Progressive Weighted Miner: An Efficient Method for Time-Constraint Mining

Chang-Hung Lee[1], Jian Chih Ou[2], and Ming-Syan Chen[2]

[1] BenQ Corporation 18, Jihu Road, Neihu,Taipei 114, Taiwan, R.O.C.,
MichaelCHLee@BenQ.com

[2] Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, ROC
mschen@cc.ee.ntu.edu.tw, alex@arbor.ee.ntu.edu.tw

Abstract. The discovery of association relationship among the data in a huge database has been known to be useful in selective marketing, decision analysis, and business management. A significant amount of research effort has been elaborated upon the development of efficient algorithms for data mining. However, without fully considering the time-variant characteristics of items and transactions, it is noted that some discovered rules may be expired from users' interest. In other words, some discovered knowledge may be obsolete and of little use, especially when we perform the mining schemes on a transaction database of short life cycle products. This aspect is, however, rarely addressed in prior studies.

To remedy this, we broaden in this paper the horizon of frequent pattern mining by introducing a weighted model of *transaction-weighted association rules* in a time-variant database. Specifically, we propose an efficient *Progressive Weighted Miner* (abbreviatedly as *PWM*) algorithm to perform the mining for this problem as well as conduct the corresponding performance studies. In algorithm *PWM*, the importance of each transaction period is first reflected by a proper weight assigned by the user. Then, *PWM* partitions the time-variant database in light of weighted periods of transactions and performs weighted mining. Algorithm *PWM* is designed to progressively accumulate the itemset counts based on the intrinsic partitioning characteristics and employ a filtering threshold in each partition to early prune out those cumulatively infrequent 2-itemsets. With this design, algorithm *PWM* is able to efficiently produce weighted association rules for applications where different time periods are assigned with different weights and lead to results of more interest.

Index Terms: Data mining, time-constraint, time-variant, weighted association rules

1 Introduction

The discovery of association relationship among the data in a huge database has been known to be useful in selective marketing, decision analysis, and business management [6, 11]. A popular area of applications is the market basket analysis, which studies the buying behaviors of customers by searching for sets of items that are frequently purchased *either* together *or* in sequence. For a given pair of confidence and support thresholds, the problem of mining association rules is to identify all association rules that have confidence and support greater than the corresponding minimum support threshold (denoted as min_supp) and minimum confidence threshold (denoted as min_conf). Association rule mining algorithms [1] work in two steps: (1) generate all frequent itemsets that satisfy min_supp; (2) generate all association rules that satisfy min_conf using the frequent itemsets.

Note that the data mining process may produce thousands of rules, many of which are uninteresting to users. Hence, several applications have called for the use of constrained rule mining [4, 12, 13, 23]. Specifically, in constraint-based mining, which is performed under the guidance of various of constraints provided by the user. The constraints addressed in the prior works include the following: (1) Knowledge type-constraints [20]; (2) Data constraints [4]; (3) Interestingness constraints [13]; and (4) Rule constraints [19, 23]. Such constraints may be expressed as meta-rules (rule templates), as the maximum or minimum number of predicates that can occur in the rule antecedent or consequent, or as relationships among attributes, attribute values, and/or aggregates. Recently, many constraint-based mining works have focused on the use of rule constraints. This form of constraint-based mining allows users of specifying the rules to be mined according to their need, thereby leading to much more useful mining results. According to our observation, these kinds of rule-constraint problems are based on the concept of embedding a variety of *item-constraints* in the mining process.

On the other hand, a time-variant database, as shown in Figure 1, consists of values or events varying with time. Time-variant databases are popular in many applications, such as daily fluctuations of a stock market, traces of a dynamic production process, scientific experiments, medical treatments, weather records, to name a few. In our opinion, the existing model of the constraint-based association rule mining is not able to efficiently handle the time-variant database due to two fundamental problems, i.e., (1) lack of consideration of the *exhibition period* of each individual transaction; (2) lack of an intelligent support counting basis for each item. Note that the traditional mining process treats transactions in different time periods indifferently and handles them along the same procedure. However, since different transactions have different exhibition periods in a time-variant database, only considering the occurrence count of each item might not lead to interesting mining results. This problem can be further explained by the example below.

