Mining Frequent Patterns with Multiple Item Support Thresholds in Tourism Information Databases

Yi-Chun Chen, Grace Lin, Ya-Hui Chan and Meng-Jung Shih*

Advanced Research Institute, Institute for Information Industry, Taipei, 105 Taiwan, R.O. C.
{divienchen, gracelin, yhchan, mengjungshih}@iii.org.tw

Abstract. Frequent pattern mining is an important model in data mining. Certain frequent patterns with low minimum support can provide useful information in many real datasets. However, the predefined minimum support value as a threshold needs to be set properly, or it may cause rare item problem. A too high threshold causes missing of rare items, whereas a too low threshold causes combinatorial explosion. In this paper, we proposed an improved FP-growth based approach to solve the rare item problem with multiple item supports, where each item has its own minimum support. Considering the difficulty of setting appropriate thresholds for all items, an automatic tuning multiple item support (MIS) approach is proposed, which is based on Central Limit Theorem. A series of experimental results on various tourism information datasets shows that the proposed approach can enhance frequent pattern mining with better efficiency and efficacy.

Keywords: frequent pattern mining, multiple item support, automatic turning minimum support

1 Introduction

Frequent patterns are an important class of regularities that exist in databases. Since it was first introduced in [1], the problem of mining frequent patterns has received a great deal of attention [3]. Most of the frequent pattern mining algorithms (e.g., Apriori [2] and FP-growth [4]) use the single minimum support framework to discover complete set of frequent patterns, where the setting of minimum support (min_sup) plays the key role to this model's success. The frequent patterns discovered with this framework satisfy *downward closure property*. That is, "all non-empty subsets of a frequent pattern must also be frequent." This property holds the key for minimizing the search space in all of the single min_sup based frequent pattern mining algorithms [2, 3]. However, this algorithm has a strong assumption that all items in the data are of the same nature and/or have similar frequencies in the database. This is generally way from features of data in real applications. In many applications, some items appear very frequently in the data, while others rarely appear. A valuable part is on these rare items, which appear with low frequency and may be missed easily. Frequent patterns containing rare items usually can reveal information with high profitable potential for further decision support. This kind of phenomenon is often seen in

*: Corresponding author

consumer market. The necessities, consumables and low-price products such as bread or jam are bought frequently, but luxury goods, electric appliance and high-price such as bed or fridge are generally bought once for a long period. In such a situation, if the min_sup is set too high, all of the frequent patterns containing rare items will be missed. On the other hand, if the min_sup is set too low, many meaningless frequent patterns will be included and misleading focus. The problem to find mining frequent patterns containing both frequent and rare items with “single min_sup framework” is so called as the *rare item problem*.

For addressing rare item problem, B. Liu et al.[7] proposed mining frequent patterns with “multiple min_sup framework”, with which each pattern can satisfy a different min_sup depending upon the items within it. The frequent patterns discovered through “multiple min_sup framework” do not satisfy downward closure property and therefore deemed not applicable for minimizing the search space in multiple min_sup based frequent pattern mining algorithms. An Apriori-based algorithm known as Multiple Support Apriori (MSApriori) was proposed in this literature to find frequent patterns with “multiple min_sup framework”[7]. Also, two FP-growth based algorithms, Conditional Frequent Pattern-growth (CFP-growth) [5] and CFP-growth++ [6], have been proposed to mine frequent patterns. Since downward closure property no longer holds in “multiple min_sup framework,” the CFP-growth algorithm [5] has to carry out exhaustive search in the constructed tree structure. In [6], it proposed an improved CFP-growth approach, CFP-growth++, by introducing four pruning techniques to reduce the search space. Multiple min_sup framework is reasonable but brings another question: users generally have trouble specifying “good” support value for each item--they need constantly tune the support value. Updating items’ min_sup is a costly work—it takes time and efforts to scan database and then to execute the mining algorithm once again—and thus not appropriate to rerun so frequently. Therefore, it is necessary to design a maintenance approach for specifying min_sup automatically.

