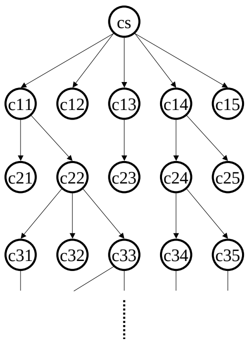

Fig. 3 Beam search in B-GBI (beam width = 5).

in the copied graphs at the respective state. At the second cycle in search, pairs in graphs are enumerated in each state and five frequent pairs are selected from all the sates. In this example, two pairs are selected from $c11$, one pair from $c13$, and two pairs from $c14$.

At the third cycle in search, graphs in $c11$ are copied into $c21$ and $c22$, graphs in $c13$ are copied into $c23$, and graphs in $c24$ are copied into $c24$ and $c25$. As in the second cycle, the selected pairs are chunked in the copied graphs. The states without the selected pairs (in this example $c12$ and $c15$) are discarded.

Another improvement made in conjunction with B-GBI is canonical labeling. GBI assigns a new label for 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 (subgraph). A simple example is shown in **Fig. 4**. They represent the same pattern but the way they are constructed is different. To identify if the two pairs represent the same pattern, each pair is represented by canonical label[4)] and they are regarded identical only when the label is the same.

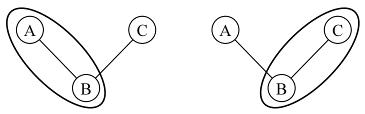

Fig. 4 Two different pairs representing identical patterns.

3. Problem of Overlapping Subgraphs

As described in Section 2.2, the GBI algorithm cannot overcome the problem of overlapping subgraphs. B-GBI can help alleviate this problem, but cannot solve it completely because the chunking process is still involved.

Any subgraph that GBI (and also B-GBI) can find is along 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 graphs. An example is shown in **Fig. 5**, where even if the pair A – B is selected for chunking and the structure D – A – B – C exists in the input graphs, we may not find that structure because an unexpected pair A – B is chunked (see Fig. 5(b)). This causes a serious problem in counting the frequency of a pattern.

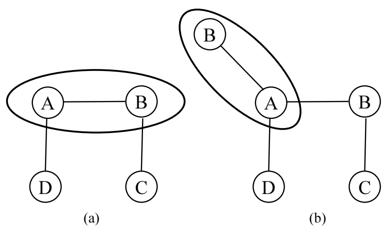

Fig. 5 A pattern is found in one input graph but not in the other.

The complete graph mining algorithms (such as AGM[7)], AcGM[8)], gSpan[13)], etc.) do not face the problem of overlapping subgraphs since they can find all frequent patterns in the graph data. These methods can help answer the query "how many transactions of the graph data contain this pattern?" They, however, are not able to provide us with information about the number of occurrences of a pattern in any transaction of the graph data. They also cannot give us the positions of the pattern in any graph transaction which is required by non-expert users. In addition, these algorithms are not designed for the purpose of enumerating frequent patterns in a single large graph.

On the other hand, SUBDUE[3)], GBI[14)] and GREW[9)] are designed for the purpose of enumerating typical patterns in a single large graph only. Specially, B-GBI[10)] can find (not all) typical patterns in either a large single graph or a set of graphs but it cannot detect the positions of patterns. In Section 4, we will introduce a novel algorithm that can overcome the problem of overlapping subgraphs imposed on GBI and B-GBI. The proposed algorithm, called Cl-GBI (Chunkingless Graph-Based Induction), employs a "chunkingless" strategy, where frequent pairs are never chunked but used as pseud-nodes in the subsequent steps, thus allowing extraction of overlapping subgraphs. It can also give the positions of patterns present in each graph transaction as well as be applied to find frequent patterns in either a single large graph or graph datasets.

4. Chunkingless Graph-Based Induction (Cl-GBI)

4.1 Approach

The basic ideas of Cl-GBI are described as follows. Those pairs that connect two adjoining nodes in the graphs are counted and the most b (beam width) frequent pairs are selected. In B-GBI, each of the selected pairs is registered as one node and this node is assigned a new label. Then, the graphs in the respective state are rewritten by replacing all the occurrences of the selected pair with a node with the newly assigned label (pair-wise chunking).

In Cl-GBI, we also register the b selected pairs as new nodes and assign b new labels to them. But those pairs are never chunked and the graphs are not "compressed". Thus, there is no need to copy the graphs into b states as in B-GBI. In the presence of the pseud-nodes (i.e., newly assigned-label nodes), we count the frequencies of pairs consisting of at least one pseud-node. The other can be either one of pseud-nodes including those already created in the previous levels or an original one. In other words, the other is one of the existing nodes. Among the remaining pairs (after selecting the most b frequent pairs) and the new pairs which have just been counted, we select the most b frequent pairs again and so on.