Example 1.1: In a transaction database as shown in Figure 2, the minimum transaction support and confidence are assumed to be $min\_supp = 30\%$ and

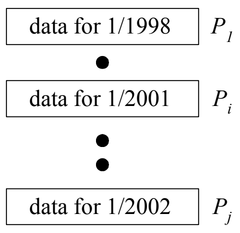
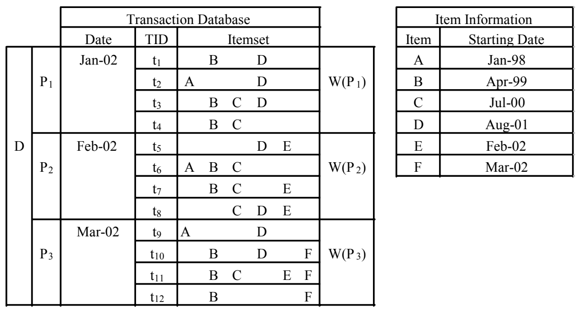


Fig. 1. A time-variant transaction database



		Transaction Database			
		Date	TID	Itemset	
D	P_1	Jan-02	t_1	B D	$W(P_1)$
			t_2	A D	
			t_3	B C D	
			t_4	B C	
	P_2	Feb-02	t_5	D E	$W(P_2)$
			t_6	A B C	
			t_7	B C E	
			t_8	C D E	
	P_3	Mar-02	t_9	A D	$W(P_3)$
			t_{10}	B D F	
			t_{11}	B C E F	
			t_{12}	B F	

Item Information	
Item	Starting Date
A	Jan-98
B	Apr-99
C	Jul-00
D	Aug-01
E	Feb-02
F	Mar-02

Fig. 2. An illustrative transaction database

$min\_conf = 75\%$, respectively. A set of time-variant database indicates the transaction records from January 2002 to March 2002. The starting date of each transaction item is also given. Based on the traditional mining techniques, the *support threshold* is denoted as $min\_S^T = \lceil 12 \times 0.3 \rceil = 4$ where 12 is the size of transaction set $\mathcal{D}$. It can be seen that only *{B, C, D, E, BC}* can be termed as frequent itemsets since their occurrences in this transaction database are all larger than the value of support threshold min_S^T. Thus, rule $C \Rightarrow B$ is termed as a frequent association rule with support $supp(C \cup B) = 41.67\%$ and confidence $conf(C \Rightarrow B) = 83.33\%$. However, it can be observed from Figure 2 that an early product intrinsically possesses a higher likelihood to be determined as a frequent itemset. In addition, different transactions are usually of different importance to the user. This aspect is not well explored by prior works.

Since the early work in [1], several efficient algorithms to mine association rules have been developed. These studies cover a broad spectrum of topics including: (1) fast algorithms based on the level-wise Apriori framework [2, 18], partitioning [16], and FP-growth methods [10]; (2) incremental updating [9, 15]; (3) mining of generalized multi-dimensional [24] and multi-level rules [21]; (4) constraint-based rule mining [13, 23]; and (5) temporal association rule discovery [3, 5, 8, 14]; While these are important results toward enabling the integration of association mining and fast searching algorithms, e.g., BFS and DFS which are classified in [11], we note that these mining methods cannot effectively be applied to the problem of *time-constraint mining* on a *time-variant database* which is of increasing importance. Note that one straightforward approach to addressing the above issues is to employ the item-constraints [13, 23] and/or multiple supports strategies [17, 22], i.e., new coming items have higher weights for their item occurrences. However, as noted in [17, 22] these approaches will encounter another problem, i.e., there is no proper confidence threshold in such cases for the corresponding rule generation.

Consequently, we broaden in this paper the horizon of frequent pattern mining by introducing a weighted model of *transaction-weighted association rules* (abbreviatedly as *weighted association rules*) in a time-variant database.