This study proposes an automatic tuning multiple item support (MIS) approach, with an illustrative example on tourism information database. We firstly categorized the data from the tourism information database to five classes, who, what, when, where and why, according to predefined ontology structure which constructed by domain experts. Fig. 1 illustrates the ontology schema applied in this case. As a general database in real world, tourism information database contains abundant data of popular sites on holidays, with significantly less records on items not so popular—frequencies of items vary widely and it is a rare item problem. Moreover, the cost of tuning multiple minimum supports is far higher than those of tuning single minimum support, because it is to tune all min_sup thresholds for all different items, which is large in number. Therefore, we developed an improved CFP-growth++ approach to solve the rare item problem with multiple item supports. An automatic tuning MIS approach is designed based on the Central Limit Theorem [8]. A series of experimental results on various tourism information datasets shows that the proposed approach can enhance frequent pattern mining with better efficiency and efficacy.

The remaining of the paper is organized as follows. In section 2, we explain the necessary background. In section 3, we proposed improved CFP-growth++ approach to mine both frequent and rare patterns. Experimental results are discussed in section 4. Finally, the conclusion and future work is drawn in section 5.

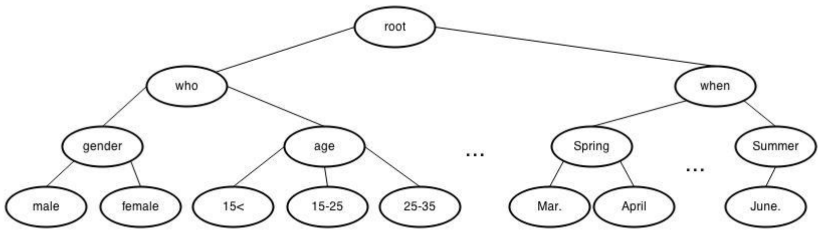


Fig. 1. An ontology schema for tourism information database

2 Problem Definition

In this section, we discuss rare item problem and extended version of frequent patterns on multiple min_sups.

Table 1. Encoded transaction database.

TID	Items	Items (ordered by MIS value)
1	{111, 121, 211, 21*}	{21*, 111, 211, 121}
2	{111, 121, 221, 22*}	{111, 22*, 221, 123}
3	{111, 123, 212, 21*}	{21*, 111, 212, 123}
4	{112, 122, 222, 22*}	{112, 22*, 222, 122}
5	{112, 122, 211, 21*}	{112, 21*, 211, 122}
6	{112, 122, 222, 22*}	{112, 22*, 222, 122}
7	{111, 123, 211, 21*}	{21*, 111, 211, 123}
8	{112, 123, 211, 21*}	{112, 21*, 211, 123}
9	{112, 123, 222, 22*}	{112, 22*, 222, 123}
10	{112, 123, 212, 21*}	{112, 21*, 212, 123}

2.1 Rare Item Problem

There are mostly non-uniform in nature containing both frequent and rare items in real world datasets. If the frequencies of items in a database vary widely, the following issues will be encountered while mining frequent patterns under single min_sup framework:

1. If min_sup is set too high, the frequent patterns containing rare items will not be exploited.
2. If min_sup is set too low, it can find frequent patterns that involve both frequent and rare items. However, it can also cause combinatorial explosion, generating too many meaningless frequent patterns.

Example 1. Consider the database shown in Table 1. At high min_sup, say min_sup = 4, we will miss the frequent patterns involving the rare items {221} and {222}. In order to exploit the frequent patterns containing {221} and {222}, we have to specify low min_sup.

2.2 Extended Version of Frequent Patterns

In order to face the rare item problem, B. Liu et al. [7] proposed the extended version of mining frequent patterns with “multiple min_sup framework” to solve this problem. In this extended version, each item in the transaction database is specified with a support constraint known as *minimum item support* (MIS, in short). Moreover, the min_sup of a itemset is represented with minimal MIS value among all its items.

$$MIN(\mathrm{X}) = \min(MIS(i_1), MIS(i_2), \ldots, MIS(i_k)) \quad (1)$$

where $X = \{i_1, \ldots, i_k\}$, $1 \leq \mathrm{k} \leq \mathrm{n}$, is a pattern and MIS($i_j$), $1 \leq \mathrm{j} \leq \mathrm{k}$, means the MIS of an item $i_j \in \mathrm{X}$.

