AN ALGORITHM FOR MINING ASSOCIATION RULES WITH WEIGHTED MINIMUM SUPPORTS

Yuchiang Li[1], Chinchen Chang[1, 2], and Jiehshan Yeh[3]
[1] *Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi 621, Taiwan*
[2] *Department of Information Engineering and Computer Science, Feng Chia University, Taichung 407, Taiwan*
[3] *Department of Computer Science and Information Management, Providence University, Taichung 433, Taiwan*

Abstract: Most existing algorithms employ a uniform minimum support for mining association rules. Nevertheless, each item in a publication database, even each set of items, is exhibited in an individual period. A reasonable minimum support threshold has to be adjusted according to the exhibition period of each *k*-itemset. Accordingly, this paper proposes a new algorithm, called WMS, for mining association rules with weighted minimum supports in publication databases. WMS discovers all frequent itemsets which satisfy their individual requirement of minimum support thresholds. WMS applies the group closure property to prune futile itemsets, to reduce the number of candidates generated, and thus to generate the candidate sets efficiently.

Key words: data mining, association rules, multiple minimum support

1. INTRODUCTION

Data mining techniques are typically used to gather hidden but potentially useful information from data in data warehouses. Such techniques have come to constitute an important field of research [6]. Mining association rules is one of the core tasks in solving many data mining problems. The Apriori algorithm [4] is the most famous means of mining association rules. The mining of association rules can be divided into two sub-problems: (1) to find all *frequent itemsets* that occur more frequently

than the minimum support threshold, and (2) to use these frequent itemsets to generate association rules. Other efficient methods have been studied to discover frequent itemsets in the past [1, 2, 8, 12].

Existing Apriori-like algorithms assume that all itemsets have a uniform minimum support. In practice, however, some items are important, but have low support. The straightforward approach is to lower the minimum support threshold to find these important but rare items. However, a lower minimum support may yield an excess of useless rules. The method of multiple minimum supports allows users to specify different minimum supports of different items, to discover rules that involve both frequent and rare items [11].

Available mining methods cannot efficiently handle a *publication-like* database. A publication database is also a transaction database wherein each item contains an individual exhibition period [9]. Two essential problems of the current model are as follows. (1) The exhibition period of each item is neglected, and (2) the support of each item is not evaluated on an equitable basis. For example, Fig. 1 refers to a publication database in a bookstore. Items $\{A\}$ and $\{B\}$ are exhibited from 2000 to the end of 2003. However, item $\{E\}$ is exhibited only in 2003. Classic mining techniques use an identical minimum support threshold and ignore the exhibition period of each item. The PPM algorithm has been proposed to solve this problem [9].

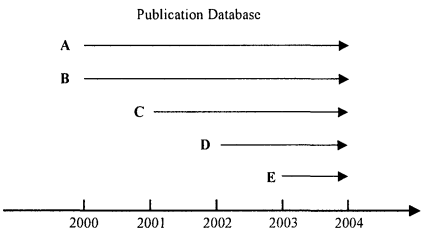

Figure 1. A publication database contains five items

LPMiner employs a linear function to tackle the problem of support that declines with the length of the itemsets. However, the linear function does not take into account the exhibition period of each item. Furthermore, the itemsets with equal length have the same minimum support threshold [13]. The PPM algorithm considers the exhibition period of each item, but it does not deal with the problem of support decrease with the length of itemsets. Moreover, the algorithm assumes that the termination times of all items are identical, but does not reflect on the exhibition period of each k-itemset. Therefore, this work proposes a novel weighted minimum supports (WMS) method that considers the individual exhibition period of each k-itemset.

Longer itemsets normally have shorter exhibition periods. Hence, WMS fulfills the requirement that the minimum support thresholds should be lower for longer itemsets. The reasonable exhibition period of a k-itemset is defined as the exhibition period shared by its k individual items. WMS also enables the exhibition period of each item to have a distinct cut-off date. This work focuses on generating frequent itemsets in the publication database.