Specifically, we propose an efficient *Progressive Weighted Miner* (abbreviatedly as *PWM*) algorithm to perform the mining for this problem. In algorithm *PWM*, the importance of each transaction period is first reflected by a proper weight assigned by the user. Then, *PWM* partitions the time-variant database in light of weighted periods of transactions and performs weighted mining. Explicitly, algorithm *PWM* explores the mining of *weighted association rules*, denoted by $(X \Rightarrow Y)^W$, which is produced by two newly defined concepts of $weighted - support$ and $weighted - confidence$ in light of the corresponding weights in individual transactions. Basically, an association rule $X \Rightarrow Y$ is termed to be a frequent weighted association rule $(X \Rightarrow Y)^W$ if and only if its weighted support is larger than minimum support required, i.e., $supp^W(X \cup Y) > min\_supp$, and the weighted confidence $conf^W(X \Rightarrow Y)$ is larger than minimum confidence needed, i.e., $conf^W(X \Rightarrow Y) > min\_conf$. Instead of using the *traditional support threshold* $min\_S^T = \lceil |\mathcal{D}| \times min\_supp \rceil$ as a minimum support threshold for each item, a weighted minimum support, denoted by $min\_S^W = \{\Sigma|P_i| \times W(P_i)\} \times min\_supp$, is employed for the mining of weighted association rules, where $|P_i|$ and $W(P_i)$ represent the amount of partial transactions and their corresponding weight values by a *weighting function* $W(\cdot)$ in the weighted period P_i of the database $\mathcal{D}$. Let $N_{P_i}(X)$ be the number of transactions in partition P_i that contain itemset X. The support value of an itemset X can then be formulated as $S^W(X) = \Sigma N_{P_i}(X) \times W(P_i)$. As a result, the weighted support ratio of an itemset X is $supp^W(X) = \frac{S^W(X)}{\Sigma|P_i| \times W(P_i)}$.

Example 1.2: Let us follow Example 1.1 with the given $min\_supp = 30\%$ and $min\_conf = 75\%$. Consider $W(P_1)$ =0.5, $W(P_2) = 1$, and $W(P_3) = 2$, we have this newly defined support threshold as $min\_S^W = \{4 \times 0.5 + 4 \times 1 + 4 \times 2\} \times 0.3 = 4.2$, we have weighted association rules, i.e., $(C \Rightarrow B)^W$ with relative weighted support $supp^W(C \cup B) = 35.7\%$ and confidence $conf^W(C \Rightarrow B) = \frac{\mathrm{supp}^W(C \bigcup B)}{\mathrm{supp}^W(C)} = 83.3\%$ and $(F \Rightarrow B)^W$ with relative weighted support $supp^W$ $(F \cup B) = 42.8\%$ and confidence $conf^W(F \Rightarrow B) = 100\%$. More details can be found in a complete example in Section 3.2.

Explicitly, *PWM* first partitions the transaction database in light of weighted periods of transactions and then progressively accumulates the occurrence count of each candidate 2-itemset based on the intrinsic partitioning characteristics. With this design, algorithm *PWM* is able to efficiently produce weighted association rules for applications where different time periods are assigned with different weights. Algorithm *PWM* is also designed to employ a filtering threshold in each partition to early prune out those cumulatively infrequent 2-itemsets. The feature that the number of candidate 2-itemsets generated by *PWM* is very close to the actual number of frequent 2-itemsets allows us of employing the scan reduction technique by generating C_ks from C_2 directly to effectively reduce the number of database scans. In fact, the number of the candidate itemsets C_ks generated by *PWM* approaches to its theoretical minimum, i.e., the number of actual frequent k-itemsets, as the value of the minimal support increases. Specifically, the execution time of PWM is, in orders of magnitude, smaller than those required by $Apriori^W$.

Note that those constraint-based rule mining methods that allow users of deriving rules of interest by providing meta-rules and item-constraints [13, 23] are not applicable to solving the weighted mining problem addressed in this paper since the constraints we consider are on individual *transactions* rather than on *items*. Indeed, the problem of mining weighted association rules will be degenerated to the traditional one of mining association rules if the weighting function is assigned to be $W(\cdot) = 1$, meaning that the model we consider can be viewed as a general framework of prior studies. In this paper, we not only explore the new model of weighted association rules in a time-variant database, but also propose an efficient *Progressive Weighted Miner* methodology to perform the mining for this problem. These features distinguish this paper from others.