This extended model enables users to generate rare item rules without causing frequent items to generate too many meaningless patterns. However, in real-world applications, users cannot specify applicable min_sup at once and always tune MIS of each item constantly. It is very time-consuming and costly because it must rescan database many times. Therefore, in this paper, an automatic tuning MIS approach is proposed. The concept of the Central Limit Theorem is utilized to specify MIS values automatically. We clarify the Central Limit Theorem in Theorem 1.

Theorem 1. (Central Limit Theorem)
Given certain conditions, the arithmetic mean of a sufficiently large number of iterates of independent random variables, each with a well-defined expected value and variance, will be approximately normally distributed. [8]

That is, suppose that a sample is obtained containing a large number of observations, the central limit theorem says that the computed values of the average will be distributed according to the normal distribution. Moreover, in our ontology, it can divide into different taxonomies and each of them is independent of each other. According to the 68-95-99.7 rule which means nearly all values lie within three standard deviations of the mean in a normal distribution, we can know that if MIS value = mean-standard deviation, approximately 84% item combinations will be found. Therefore, it can avoid that occur combinatorial explosion, producing too many meaningless frequent patterns. We define the MIS value of each item as follows:

$$\mu(i_j) = \frac{1}{N}\sum_{j=1}^{N} Sup(i_j) \quad (2)$$

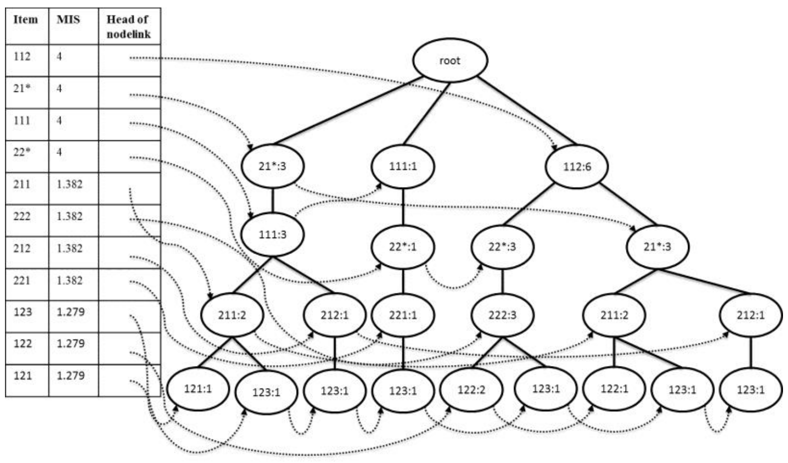
$$\mathrm{MIS}(i_j) = \mu(i_j) - \sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(Sup(i_j) - \mu(i_j)\right)^2} \quad (3)$$

$\mu(i_j)$ means the average frequency of the items $i_1, \ldots, i_N$, where items $i_1, \ldots, i_N$ belong to the same level nodes of each category in the ontology.

3 Algorithm

The proposed approach is an improvement over CFP-growth++ algorithm proposed in [6]. The basic concepts of the algorithm can be decomposed into three stages: (1) construction of MIS-tree (2) generating compact MIS-tree and (3) mining frequent patterns from the compact MIS-tree.

A detailed description of the algorithm is given in Algorithm 1 and Algorithm 2. First, we construct MIS-tree. The MIS-tree consists of two components: prefix-tree structure and MIS header table. The prefix-tree structure is similar to FP-tree structure [4]. The difference is that items in the MIS-tree are ordered in descending order according to their MIS values. Each entry in the MIS header table consists of three fields: item, the MIS value of each item, and head of node link which point to the first node in the MIS-tree with that item. The following example is used to illustrate the MIS-tree construction process.



Item	MIS	Head of nodelink
112	4	
21*	4	
111	4	
22*	4	
211	1.382	
222	1.382	
212	1.382	
221	1.382	
123	1.279	
122	1.279	
121	1.279	

Fig. 2. Initial MIS-tree

Example 2. Following the example above, let us consider the database shown in Table 1. The MIS value of each item is shown in the MIS header table in Fig. 2, which is computed by using equation 3. According to Algorithm 1, the items in each transaction (see the third column in Table 1) and the order of item list in MIS header table are arranged in decreasing order according to their MIS value.