2. RELATED WORK

Given a transaction database, the mining of association rules is defined to discover the important rules that relate *items*. In 1993, Agrawal *et al.* first defined such mining formally as follows [3]. Let $I = \{i_1, i_2, ..., i_m\}$ represent a set of items (*itemset*). Each transaction T is also an itemset such that $T \subseteq I$. Let DB be the transaction database, which is a set of transactions. Let X be an itemset; X is contained in T if and only if $X \subseteq T$. An association rule is used in the form $X \Rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \phi$ (For example, $I=\{A, B, C, D, E\}$, $X=\{A, C\}$, $Y=\{B, D\}$). An rule $X \Rightarrow Y$ has two essential properties, called *support* and *confidence*. The support of the rule $X \Rightarrow Y$, defined as $Sup(X \cup Y)$, is defined as the percentage of the transactions in DB that contain $X \cup Y$. The confidence of the rule $X \Rightarrow Y$, denoted as $Conf(X \Rightarrow Y)$, is c% when c% of the transactions in DB that contain X, also contain Y. The mathematical expression of confidence is $Conf(X \Rightarrow Y) = Sup(X \cup Y)/Sup(X)$. Given the minimum support (*minSup*) and minimum confidence (*minConf*) thresholds for the transaction database DB, the mining of association rules discovers all rules whose support, and the confidence in which, are greater than the two minimum thresholds, respectively. An itemset is called a frequent itemset when its support is above the *minSup* threshold.

Given a user-specified *minSup*, the Apriori algorithm takes advantage of the downward closure property to filter some infrequent itemsets beforehand [4]. The downward closure property is that any subset of a frequent itemset must be frequent; otherwise the itemset is infrequent. In each pass, Apriori constructs a candidate set of frequent itemsets. Then, the algorithm scans the entire transaction database to determine the frequent itemsets. A candidate k-itemset (an itemset with k items) is joined from two arbitrary frequent (k-1)-itemsets, whose first k-1 items are identical. If $k \geq 3$, Apriori uses the downward closure property to prune some infrequent itemsets. All remaining k-itemsets constitute candidate k-itemsets. The process is repeated until no candidate can be generated.

Apriori employs a single minimum support threshold for whole transaction database. In reality, the minimum support may not be uniform. In 1998, Cai *et al.* developed the notion of the multiple minimum support thresholds to reflect the importance of each item [5]. In 1999, Liu *et al.* described an algorithm that exhibits the sorted closure property to solve the problem that the downward closure characteristic cannot be employed in the multiple minimum supports scenario [11].

Example 2.1 Consider the three items $\{A\}$, $\{B\}$, and $\{C\}$ in a database. The individual minimum support threshold of each item is as follows.

$minSup(A) = 5\%$, $minSup(B) = 10\%$, and $minSup(C) = 15\%$.

The *minSup* threshold of an itemset is defined as the lowest support value among all items in the itemset. If the support of the itemset $\{B, C\}$ is 8%, then it is under both $minSup(A)$ and $minSup(B)$. $\{B, C\}$ is discarded because of downward closure. However, itemset $\{A, B, C\}$ may be frequent because $minSup(A)$ is 5%. The sorted closure property is used to solve the problem by sorting in order of ascending minimum support.

In 2000, Wang *et al.* presented a "pushing support constraints" method to "push" support constraints into the generation of the Apriori itemset to conserve the essence of Apriori and determine the minimum support for each itemset durning running time [15]. Recent research has developed numerous other mining association rules with multiple minimum supports [7, 9, 10, 13, 14]. In this work, the WMS algorithm is also a variant of mining association rules with multiple minimum supports.

3. WEIGHTED MINIMUM SUPPORTS (WMS)

The main idea behind the previous multiple minimum supports techniques is the assigning of an individual minimum support threshold to each item. Although the techniques can yield interesting itemsets with low support, they do not account for the lower support threshold of long itemsets. Short itemsets tend to be interesting when they have high support, whereas long itemsets may be of interest but occur relatively infrequently. To solve this problem, the LPMiner algorithm incorporates a support-constraining function, which reduces the support threshold as the length of the itemset increases. However, the minimum support thresholds of equally long itemsets are identical in LPMiner. This study proposes the Weighted Minimum Supports (WMS) algorithm, in which for each itemset, the minimum support is individually weighted, according to its period of exhibition.