These steps are repeated N_e 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 below its threshold even if their frequency count is above the threshold. The two parameters b and N_e control the search space. Frequency threshold is another important parameter. With these parameters, there are two options to control the search.

Option 1: Find the subgraphs above the frequency threshold within the search space defined by the parameters b and N_e.

Option 2: Find all the subgraphs above the frequency threshold. In this case, the threshold value is assumed to be large enough so that the search space is manageable with appropriate values of b and N_e.

As in B-GBI, the Cl-GBI approach can handle both directed and undirected graphs. It also can handle both general subgraphs and induced subgraphs.

4.2 Algorithm of Cl-GBI

Input A graph database, two natural numbers b (beam width) and N_e (number of levels), and a frequency threshold θ.

Step 1 Extract all the pairs consisting of connected two nodes in the graphs, register their positions using node id (identifier) lists, and count their frequencies. Since the 2^{nd} level, extract all the pairs consisting of connected two nodes with at least one of which is a newly assigned-label node, the other can be either a newly assigned-label node including those already created in the previous levels or an original one (i.e., the other is one of the existing nodes).

Step 2 Select the most b frequent pairs from among the pairs extracted at Step 1 (since the 2^{nd} level, 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 in the set of original nodes of the graphs (i.e., they are newly assigned-label 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.

These steps are repeated N_e times (N_e levels). All the pairs extracted at Step 1 in all the levels (i.e. level 1 to level N_e) including those that are not newly labeled are ranked based on a typicality criterion (e.g., Information Gain). It is worth noting that those pairs that have fre-

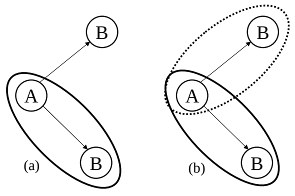
Fig. 6 An example of frequency counting.

quency count below a frequency threshold θ are eliminated, which means that there are three parameters b, N_e, θ to control the search.

To count the number of occurrences of a pattern in a graph transaction, the canonical labeling employed in 10) is adopted. However, only canonical label cannot solve the problem completely as shown in **Fig. 6**. Suppose that the pair A $\rightarrow$ B is registered as a pseudnode N in Fig. 6(a). How many times the pair N $\rightarrow$ B should be counted? If only canonical label is considered, the answer is 2 as shown in Fig. 6(b). However, N $\rightarrow$ B should be counted once. Our solution is to incorporate the canonical label with the node id set. This solution cannot be applied to count the frequency of pattern A – A – A in **Fig. 7**(a) as illustrated in Figs. 7 (b), (c), and (d). However, in the case of enumerating frequent induced subgraphs, it causes no problem.

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

5. Experiments

To assess the performance of the Cl-GBI approach, we conducted two experiments on both synthetic and real-world graph datasets. The proposed Cl-GBI algorithm was implemented in C^{++}. Since the current implementation is very naive, we do not evaluate the computation time. It should be noted that all graphs/subgraphs reported here are connected ones.

In the first experiment, we show that Cl-GBI is capable of finding all frequent patterns in a single large graph. An example of a single graph is shown in **Fig. 8**(a). The problem here is to find frequent induced subgraphs that occur at least 3 times in the graph. Figure 8(c) shows an example of frequent induced subgraph which has the support of 3.

The current algorithms that are designed for extracting frequent patterns in a single large graph (such as GBI[14], B-GBI[10], SUBDUE[3], or GREW[9], etc.) cannot discover the pattern shown in Fig. 8(c) because three occurrences of this pattern are not disjoint, but overlapping. Meanwhile, the complete graph mining algorithms (like AcGM[8], gSpan[13], etc.), in case that they are adapted to find frequent patterns in a single graph, also cannot find that pattern because of the monotonic nature. Since the pattern shown in Fig. 8(b) occurs only once in the graph and thus cannot be extracted, the pattern shown in Fig. 8(c) which is one of its super-graph is also unable to be found.

The proposed Cl-GBI, on the other hand, can find 35 frequent induced subgraphs, including the one shown in Fig. 8(c), with $b = 3, N_e = 5$.

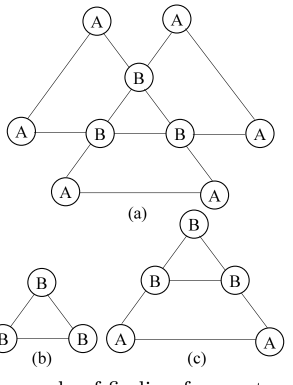
Fig. 8 An example of finding frequent patterns in a single graph.

