Mining Frequent Graph Patterns Considering Both Different Importance and Rarity of Graph Elements[*]

Gangin Lee and Unil Yun[**]

Department of Computer Engineering, Sejong University, Republic of Korea
ganginlee@sju.ac.kr, yunei@sejong.ac.kr

Abstract. Since frequent graph pattern mining was proposed, various approaches have been suggested by devising efficient techniques or integrating graph mining with other mining areas. However, previous methods have limitations that cannot reflect the following important characteristics in the real world to their mining processes. First, elements in the real world have their own importance as well as frequency, but traditional graph mining methods do not consider such features. Second, various elements composing graph databases may need thresholds different from one another according to their characteristics. However, since traditional approaches mine graph patterns on the basis of only a single threshold, losses of important pattern information can be caused. Motivated by these problems, we propose a new graph mining algorithm that can consider both different importance and multiple thresholds for each element of graphs. We also demonstrate outstanding performance of the proposed algorithm by comparing ours with previous state-of-the-art approaches.

Keywords: Frequent pattern mining, Graph mining, Graph enumeration, Multiple minimum supports, Weight constraint.

1 Introduction

Since the concept of frequent graph pattern mining was proposed to overcome the limitations of traditional frequent pattern mining approaches that perform mining operations with respect to transaction databases composed of simple items only, a variety of methods have been devised [1, 3, 5, 6] by suggesting novel techniques for performance improvement or effectively integrating graph mining with other mining fields. However, previous graph mining methods have faced problems that cannot consider the following issues in the real world: 1) the *rare item problem* [4] that not only items or patterns with high supports but also ones with relatively low supports may have valuable information, and 2) the *different importance problem* [7, 8] that

[*] This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under ICT/SW Creative research program supervised by the NIPA (National ICT Industry Promotion Agency) (NIPA-2014-H0502-14-3008) and the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF No. 2013-005682).

[**] Corresponding author.

elements obtained from the real world have their own importance or weights different from one another according to their characteristics. Motivated by these issues, we propose a new algorithm that can mine Weighted Rare Graph patterns (WRGs) considering both different importance and multiple thresholds depending on the features of elements, called *WRG-Miner*. In addition, we demonstrate that our algorithm outperforms previous state-of-the-art ones through various performance evaluation tests.

2 Related Work

In frequent pattern mining, multiple minimum support threshold-based various researches [1, 2, 3, 4] have been conducted to solve the *rare item problem*. *MSApriori* [4] is an initial solution based on a level-wise manner. After that, a tree search-based algorithm, *CFP-growth* [1], and its advanced version, *CFP-growth++* [2], were proposed. Although they are possible to mine rare patterns applying multiple minimum support thresholds, they cannot consider different importance characteristics and their coverages are limited to simple itemset-based databases. *WFIM* [8] and *WIT* [7] are tree-based frequent pattern mining algorithms considering importance of items' own, but they cannot solve the *rare item problem* and process complex data such as graph databases.

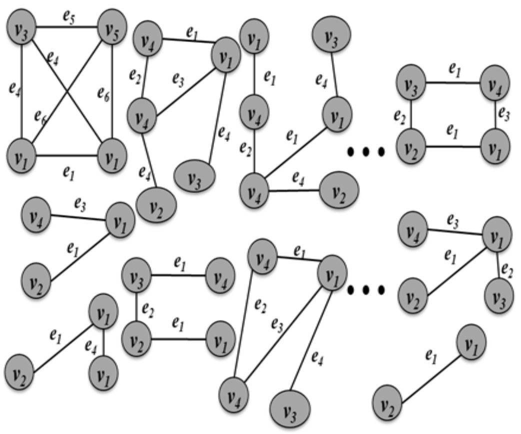
As one of the fundamental graph mining algorithms, *Gaston* [6] is known as the most efficient method in terms of runtime speed. The algorithm, which is an effective integration of multiple enumeration methods for path, free-tree, and cyclic graph forms, improves its mining performance by utilizing its own special techniques and list-based data structure. *FGM-MMS* [3] mines frequent and rare graph patterns by applying multiple minimum support constraints in a graph mining environment, but it still has a problem that cannot consider different importance or weights for each element within graphs.