3.1 Common Exhibition Period

In a publication database, the exhibition period of each individual itemset may be different, so measurements based on the assumption that all itemsets have an identical "life cycle" are biased. In this work, the exhibition period of each itemset is defined as a span of time granularities. Let a transaction database *DB* logically be divided into n partitions of unit time granularity (month, quarter or year, for example). The symbol db^{t_1,t_2} indicates the continuous portion of the database from partition P_{t_1} to partition P_{t_2}, where $t_2 \geq t_1 \geq 1$ and $t_1, t_2 \in N$. A k-itemset $I=\{i_1, i_2, ..., i_k\}^{t_1,t_2}$ denotes that the exhibition period of $\{i_1, i_2, ..., i_k\}$ is from P_{t_1} to P_{t_2}. For the entire database, given a *minSup*, the weighted minimum support (*WminSup*) threshold of each 1-itemset is $minSup*|db^{t_1,t_2}|/|DB|$, where $|db^{t_1,t_2}|$ indicates the number of transactions of db^{t_1,t_2} and $|DB|$ represents the number of transactions of *DB*. The weighted minimum support threshold of a k-itemset is derived from the common exhibition period of all k items in the k-itemset. Consider the k 1-itemsets $\{i_1\}^{t_{11},t_{12}}$, $\{i_2\}^{t_{21},t_{22}}$, ..., $\{i_k\}^{t_{k1},t_{k2}}$; the start time of the exhibition of the k-itemset $\{i_1, i_2, ..., i_k\}$ is $t_1 = \max\{t_{11}, t_{21}, ..., t_{k1}\}$ and the end time is $t_2 = \min\{t_{12}, t_{12}, ..., t_{k2}\}$. The period in which these k individual items occur at the same time is from t_1 to t_2. If $t_2 < t_1$, then the k-itemset does not appear in *DB*.

Example 3.1 Consider a database that contains five individual items. Figure 2 illustrates the exhibition periods of these five 1-itemsets. The common exhibition period of $\{B\}^{2000,2001}$, $\{C\}^{2001,2003}$ and $\{D\}^{2001,2002}$ is the exhibition period of $\{B, C, D\}$. Therefore, the start time of $\{B, C, D\}$ is $\max\{2000, 2001, 2001\} = 2001$ and the end time is $\min\{2001, 2003, 2002\} = 2001$. $\{B, C, D\}^{2001,2001}$ denotes that the exhibition period of $\{B, C, D\}$ is one year, 2001.

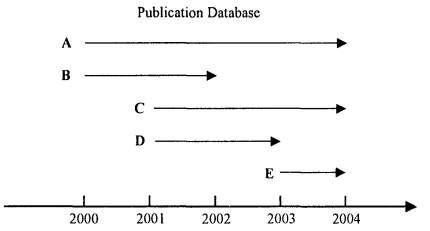

Figure 2. Exhibition period of each 1-itemset

The duration of the exhibition period of an itemset is turned into the minimum support weight. Users can effectively express various minimum support requirements for different rules, in light of the individual weight of each itemset. The use of weighted minimum supports can enable itemsets

with long exhibition periods to have higher minimum supports and those short exhibition periods itemsets to have lower minimum supports.

3.2 Candidate Generation

In [11], Liu *et al.* used the sorted closure property to solve the problem of mining with multiple minimum supports. However, in this work, the sorted closure property does not be applied.

Example 3.2 Consider three items $\{A\}$, $\{B\}$ and $\{C\}$ in a transaction database, with exhibition periods $\{A\}^{1,4}$, $\{B\}^{1,3}$, and $\{C\}^{3,4}$, respectively. The distributions of the records in the database stretch over four time units. Assume that all partitions include the same number of transactions. The minimum support for the entire database is 40%. Thus, *WminSup*(A) = 40%, *WminSup*(B) = 30%, *WminSup*(C) = 20% and *WminSup*(BC) = 10%. If the support of itemset $\{B\}$=15%, then the itemset is discarded using sorted closure. The occurrence count is below the minimum of all 1-itemset *WminSup* values, so the potentially frequent itemset $\{BC\}^{2,2}$ is pruned by using the sorted closure property.