Next, a MIS-tree is created as follows. First, the root node is created. Then, the scan of the first transaction leads to the construction of the first branch of the tree with four nodes <21*:1>, <111:1>, <211:1> and <121:1>. The second transaction containing the items '111', '22*', '221' '123' will result in a branch where '111' is linked to root node, '22*' is linked to '111', '221' is linked to '22*' and '123' is linked to '221'. The remaining transactions in TDB can be done in the same way. To facilitate tree traversal, a MIS header table is built in which each item points to its occurrences in the MIS-tree via the head of node link. Moreover, nodes with the same item are linked in sequence via such node links. After scanning all the transactions in Table 1, we will get the initial MIS-tree shown in Fig. 2.

To decrease the search space, the compact MIS-tree is generated by pruning items with supports less than MIN value and merging nodes with the same item name. For example, we remove the nodes with items, 121 and 221, and the complete and compact MIS-tree is shown in Fig. 3.

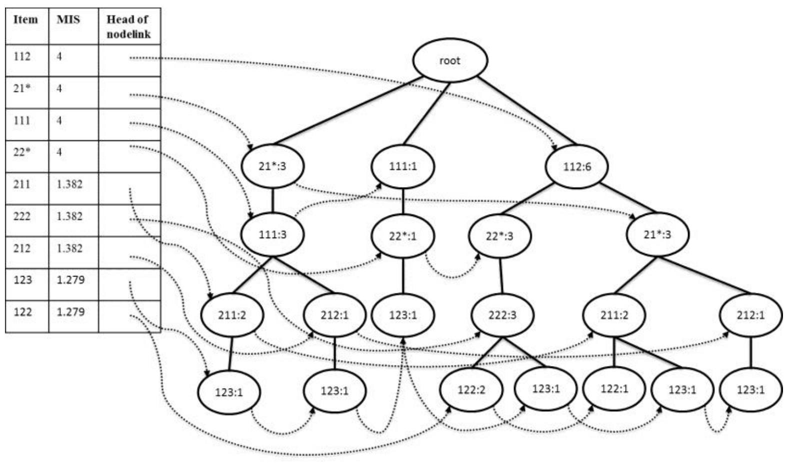


Fig. 3. Compact MIS-tree

```
Input: a transaction database DB and an ontology
Output: MIS-tree
Method:
1. Create the root of a MiS-tree, T, and label it as null
2. For each Transaction, $t_n$, in encoded TDB do
3.  Sort all items in $t_n$ according to their MIS(i)in
    deccending order
4.  Count the support values of any item i, denoted as
    Sup(i) in $t_n$.
5.  Insertion($t_n$[p|P], T), where p is the first element
    and P is the remaining list.
6. End for
7. For each item ~~f~~ in unfrequent itemset ~~F~~ do where ~~F~~ is
   the set of items with support smaller than MIN(~~F~~)
8.  Delete the entry in the header table with item_name =
    ~~f~~
9.  Pruning(T, ~~f~~)
10. End for
11. Get MIN frequent item header table
12. Merge(T)
Procedure Insertion($t_n$[p|P], T)
1. While P is not empty
```

```
2.  If T has a child N = p
3.   N.count++
4.  Else
5.   Creat a new node N, N.count =1
6.   Let its parent link be linked to T
7.   Let its node-link be linked to the nodes with the
   same item-name via the node-link structure
8. End while
Procedure Pruning(T, $i_j$)
1. For each node n in the node-link $i_j$ in T do
2.  If n is a leaf
3.   Remove n
4.  Else
5.   Remove n && its parent link will be linked to its
   child
6. End for
Procdeure Merge(T)
1. For each $i_j$ in MIN frequent item header table do
2.  If there are childnodes with the same item name
3.   Merge these childnodes
4.   Set the count as the summation of these nodes' count
5. End for
```