3 Mining Weighted Rare Graph Patterns from Graph Databases with Multiple Minimum Support Thresholds

In this section, we describe techniques for applying multiple minimum support thresholds and importance characteristics of graph elements into a frequent graph mining environment and explain strategies for preventing fatal problems such as pattern losses that can be caused in the mining process. In addition, we also show how the proposed algorithm is performed through its overall operational procedure.

3.1 Employing Element Importance and Multiple Minimum Support Thresholds in Frequent Graph Pattern Mining

Fig. 1 shows an example of a graph database including the information of multiple minimum support thresholds and edge weights. To apply different importance for each element composing graphs, we consider mining graph patterns from graph databases that assign a different weight value to each edge of graphs. As a result, we can obtain a set of graph patterns considering both supports and weights of edges connected among nodes. The weighted support of a graph pattern is calculated as follows.



Vertex label	Support	Threshold
v_1	$Sup(v_1)$	δ_1
v_2	$Sup(v_2)$	δ_2
v_3	$Sup(v_3)$	δ_3
…	…	…
v_n	$Sup(v_n)$	δ_n

Edge label	Support	Threshold	Weight
e_1	$Sup(e_1)$	δ_{n+1}	w_1
e_2	$Sup(e_2)$	δ_{n+2}	w_2
e_3	$Sup(e_3)$	δ_{n+3}	w_3
…	…	…	…
e_m	$Sup(e_m)$	δ_{m+n}	w_m

Fig. 1. Example of a graph database with weight and multiple minimum support information

Definition 1. (*Weighted support of a graph pattern*) Let $V_G = \{v_1, v_2, \ldots, v_m\}$, $E_G = \{e_1, e_2, \ldots, e_n\}$, and $W_G = \{w_1, w_2, \ldots, w_n\}$ be a set of vertices in a graph pattern, G, a set of edges in G, and a set of edge weights in G. Then, the weighted support of G, $Wsup(G)$ is calculated as follows: $Wsup(G) = Sup(G) * Avg(W)$, where $Sup(G)$ is a support of G and $Avg(W)$ is an average weight of W. If $Wsup(G)$ is not lower than a given minimum support threshold, G is regarded as a weighted frequent graph pattern.

In order to consider the *rare item problem* in the graph pattern mining area, we need multiple minimum support thresholds for each element (vertex and edge) composing a given graph database.

Definition 2. (*Minimum element support threshold*)) Let $GDB = \{Gtr_1, Gtr_2, \ldots, Gtr_k\}$ be a given graph database composed of multiple graph transactions, *Gtr*s, $V_{GDB} = \{v_1, v_2, \ldots, v_x\}$ be a set of separate vertices included in *GDB*, and $E_{GDB} = \{e_1, e_2, \ldots, e_y\}$ be a set of separate edges in *GDB*. Then, a minimum element support threshold for each element (v or e), δ_i ($1 \leq i \leq x + y$) is set as a value specified by a user.

If a weighted support of any element is lower than the corresponding δ value, it becomes a useless one. Consequently, the threshold of a graph pattern is determined as follows.

Definition 3. (*Minimum graph support threshold*) Given a graph pattern, G, a set of vertices in G, $V_G = \{v_1, v_2, \ldots, v_n\}$, and a set of edges in G, $E_G = \{e_1, e_2, \ldots, e_m\}$, a minimum graph support threshold for G, $MGST(G)$, is set as the minimum value of all the δ values in G. Therefore, if $Wsup(G)$ is lower than $MGST(G)$, it becomes an invalid graph pattern.

By calculating the threshold for a graph pattern in the above manner, we can consider the rarity of the graph's each element. In other words, only weighted frequent graph patterns satisfying their own *MGST* conditions are finally regarded as valid results.

3.2 Maintaining the Correctness of the Proposed Algorithm

Through the aforementioned conditions, we can obtain a set of graph patterns that completely consider both the weight and rarity of elements. However, if we simply apply such conditions into the mining process without any additional considerations, fatal pattern losses can be caused since these conditions violate the anti-monotone property (or the downward closure property). Therefore, to guarantee not only the efficiency of the proposed algorithm but also its correctness, we employ 1) an overestimated weight method and 2) an underestimated minimum support method. In the first method, the maximum edge weight within a given graph database, called *MaxW*, is used instead of the real average weight factor used in Definition 1. That is, given a graph pattern, *G*, multiplying *Sup*(*G*) by *MaxW*, called $Wsup_{over}(G)$, is first applied in the mining process. Through such overestimation method, we can maintain the anti-monotone property and prevent unintended pattern losses by the weight constraints. In the second method, among all the thresholds in Definition 2, we set the least value that does not violate the property and apply it into the mining process. Let $L = \{\delta_1, \delta_2, \ldots, \delta_x\}$ be a list of all the elements' δ values in *GDB* that are sorted in the descending order of their values. Then, starting from the last element in the list, we check whether or not the overestimated weighted support of the element is higher than its own δ value. If there is the first element satisfying this condition, its δ value becomes an underestimated minimum support threshold, called *Least Minimum Support* (*LMS*).