In the second experiment, we evaluate the performance of Cl-GBI on the promoter dataset from UCI repository[1] (promoter sequences only) and the hepatitis dataset provided by Chiba University (response to interferon therapy only). These promoter and hepatitis datasets were converted to undirected and di-

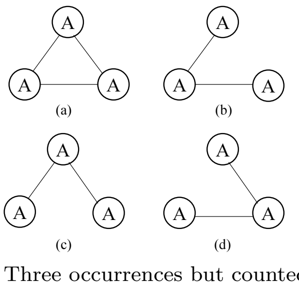
Fig. 7 Three occurrences but counted once.

rected graphs, respectively. The former contains of 53 undirected graphs having the same size of 57 (note: for this experiment only), while the latter has 56 directed graphs having the average size of 75.4. We compare the number of frequent induced subgraphs discovered by Cl-GBI with B-GBI[10] and AcGM[8] given the minimum support threshold of 50%. B-GBI is an improved version of GBI, while AcGM can extract all frequent induced subgraphs.

Table 1 Number of frequent induced subgraphs obtained from the promoter dataset.

Algorithm	#discovered patterns	Parameters
Cl-GBI	2164	b =5, $N_e = 5$
Cl-GBI	4638	b =6, $N_e = 10$
B-GBI	580	$b = 5$
AcGM	4638	N/A

Table 1 shows some experimental results obtained from the promoter dataset. It is shown that Cl-GBI can find more frequent patterns than B-GBI given the same beam width. Also, Cl-GBI can find all the frequent patterns in graph data by selecting suitable parameters.

For the hepatitis dataset, Cl-GBI extracts 4488 frequent patterns with $b = 5$, $N_e = 10$ and B-GBI finds 870 frequent patterns with $b = 5$. Meanwhile, since AcGM has not been implemented to handle directed graphs, we cannot use it to find frequent patterns in this graph dataset. However, suppose that the graphs in this dataset are undirected, AcGM cannot give the results due to the large graph size.

6. Conclusion

A novel algorithm, Cl-GBI, was introduced for the purpose of discovering typical patterns in either a single large graph or graph databases. The proposed method employs a "chunkingless" strategy which helps overcome the problem of overlapping subgraphs. Also, Cl-GBI can give the number of occurrences as well as the positions of patterns in each transaction of the graph data. Experiments conducted on both synthetic and real-world graph data confirm its effectiveness.

References

1) Blake, C.L., Keogh, E., and Merz, C.J.: UCI Repository of Machine Learning Database, http://www.ics.uci.edu/mlearn/MLRepository.html (1998).
2) Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J.: *Classification and Regression Trees*, Wadsworth & Brooks/Cole Advanced Books & Software (1984).
3) Cook, D. J., Holder, L. B.: Substructure Discovery Using Minimum Description Length and Background Knowledge, *Artificial Intelligence Research*, Vol. 1, pp. 231–255 (1994).
4) Fortin, S.: The Graph Isomorphism Problem, *Technical Report TR96-20*, Department of Computer Science, University of Alberta, Edmonton, Canada (1996).
5) Gaemsakul, W., Matsuda, T., Yoshida, T., Motoda, M., and Washio, T.: Classifier Construction by Graph-Based Induction for Graph-Structured Data, *Proc. of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pp. 52–62 (2003).
6) Huan, J., Wang, W., Washington, A., Prins, J., Shah, R., and Tropsha, A.: Accurate Classification of Protein Structural Families using Coherent Subgraph Analysis, *Proc. of the 9th Pacific Symposium on Biocomputing (PSB)*, pp. 411–422 (2004).
7) 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).
8) Inokuchi, A., Washio, T., Nishimura, K. and Motoda, H.: A Fast Algorithm for Mining Frequent Connected Subgraphs, *IBM Research Report RT0448*, Tokyo Research Laboratory, IBM Japan (2002).
9) Kuramochi, M., Karypis, G.: GREW–A Scalable Frequent Subgraph Discovery Algorithm, *Proc. of the IEEE International Conference on Data Mining (ICDM)*, in press (2004).
10) Matsuda, T., Motoda, H., Yoshida, T., and Washio, T.: Mining Patterns from Structured Data by Beam-wise Graph-Based Induction, *Proc. of the International Conference on Discovery Science (DS)*, pp. 422–429 (2002).
11) Quinlan, J.R.: Induction of Decision Trees, *Machine Learning*, Vol. 1, pp. 81–106 (1986).
12) Quinlan, J.R.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers (1993).
13) Yan, X., Han, J: gSpan: Graph-Based Structure Pattern Mining, *Proc. of the IEEE International Conference on Data Mining (ICDM)*, pp. 721–724 (2002).
14) Yoshida, K., Motoda, M.: CLIP: Concept Learning from Inference Pattern, *Artificial Intelligence*, Vol. 75, No. 1, pp. 63–92 (1995).