Table 2. All conditional pattern bases and conditional frequent patterns

Item	MIS	Cond. Pattern bases	Cond. Frequent patterns
122	1.279	<112, 22*, 223: 2>, <112, 21*, 211: 1>	{222, 122:2}, {22*, 122:2}, {112, 122:2}, {22*, 222, 122:2}, {112, 222, 122:2}, {112, 22*, 122:2}, {112, 22*, 222, 122:2}
123	1.279	<21*, 111, 211:1>, <21*, 111, 212:1>, <111, 22*:1>, <112, 22*, 222: 1>, <112, 21*, 211:1>, <112, 21*, 212:1>	{112, 123:3}, {21*, 123:4}, {111, 123:3}, {22*, 123:2}, {211, 123:2}, {212, 123:2}, {112, 21*, 123:2}, {21*, 111, 123:2}, {21*, 211, 123:2}, {21*, 212, 123:2}
212	1.382	<21*, 111>, <112, 21*>	{21*, 212:2}
222	1.382	<112, 22*>	{112, 222:3}, {22*, 222:3}, {112, 22*, 222:3}
211	1.382	<21*, 111>, <112, 21*>	{21*,211:4}, {111, 211:2}, {112, 211:4}, {21*, 111, 211:2}, {112, 21*, 211:2}
22*	4	<111>, <112>	Φ
111	4	<21*>	Φ
21*	4	<112>	Φ
112	4	Φ	Φ

The procedure for mining the complete set of frequent patterns from compact MIS-tree is shown in Algorithm 2. The process of mining frequent patterns is described as follows. Consider the item ‘122’ that has lowest MIS among all items in the compact MIS tree. It occurs in two branches of compact MIS-tree, <112, 22*, 223, 122: 2>, <112, 21*, 211, 122: 1>. Consider the item ‘122’ as a suffix item , its conditional prefix paths are <112, 22*, 223: 2> and <112, 21*, 211: 1>, which form the conditional pattern base. As the compact MIS-tree is constructed, ‘122’ will have MIN value among all items in its conditional pattern base. Therefore, using MIS value of ‘122’ as the conditional min_sup, conditional MIS-tree is generated with <112, 22*, 223: 2> and <112, 21*, 211: 1>. The rightmost column of Table 2 lists all the frequent patterns. Similar process is repeated for other remaining items in the compact MIS-tree to discover the complete set of frequent patterns. The complete set of frequent patterns is shown in Table 2.

```
Algorithm 2 CFP-growth++
Input: MIS-tree, a set of MIN frequent items F, MIS($i_j$)
of each item $i_j$ in F
Output: the complete set of all $i_j$'s conditional frequent
itemsets
Method:
1.  For each item $i_j$ in the header of Tree do
2.   Set conditional min_sup, Cmin_sup = MIS($i_j$)
3.   If $i_j$ is a frequent item
4.    Generate pattern $\beta= i_j \cup \alpha$ with support = $i_j$.support.
5.    Construct $\beta$'s conditional pattern base and $\beta$'s
   conditional MIS-tree $T_\beta$
6.    If $T_\beta \neq \Phi$
7.     CFPGrowth++( $T_\beta$, $\beta$, Cmin_sup)
8.  End for
Procedure CFPGrowth++(Tree, $\alpha$, Cmin_sup)
1.  For each item $i_j$ in the header of Tree do
2.   Generate pattern $\beta= i_j \cup \alpha$ with support = $i_j$.support.
3.   Construct $\beta$'s conditional pattern base and $\beta$'s
   conditional MIS-tree $T_\beta$
4.   If $T_\beta \neq \Phi$
5.    If $T_\beta$ contains a single path P
6.     For each combination $\gamma$ of the nodes in the path P
   do
7.      Generate pattern $\gamma \cup \beta$ with support = minimum
   support count of nodes in $\gamma$.
8.     End for
9.    Else
10.     CFPGrowth++( $T_\beta$, $\beta$, Cminsup).
11.  End for
```

4 Experimental Results

This section describes a set of experiments performed to assess the benefit of our approach. We compare the method proposed in [7] to assign MIS values to items with our approach. The method [7] is as follows:

$$\mathrm{MIS}(i_j) = \begin{cases} \sigma \times f(i_j) & \sigma \times f(i_j) > MIN \\ MIN & otherwise \end{cases} \tag{4}$$

The $f(i_j)$ is the actual frequency of item i_j in the TDB. *MIN* denotes the user-specified least minimum item support value of all items. $\sigma(0 \leq \sigma \leq 1)$ is a parameter that controls how the MIS values for items should be related to their frequencies. If $\sigma = 0$, we have only one min_sup, MIN, which is the same as the traditional frequent pattern mining. If $\sigma = 1$ and $f(i_j) > MIN$, $f(i_j)$ is the MIS value for item i_j.The approaches are implemented in Java and all the experiments are performed in an Intel Core i7 2.90GHz with 7.7 GB of memory, and running on Windows 7.