The rest of this paper is organized as follows. Problem description is given in Section 2. Algorithm PWM is described in Section 3. This paper concludes with Section 4.

2 Problem Description

Let n be the number of partitions with a time granularity, e.g., *business-week, month, quarter, year*, etc., in database $\mathcal{D}$. In the model considered, P_i denotes the part of the transaction database where $P_i \subseteq \mathcal{D}$. Explicitly, we explore in this paper the mining of *transaction-weighted association rules* (abbreviatedly as *weighted association rules*), i.e., $(X \Rightarrow Y)^W$, where $X \Rightarrow Y$ is produced by the concepts of $weighted - support$ and $weighted - confidence$. Further, instead of using the *traditional support threshold* $min\_S^T = \lceil |\mathcal{D}| \times min\_supp \rceil$ as a minimum support threshold for each item in Figure 2, a weighted minimum support for mining an association rules is determined by $min\_S^W = \{\Sigma |P_i| \times W(P_i)\} \times min\_supp$ where $|P_i|$ and $W(P_i)$ represent the amount of partial transactions and their corresponding weight values by a *weighting function* $W(\cdot)$ in the weighted period P_i of the database $\mathcal{D}$. Formally, we have the following definitions.

Definition 1: Let $N_{P_i}(X)$ be the number of transactions in partition P_i that contain itemset X. Consequently, the weighted support value of an itemset X can be formulated as $S^W(X) = \Sigma N_{P_i}(X) \times W(P_i)$. As a result, the weighted support ratio of an itemset X is $supp^W(X) = \frac{S^W(X)}{\Sigma |P_i| \times W(P_i)}$.

In accordance with Definition 1, an itemset X is termed to be frequent when the weighted occurrence frequency of X is larger than the value of min_supp required, i.e., $supp^W(X) > min\_supp$, in transaction set $\mathcal{D}$. The weighted confidence of a weighted association rule $(X \Rightarrow Y)^W$ is then defined below.

Definition 2: $conf^W(X \Rightarrow Y) = \frac{\mathrm{supp}^{\mathrm{W}}(X \bigcup Y)}{\mathrm{supp}^{\mathrm{W}}(X)}$.

Definition 3: An association rule $X \Rightarrow Y$ is termed a frequent weighted association rule $(X \Rightarrow Y)^W$ if and only if its weighted support is larger than minimum support required, i.e., $supp^W(X \cup Y) > min\_supp$, and the weighted confidence $conf^W(X \Rightarrow Y)$ is larger than minimum confidence needed, i.e., $conf^W(X \Rightarrow Y) > min\_conf$.

Example 2.1: Recall the illustrative weighted association rules, e.g., $(F \Rightarrow B)^W$ with relative weighted support $supp^W(F \cup B) = 42.8\%$ and confidence $conf^W(F \Rightarrow B) = 100\%$, in Example 1.2. In accordance with Definition 3, the implication $(F \Rightarrow B)^W$ is termed as a frequent weighted association rule if and only if $supp^W(F \cup B) > min\_supp$ and $conf^W(F \Rightarrow B) > min\_conf$. Consequently, we have to determine if $supp^W(F) > min\_supp$ and $supp^W(FB) > min\_supp$ for discovering the newly identified association rule $(F \Rightarrow B)^W$.

Note that with this weighted association rule $(F \Rightarrow B)^W$, we are able to discover that a new coming product, e.g., a high resolution digital camera, may promote the selling of an existing product, e.g., a color printer. This important information, as pointed out earlier, will not be discovered by the traditional mining schemes, showing the usefulness of this novel model of weighted association rule mining. Once, $\mathcal{F}^{\mathcal{W}} = \{\ X \subseteq \mathcal{I} \mid X \text{ is } frequent\}$, the set of all frequent itemsets together with their support values is known, deriving the desired weighted association rules is straightforward. For every $X \in \mathcal{F}^{\mathcal{W}}$, one can simply check the confidence of all rules $(X \Rightarrow Y)^W$ and drop those whose $\frac{\mathrm{supp}^{\mathrm{W}}(X \bigcup Y)}{\mathrm{supp}^{\mathrm{W}}(X)} < min\_conf$ for rule generation. Therefore, in the rest of this paper we concentrate our discussion on the algorithms for mining frequent itemsets.