A publication database is logically divided into n partitions with unit time granularity. *WminSup* of each partition can be derived easily from the user-defined *minSup*. The minimum *WminSup* is the minimum among all of these *WminSup*s. If the percentage of occurrences of a k-itemset is below the minimum *WminSup*, then all of its supersets must be infrequent. Hence, the k-itemset can be pruned. A superset of a k-itemset becomes a frequent itemset if the number of supports of the k-itemset exceeds the minimum *WminSup*. Sorted closure is irrelevant to WMS. Too many candidate itemsets will be generated if the minimum *WminSup* is the only pruned threshold. Therefore, this study describes another property, called the Group Closure Property (GCP), used to prune the number of candidate itemsets and generate the candidate set efficiently.

Group Closure Property (GCP)

Consider a database with n grains of time. The total number of distinct continuous exhibition periods is $n(n+1)/2$. Accordingly, each k-itemset can be assigned to one of the $n(n+1)/2$ groups such that each k-itemset in the same group has an identical exhibition period. That is, each k-itemset in the same group has an identical *WminSup*. After the k-th pass, WMS employs the minimum *WminSup* value to delete useless candidates. The remaining k-itemsets form a set RC_k. The set C_{k+1} is generated from RC_k. Let *WminSup*(g_i)

denote the *WminSup* value of the group g_i, and let *WminSup*($g_i \cap g_j$) denote the *WminSup* value of the common exhibition period of the two groups g_i and g_j, the generation of C_{k+1} proceeds as follows.

1. Sort the groups in order of descending *WminSup*. *Gr*(first) denotes the set of groups which is sorted, where *Gr*(first) = $\{g_1, g_2, \ldots, g_{n(n+1)/2}\}$. Select a group $g_i \in Gr$(first) in order.
2. Compute the common exhibition period of g_i with each other group g_j, where $i < j$. WMS sorts other groups in order of increasing *WminSup*($g_i \cap g_j$) value; it discards groups that do not share common exhibition period. *Gr*(second) represents the set of groups which is sorted.
3. Select a group $g_j \in Gr$(second) in order.
4. As required by the *WminSup*($g_i \cap g_j$), delete the k-itemsets in g_i whose occurrence counts are smaller than *WminSup*($g_i \cap g_j$).
5. Join each k-itemset in g_i with each k-itemset in g_j, whose occurrence counts are above the *WminSup*($g_i \cap g_j$) requirement, to generate candidate (k+1)-itemsets.
6. Select the next g_j and return to Step 4 until all groups have been chosen.
7. Delete all infrequent k-itemsets in g_i. Output frequent k-itemsets. Join each frequent k-itemset in g_i with each other to generate (k+1)-itemsets.
8. Select the next g_i and return to Step 2 until all groups have been selected.

In Step 4, if the number of occurrences of the k-itemset in g_i is smaller than *WminSup*($g_i \cap g_j$), then any superset of the k-itemset that has not yet be generated must be infrequent, because *Gr*(second) is sorted in order of increasing *WminSup*($g_i \cap g_j$). In Step 5, the candidate (k+1)-itemset is not produced if the support value is under the *WminSup*($g_i \cap g_j$) threshold. The method of joining artibrary two candidates is the same as that in Apriori. In Step 7, all k-itemsets which are in the same group exhibit the downward closure property.

Example 3.3 Consider a database in which the distribution of transactions stretches over four time grains. Assume that each partition contains the same number of transactions. The minimum support for the entire database is 40%. Let g^{t_1,t_2} expresses the group whose exhibition period is from t_1 to t_2. The first column in Table 1 lists the ten groups in the *Gr*(first) sequence from top to bottom. The second column presents the corresponding *Gr*(*second*) sequences. Assume that the support value of itemset $\{A, B\}^{1,4}$ is 15%. First, $\{A, B\}$ is joined with other 2-itemsets that in the group $g^{1,1}$ to generate the 3-itemset candidates. According to the group order $g^{2,2}$, $g^{3,3}$, ..., $g^{2,4}$, $\{A, B\}$ joins with other 2-itemsets until the process arrives at group $g^{1,2}$. Because, *WminSup*($g^{1,4} \cap g^{1,2}$) = 20% exceeds 15%, so $\{A, B\}$ is deleted from $g^{1,4}$ after the group $g^{1,2}$ is selected.

The pseudo-code for generating candidates is as follows.