If any graph pattern does not satisfy the *LMS* condition, the pattern and all of its possible supersets become useless ones; hence, it can permanently be pruned in advance. Note that graph patterns obtained through the above conditions are candidate patterns, not final results. Among them, only partial ones satisfying the corresponding *MGST* conditions in Definition 3 finally become valid results.

```
Input : a graph database, GDB, a list of minimum element support thresholds, L
Output : a set of WRGs, S
Procedure: WRG-Miner
01. a set of vertices, $V \leftarrow$ vertices such that each of their supports $\geq LMS$;
02. a set of edges, $E \leftarrow$ edges such that each of their supports * $MaxW \geq LMS$;
03. for each vertex, $v_i$ in $V$
04.     a graph pattern, $G \leftarrow v_i$; a set of edges, $E' \leftarrow$ edges that can be attached to $v_i$ in $E$;
05.     $S \leftarrow S \cup$ call WRG-Growth($G$, $E'$);
Function: WRG-Growth(G,E)
06. for each edge, $e_i$, in $E$
08.     if $G$ is a path or free tree: an expanded graph of $G$, $G' \leftarrow G \cup e_i$ and the vertex in $e_i$;
09.     else if $G$ is a cyclic graph: $G' \leftarrow G \cup e_i$;    //only a cyclic edge
10.     if $Wsup_{over}(G') \geq LMS$
11.         if $Wsup(G') \geq MGST(G')$, $S \leftarrow S \cup G'$;
12.         a set of edges, $E' \leftarrow$ edges that can be attached to $G'$;
13.         $S \leftarrow S \cup$ call WRG-Growth($G'$, $E'$);
```

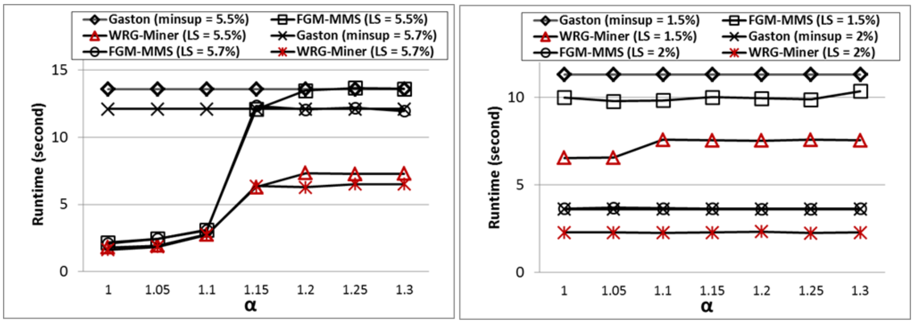
Fig. 2. Procedure of *WRG-miner*

3.3 WRG-Miner Algorithm

Fig. 2 shows the overall procedure of the proposed algorithm. After finding valid vertices and edges from a given graph database (lines 1-2), the algorithm performs graph pattern growth processes for mining WRGs with respect to each vertex (lines 3-5). In this phase, *WRG-Miner* continues to expand graphs for each edge considering their current states (lines 6-9). For each expanded graph, if the overestimated weighted support of the graph is smaller than *LMS*, it is permanently pruned (line 10). If the graph pattern's real weighted support is not lower than its corresponding *MGST* value, it is regarded as a valid result and the algorithm outputs it (line 11). Once the graph pattern has an overestimated weighted support higher than or equal to *LMS*, growth operations for the pattern are recursively conducted regardless of whether it is really outputted or not (lines 12-13).