It is worth mentioning that there is no restriction imposed on the weighting functions assigned by users. In fact, in addition to the time periods of transactions, other attributes of transactions, such as ownership, transaction length, etc., can also be incorporated into the determination of weights for individual transactions.

3 Progressive Weighted Mining

It is noted that most of the previous studies, including those in [1, 9, 18], belong to Apriori-like approaches. Basically, an Apriori-like approach is based on an anti-monotone Apriori heuristic [1], i.e., if any itemset of length k is not frequent in the database, its length $(k + 1)$ super-itemset will never be frequent. The essential idea is to iteratively generate the set of candidate itemsets of length $(k + 1)$ from the set of frequent itemsets of length k (for $k \geq 1$), and to check their corresponding occurrence frequencies in the database. As a result, if the largest *frequent* itemset is a j-itemset, then an Apriori-like algorithm may need to scan the database up to $(j+1)$ times. This is the basic concept of an extended version of Apriori-based algorithm, referred to as $Apriori^W$.

In [7], the technique of scan-reduction was proposed and shown to result in prominent performance improvement. By scan reduction, C_k is generated from $C_{k-1} \star C_{k-1}$ instead of from $L_{k-1} \star L_{k-1}$. Clearly, a C_3' generated from $C_2 \star C_2$, instead of from $L_2 \star L_2$, will have a size greater than $|C_3|$ where C_3 is generated from $L_2 \star L_2$. However, if $|C'_3|$ is not much larger than $|C_3|$, and both C_2 and C_3 can be stored in main memory, we can find L_2 and L_3 together when the next scan of the database is performed, thereby saving one round of database scan. It can be seen that using this concept, one can determine all L_ks by as few as two scans of the database (i.e., one initial scan to determine L_1 and a final scan to determine all other frequent itemsets), assuming that C'_k for $k \geq 3$ is generated from C'_{k-1} and all C'_k for $k > 2$ can be kept in the memory.

3.1 Algorithm of PWM

In general, databases are too large to be held in main memory. Thus, the data mining techniques applied to very large databases have to be highly scalable for efficient execution. As mentioned above, by partitioning a transaction database into several partitions, algorithm *PWM* is devised to employ a progressive filtering scheme in each partition to deal with the candidate itemset generation and process one partition at a time. For ease of exposition, the processing of a partition is termed a *phase* of processing. Under *PWM*, the cumulative information in the prior phases is selectively carried over toward the generation of candidate itemsets in the subsequent phases. After the processing of a phase, algorithm *PWM* outputs a progressive candidate set of itemsets, their occurrence counts and the corresponding partial supports required.

The procedure of algorithm *PWM* is outlined below, where algorithm *PWM* is decomposed into four sub-procedures for ease of description. C_2 is the set of progressive candidate 2-itemsets generated by database $\mathcal{D}$. Recall that $N_{P_i}(X)$ is the number of transactions in partition P_i that contain itemset X and $W(P_i)$ is the corresponding weight of partition P_i.

Algorithm *PWM (n, min_ supp)*

```
Procedure I:InitialPartition
1. $|D| = \sum_{i=1,n} |P_i|$;

Procedure II: Candidate 2-Itemset Generation
1. begin for $i = 1$ to $n$        // 1st scan of $D$
2.    begin for each 2-itemset $X_2 \in P_i$
3.      if ( $X_2 \notin C_2$ )
4.        $X_2.count = N_{P_i}(X_2) \times W(P_i)$;
5.        $X_2.start = i$;
6.        if ( $X_2.count \geq min\_supp \times |P_i| \times W(P_i)$ )
7.            $C_2 = C_2 \cup X_2$;
8.      if ( $X_2 \in C_2$ )
9.        $X_2.count = X_2.count + N_{P_i}(X_2) \times W(P_i)$;
10.       if ($X_2.count < min\_supp \times \sum_{m=X_2.start,i} (|P_m| \times W(P_m))$)
11.           $C_2 = C_2 - X_2$;
12.   end
13. end

Procedure III: Candidate k-Itemset Generation
1. begin while ($C_k \neq \emptyset$ & $k \geq 2$)
```

```
2.     $C_{k+1} = C_k \star C_k$;
3.     $k = k + 1$;
4. end
```