Procedure candidate-gen(RC_{k-1})

```
01 Gr(first) = sort(G, WminSup(g_i ∈ G)); // G is the set of all groups
02 for each group g_i ∈ Gr(first) do
03    G = {g_j ∈ Gr(first) | i < j};
04    for each g_j ∈ G do
05       if WminSup(g_i ∩ g_j) = 0;
06          delete g_j; endif endfor
07    Gr(second) = sort(G, WminSup(g_i ∩ g_j) ≠ 0);
08    for each c_i ∈ g_i do
09       if c_i.count < WminSup(g_i ∩ g_j)
10          delete c_i; endif
11       for each c_j ∈ g_j do
12          if c_j.count < WminSup(g_i ∩ g_j)
13             next; endif
14          apriori-join(c_i, c_j);
15       endfor
16    endfor
17    for each c_i ∈ g_i do
18       if c_i.count < WminSup(g_i)
19          delete c_i; endif endfor
20    for each c_i ∈ g_i do
21       for each (c_j ≠ c_i) ∈ g_i do
22          apriori-join(c_i, c_j); endfor
23    endfor
24 endfor
```

Table 1. Example of the Gr(first) sequence and the Gr(second) sequence

Gr(first)	Gr(second) sequence, in order of increasing $WminSup(g_i \cap g_j)$
$g_1 = g^{1,4}$	$\{g^{1,1}, g^{2,2}, g^{3,3}, g^{4,4}, g^{1,2}, g^{2,3}, g^{3,4}, g^{1,3}, g^{2,4}\}$
$g_2 = g^{1,3}$	$\{g^{1,1}, g^{2,2}, g^{3,3}, g^{3,4}, g^{1,2}, g^{2,3}, g^{2,4}\}$
$g_3 = g^{2,4}$	$\{g^{2,2}, g^{3,3}, g^{4,4}, g^{1,2}, g^{2,3}, g^{3,4}\}$
$g_4 = g^{1,2}$	$\{g^{1,1}, g^{2,2}, g^{2,3}\}$
$g_5 = g^{2,3}$	$\{g^{2,2}, g^{3,3}, g^{3,4}\}$
$g_6 = g^{3,4}$	$\{g^{3,3}, g^{4,4}\}$
$g_7 = g^{1,1}$	None
$g_8 = g^{2,2}$	None
$g_9 = g^{3,3}$	None
$g_{10} = g^{4,4}$	None

In Lines 14 and 22, the join function is the same as the join step in Apriori. This function joins two distinct (k-1)-itemsets whose first k-2 items are identical. Lines 8 to 16 generate candidate k-itemsets when arbitrary two (k-1)-itemsets are in different groups, whereas from Lines 17 to 23 relate to the case in which all (k-1)-itemsets appear in the same group. In Line 22, although the itemsets in the same group satisfy the property of download closure, the exhibition periods of the subset of a frequent k-itemset may be different with the k-itemset. Thus, omit the pruning step following the joining step. The candidate generation function deletes in advance (k-1)-itemsets whose supersets must be infrequent. Then, the candidate k-itemsets

are generated. The pruning step is implemented only before the joining step, unlike in Apriori, whose pruning step is implemented only after the joining step. Lines 9-10, 12-13 and 18-19 are the pruning steps.

3.3 Weighted Minimum Supports

The WMS algorithm is based on a level-wise technique like Apriori. WMS passes over the database k or k+1 times to generate all frequent k-itemsets. In the first pass, each item is a candidate 1-itemset. WMS scans the database to count the supports of each candidate 1-itemset and determines whether these 1-itemsets are frequent. If the support values of the 1-itemsets are greater than or equal to the minimum *WminSup*, then these 1-itemsets are added to the set RC_1. In the pass k, C_k is generated from RC_{k-1}; then WMS scans the database to count the supports of each candidate k-itemset and determines whether they are frequent. The WMS algorithm pseudo-code is stated as follows.