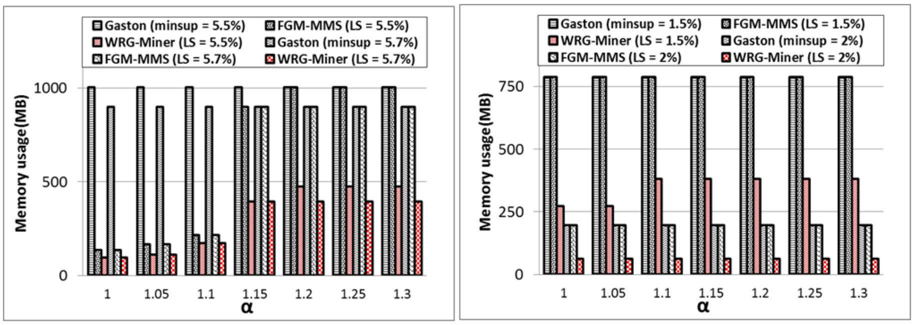
4 Performance Evaluation

In this section, the proposed algorithm is compared to *FGM-MMS* that is a state-of-the-art graph mining algorithm based on multiple minimum support constraints, and *Gaston* that is a well-known fundamental graph mining algorithm, with respect to real datasets, DTP and PTE [5]. Edge weight ranges of the datasets were set between 0.5-0.8. All the algorithms were written in C++ and executed in an environment with 3.33GHz CPU, 3GB RAM, and Windows 7 OS.



(a) DTP (b) PTE

Fig. 3. Runtime results of DTP and PTE



(a) DTP (b) PTE

Fig. 4. Memory usage results of DTP and PTE

To set the δ value for each element of graph datasets, the methodology employed in the literature [1, 2, 3, 4] was applied. That is, for each element, e_i, $\delta_i = MAX(\beta$ *

Sup(e_i), *LS*), where *LS* is the lowest one among all the δ values and is set to the same as the threshold of *Gaston* for reasonable comparisons. $\beta = 1/\alpha$ ($0 < \beta \leq 1$, $1 \leq \alpha$) is a variable that represents how closely the real support of each element is related to its own threshold value. That is, as β becomes closer to 1, δ is more likely to be assigned as a value more similar to *Sup*(e_i) rather than *LS*.

As shown in Figs. 3 and 4, the proposed algorithm guarantees the best runtime and memory usage performance regardless of the α settings because it can selectively mine more valuable graph patterns considering both rarity and importance of graph elements. Moreover, at lower α values, our *WRG-Miner* has better mining efficiency. Meanwhile, *Gaston* always shows the worst result that is the same regardless of α since it cannot consider the rarity of graph patterns. Although *FGM-MMS* has good performance at lower α values, it falls behind the proposed algorithm in every case since it does not consider element importance different from one another and has to mine all of the possible rare graph patterns regardless of their importance degrees.

5 Conclusion

In this paper, we proposed a new approach that mined useful graph patterns with high importance and rarity by considering multiple thresholds and weights different from each element of graphs. In addition, by devising the techniques for preventing unintended graph pattern losses occurring in the process of applying such conditions, we also guaranteed the correctness of out algorithm. The result of performance evaluation in this paper reported that out method outperformed the previous state-of-the-art ones in terms or both runtime and memory usage.

References

[1] Hu, Y.H., Chen, Y.L.: Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism. Decision Support Systems 42(1), 1–24 (2006)

[2] Kiran, R.U., Reddy, P.K.: Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms. In: EDBT, pp. 11–20 (2011)

[3] Lee, G., Yun, U.: Frequent Graph Mining Based on Multiple Minimum Support Constraints. In: Park, J.J.(J.H.), Adeli, H., Park, N., Woungang, I. (eds.) Mobile, Ubiquitous, and Intelligent Computing. LNEE, vol. 274, pp. 19–23. Springer, Heidelberg (2014)

[4] Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: KDD, pp. 337–341 (1999)

[5] Nijssen, S., Kok, J.N.: A quickstart in frequent structure mining can make a difference. In: KDD, pp. 647–652 (2004)

[6] Samiullah, M., Ahmed, C.F., Fariha, A., Islam, M.R., Lachiche, N.: Mining frequent correlated graphs with a new measure. Expert Systems with Applications 41(4), 1847–1863 (2014)

[7] Vo, B., Coenen, F., Le, B.: A new method for mining Frequent Weighted Itemsets based on WIT-trees. Expert Systems with Applications 40(4), 1256–1264 (2013)

[8] Yun, U.: On pushing weight constraints deeply into frequent itemset mining. Intelligent Data Analysis 13(2), 359–383 (2009)