Procedure IV: Frequent Itemset Generation

```
1. begin for $i = 1$ to $n$
2.     begin for each itemset $X_k \in C_k$
3.         $X_k.count = X_k.count + N_{P_i}(X_k) \times W(P_i)$;
4. end
5. begin for each itemset $X_k \in C_k$
6.     if $(X_k.count \geq min\_supp \times \sum_{m=1,n} (|P_m| \times W(P_m)))$
7.         $L_k = L_k \cup X_k$;
8. end
9. return $L_k$;
```

Procedure II (Candidate 2-Itemset Generation) first scans partition P_i, for $i = 1$ to n, to find the set of all local frequent 2-itemsets in P_i. Note that C_2 is a superset of the set of all frequent 2-itemsets in $\mathcal{D}$. Algorithm *PWM* constructs C_2 incrementally by adding candidate 2-itemsets to C_2 as well as counting the number of occurrences for each candidate 2-itemset X_2 in C_2. If the cumulative occurrences of a candidate 2-itemset X_2 does not meet the partial minimum support, X_2 is removed from the progressive C_2. In *Procedure II (Candidate 2-Itemset Generation)*, algorithm *PWM* processes one partition at a time for all partitions. The number of occurrences of an itemset X_2 and its starting partition which keeps its first occurrence in C_2 are recorded in $X_2.count$ and $X_2.start$, respectively. As such, in the end of processing P_i, an itemset X_2 will be kept in C_2 only if $X_2.count > min\_supp \times \sum_{m=X_2.start,i}(|P_m| \times W(P_m))$. Next, in *Procedure III (Candidate k-Itemset Generation)*, with the scan reduction scheme [18], C_2 produced by the first scan of database is employed to generate C_ks in main memory.

Then, from *Procedure IV (Frequent Itemset Generation)* we begin the second database scan to calculate the support of each itemset in C_k and to find out which candidate itemsets are really frequent itemsets in database $\mathcal{D}$. As a result, those itemsets whose $X_k.count \geq min\_supp \times \sum_{m=1,n}(|P_m| \times W(P_m))$ are the frequent itemsets L_ks. Finally, according to these output L_ks in Step 9, all kinds of weighted association rules implied in database $\mathcal{D}$ can be generated in a straightforward manner.

Note that *PWM* is able to filter out false candidate itemsets in P_i with a hash table. Same as in [18], using a hash table to prune candidate 2-itemsets, i.e., C_2, in each accumulative ongoing partition set P_i of transaction database, the CPU and memory overhead of *PWM* can be further reduced. Owing to the small number of candidate sets generated, the scan reduction technique can be applied efficiently. As a result, only two scans of the database are required.

3.2 An illustrative example of PWM

Recall the transaction database shown in Figure 2 where the transaction database $\mathcal{D}$ is assumed to be segmented into three partitions P_1, P_2 and P_3, which