For our experiments, we generated a number of data sets from the tourism information database to test our approach. For each point of interest (poi, in short), we used other four categories from ontology, who, when, why and what, which are utilized to describe poi to construct a dataset. Here, we use the results from one data set to illustrate. The others are similar and thus omitted. The number of transaction is 97. We set up MIN values =1. Fig. 4 shows the number of frequent patterns found. We let $\sigma = 1/\alpha$ and vary α from 1 to 5. We see from Fig. 4 that the number of frequent patterns is significantly reduced by the method proposed in [7] when α is not too large. The number of frequent patterns found by our approach is close to the average of number of frequent patterns found by the method proposed in [7]. This result indicates that it can prevent to generate meaningless frequent patterns by using our maintenance method for automatic min_sup specifying.

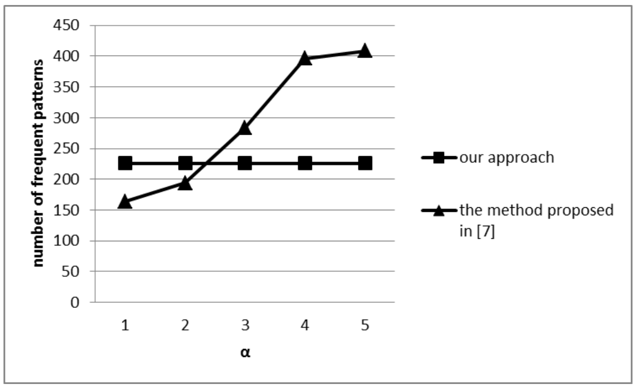


Fig. 4. POI dataset

5 Conclusions

This paper has proposed how to mine frequent pattern containing both frequent and rare items with multiple min_sups and presented a maintenance method for automatic min_siup specifying without rescanning database. We have implemented the improved CFP-growth++ method. By conducting experiments on tourism information dataset, the effectiveness of the maintenance method for automatic min_sup specifying is shown experimentally and practically.

There remain some problems that are worth studying in the future. First, the MIS-tree maintenance problem will be considered. Since the database is updated continuously, how to maintain the MIS-tree structure is an interesting problem. Second, we are planning to use the concept of frequent closed pattern to make the mining process efficiently.

References

1. R. Agrawal, T.Imieliński, and A. Swami.: Mining association rules between sets of items in large databases. SIGMOD Rec., 22:207-216, June 1993.
2. R. Agrawal and R. Srikant.: Fast algorithms for mining association rules in large databases. In Proc. Of the 20th International Conference on Very Large DataBases, 487-499, 1994.
3. J. Han, H. Cheng, D. Xin, and X. Yan.: Frequent pattern mining: current status and future directions. Data Min. Knowl. Discov., 15(1):55-86, 2007.
4. J. Han, J. Pei, Y. Yin, and R. Mao.: Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Min. Knowl. Discov., 8(1):53-87, 2004.
5. Y.-H. Hu and Y.-L. Chen.: Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism. Decis. Support Syst., 42(1):1-24, 2006.
6. B. U. Kiran and P. K. Reddy.: Novel techniques to reduce search space in multiple minimum supports-based frequent pattern mining algorithms. EDBT/ICDT '11 Proceedings of the 14th International Conference on Extending Database Technology, Pages 11-20, 2011
7. B. Liu, W. Hsu, and Y. Ma.: Mining association rules with multiple minimum supports. In KDD'99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 337-341, 1999.
8. J. Rice.: Mathematical statistics and data analysis (2nd ed.), Duxbury Press, ISBN 0-534-20934-3, 1995.