Algorithm WMS
01 G = group();
02 C_1 = {All 1-itemsets};
03 count(C_1); // scan *DB*
04 $RC_1 = \{c \in C_1 \mid$ c.count $\geq$ minimum *WminSup*$\}$;
05 $F_1 = \{c \in RC_1 \mid$ c.count $\geq$ *WminSup*(c.group)$\}$;
06 **for** (k = 2; $RC_{k-1} \neq \phi$; k++) **do**
07 C_k = candidate-gen(RC_{k-1});
08 count(C_k); // scan *DB*
09 $RC_k = \{c \in C_k \mid$ c.count $\geq$ minimum *WminSup*$\}$;
10 $F_k = \{c \in RC_k \mid$ c.count $\geq$ *WminSup*(c.group)$\}$;
11 **endfor**
12 Answer = $\bigcup_k F_k$;

In Line 1, all itemsets are assigned to their corresponding groups, according to the distinct exhibition period of the itemsets. In Line 7, the set RC_{k-1} is used to generate candidate k-itemsets. WMS exploits the group closure property to prune some useless candidates and generate the candidate k-itemsets. Lines 3 and 8 scan the database and count the support value of the candidate k-itemsets.

4. CONCLUSIONS

In a publication database, each itemset has an individual exhibition period. Mining association rules on publication databases with a uniform minimum support is unpractical. The reasonable exhibition period of a k-itemset is the common exhibition period of its k individual items. The sensible minimum supports of distinct k-itemsets should be adjusted in the

light of their exhibition periods. This investigation introduced a novel approach, WMS, for mining association rules with weighted minimum supports that consider the exhibition period of each itemset in a publication database. WMS also accounts for the distinct end time of the exhibition period of each item. The proposed method mines frequent itemsets whose percentage of occurrences exceeds the individual *WminSup* threshold. WMS eliminates some useless itemsets using the group closure property to avoid generating an excess of candidates. In the future, we will consider the extension of the WMS algorithm to multiple level association rules.

REFERENCES

[1] R. C. Agarwal, et al. A tree projection algorithm for generation of frequent itemsets. *Journal of Parallel and Distributed Computing*, 2000, 61(3): 350~361

[2] C. C. Aggarwal and P. S. Yu, Mining associations with the collective strength approach. *IEEE Trans. Knowledge and Data Engineering*, 2001, 13(6): 863~873

[3] R. Agrawal, et al. Mining association rules between sets of items in large databases. 1993 *ACM SIGMOD Intl. Conf. on Management of Data*, Washington: 207~216

[4] R. Agrawal and R. Srikant, Fast algorithms for mining association rules. *20th Intl. Conf. on Very Large Data Bases*, 1994, Santiago: 487~499.

[5] C. H. Cai, et al. Mining association rules with weighted items. *1998 Intl. Database Engineering and Applications Symposium*, Cardiff: 68~77

[6] M. S. Chen, et al. Data mining: An overview from a database perspective. *IEEE Trans. Knowledge Data Engineering*, 1996, 8(6): 866~883

[7] F. L. Chung and C. L. Lui, A post-analysis framework for mining generalized association rules with multiple minimum supports. *6th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining Workshop on Post-Processing in Machine Learning and Data Mining*, 2000, Boston: 9~14

[8] J. Han, et al. Mining frequent patterns without candidate generation. 2000 *ACM-SIGMOD Intl. Conf. on Management of Data*, Dallas: 1~12

[9] C. H. Lee, et al. Progressive partition miner: An efficient algorithm for mining general temporal association rules. *IEEE Trans. Knowledge Data Engineering*, 2003, 15(4): 1004~1017

[10] C. L. Liu and F. L. Chung, Discovery of generalized association rules with multiple minimum supports. *4th European Conf. on Principles of Data Mining and Knowledge Discovery*, 2000, Lyon: 510~515

[11] B. Liu, et al. Mining association rules with multiple minimum support. *1999 ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, San Diego: 337~341

[12] J. S. Park et al. An effective hash-based algorithm for mining association rules. *1995 ACM*-SIGMOD *Intl. Conf. on Management of Data*, San Jose: 175~186

[13] M. Seno and Karypis, LPMiner: An algorithm for finding frequent itemsets using length-decreasing support constraint. *2001 IEEE Intl. Conf. on Data Mining*, San Jose: 505~512

[14] M. C. Tseng, et al. Maintenance of generalized association rules with multiple minimum support. *Joint 9th IFSA World Congress and 20th NAFIPS Intl. Conf.*, 2001, Vancouver: 1294~1299

[15] K. Wang, et al. Pushing support constraints into association rules mining. *IEEE Trans. Knowledge Data Engineering*, 2003, 15(3): 642~658