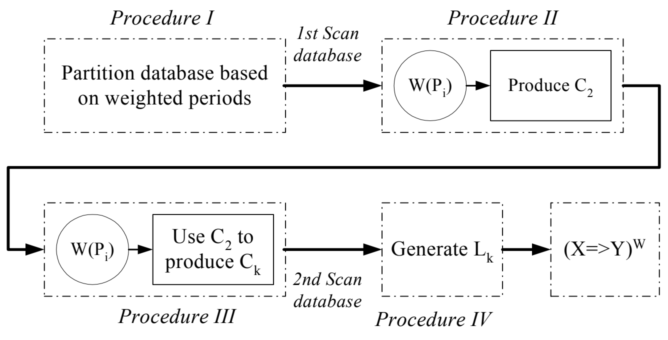


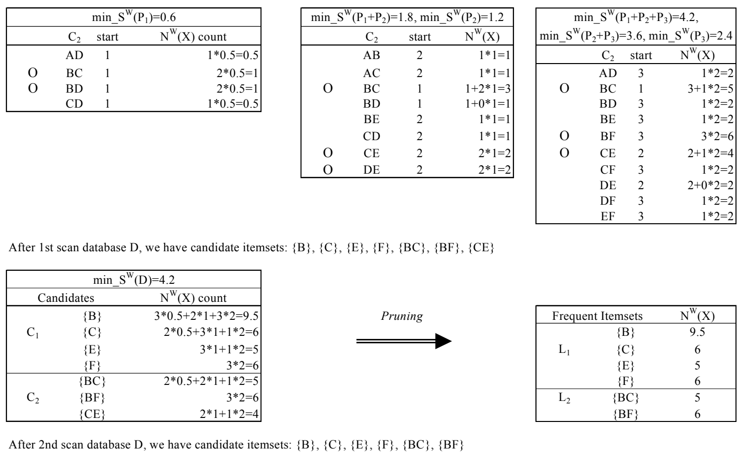
Fig. 3. The flowchart of *PWM*

correspond to the three time granularities from January 2001 to March 2001. Suppose that $min\_supp = 30\%$ and $min\_conf = 75\%$. In addition, the weight value of each partition is given as follows: $W(P_1) = 0.5$, $W(P_2) = 1$, and $W(P_3) = 2$. The operation of algorithm PWM can be best understood by an illustrative example described below. The flowchart of *PWM* is given in Figure 3.

Specifically, each partition is scanned sequentially for the generation of candidate 2-itemsets in the first scan of the database $\mathcal{D}$. After scanning the first segment of 4 transactions, i.e., partition P_1, 2-itemsets $\{BD, BC, CD, AD\}$ are sequentially generated as shown in Figure 4. In addition, each potential candidate itemset $c \in C_2$ has two attributes: (1) *c.start* which contains the partition number of the corresponding starting partition when c was added to C_2, and (2) *c.count* which contains the number of *weighted occurrences* of c. Since there are four transactions in P_1, the partial weighted minimal support is $min\_S^W(P_1) = 4 \times 0.3 \times 0.5 = 0.6$. Such a partial weighted minimal support is called the *filtering threshold*. Itemsets whose occurrence counts are below the filtering threshold are removed. Then, as shown in Figure 4, only $\{BD, BC\}$, marked by "$\bigcirc$", remain as candidate itemsets whose information is then carried over to the next phase P_2 of processing.

Similarly, after scanning partition P_2, the occurrence counts of potential candidate 2-itemsets are recorded. From Figure 4, it is noted that since there are also 4 transactions in P_2, the filtering threshold of those itemsets carried out from the previous phase is $min\_S^W(P_1 + P_2) = 4 \times 0.3 \times 0.5 + 4 \times 0.3 \times 1 = 1.8$ and that of newly identified candidate itemsets is $min\_S^W(P_2) = 4 \times 0.3 \times 1 = 1.2$. It can be seen that we have 3 candidate itemsets in C_2 after the processing of partition P_2.

Finally, partition P_3 is processed by algorithm *PWM*. The resulting candidate 2-itemsets are $C_2 = \{BC, CE, BF\}$ as shown in Figure 4. Note that though appearing in the previous phase P_2, itemset $\{DE\}$ is removed from C_2 once P_3 is taken into account since its occurrence count does not meet the filtering threshold then, i.e., $2 < 3.6$. Consequently, we have 3 candidate 2-itemsets generated by *PWM*. Note that only 3 candidate 2-itemsets are generated by *PWM*.



$min\_S^W(P_1)=0.6$			
	C_2	start	$N^W(X)$ count
	AD	1	1*0.5=0.5
O	BC	1	2*0.5=1
O	BD	1	2*0.5=1
	CD	1	1*0.5=0.5

$min\_S^W(P_1+P_2)=1.8$, $min\_S^W(P_2)=1.2$			
	C_2	start	$N^W(X)$
	AB	2	1*1=1
	AC	2	1*1=1
O	BC	1	1+2*1=3
	BD	1	1+0*1=1
	BE	2	1*1=1
	CD	2	1*1=1
O	CE	2	2*1=2
O	DE	2	2*1=2

$min\_S^W(P_1+P_2+P_3)=4.2$, $min\_S^W(P_2+P_3)=3.6$, $min\_S^W(P_3)=2.4$			
	C_2	start	$N^W(X)$
	AD	3	1*2=2
O	BC	1	3+1*2=5
	BD	3	1*2=2
	BE	3	1*2=2
O	BF	3	3*2=6
O	CE	2	2+1*2=4
	CF	3	1*2=2
	DE	2	2+0*2=2
	DF	3	1*2=2
	EF	3	1*2=2

After 1st scan database D, we have candidate itemsets: {B}, {C}, {E}, {F}, {BC}, {BF}, {CE}

$min\_S^W(D)=4.2$		
Candidates		$N^W(X)$ count
	{B}	3*0.5+2*1+3*2=9.5
C_1	{C}	2*0.5+3*1+1*2=6
	{E}	3*1+1*2=5
	{F}	3*2=6
	{BC}	2*0.5+2*1+1*2=5
C_2	{BF}	3*2=6
	{CE}	2*1+1*2=4

Frequent Itemsets		$N^W(X)$
	{B}	9.5
L_1	{C}	6
	{E}	5
	{F}	6
L_2	{BC}	5
	{BF}	6

After 2nd scan database D, we have candidate itemsets: {B}, {C}, {E}, {F}, {BC}, {BF}

Fig. 4. Frequent itemsets generation for mining weighted association rules by PWM

After generating C_2 from the first scan of database $\mathcal{D}$, we employ the scan reduction technique [18] and use C_2 to generate C_k. As discussed earlier, since the $|C_2|$ generated by PWM is very close to the theoretical minimum, i.e., $|L_2|$, the $|C'_3|$ is not much larger than $|C_3|$. Similarly, the $|C'_k|$ is close to $|C_k|$. Since $C_2 = \{BC, CE, BF\}$, no candidate k-itemset is generated in this example for $k \geq 3$. Thus, $C'_1 = \{B, C, E, F\}$ and $C'_2 = \{BC, CE, BF\}$, where all C'_ks can be stored in main memory. Then, we can find L_ks $(k = 1, 2, ..., m)$ together when the second scan of the database $\mathcal{D}$ is performed. Finally we get $L_2 = \{BC, BF\}$ in Figure 4.

It is important to note that if we adopt single $min\_supp = 30\%$ by Apriori, then the itemset $\{BF\}$ will not be large since its occurrence in this transaction database is 3 which is smaller than $min\_S^T = \lceil 12 \times 0.3 \rceil = 4$. However, itemset $\{BF\}$ appears very frequently in the most recent partition of the database of which the weight is relatively large, thus discovering more desirable information. It can be seen that the algorithm Apriori is not able to discover the information behind the new coming data in the transaction database.

Note that since there is no candidate k-itemset $(k \geq 2)$ containing A or D in this example, A and D are not necessary taken as potential itemsets for generating weighted association rules. In other words, we can skip them from the set of candidate itemsets C'_ks. Finally, all occurrence counts of C'_ks can be calculated by the second database scan. As shown in Figure 5, after all frequent k-itemsets are identified, the corresponding weighted association rules can be derived in a straightforward manner. Explicitly, the weighted association rule of $(X \Rightarrow Y)^W$ holds if $conf^W(X \Rightarrow Y) \geq min\_conf$.

Rules	Support	Confidence
B => C	5/(4*0.5+4*1+4*2)=35.7%	5/9.5=52.6%
B => F	6/(4*0.5+4*1+4*2)=42.8%	6/9.5=63.1%
C => B	5/(4*0.5+4*1+4*2)=35.7%	5/6=83.3%
F => B	6/(4*0.5+4*1+4*2)=42.8%	6/6=100%

Pruning $\Rightarrow$

Rules	Support	Confidence
C => B	35.7%	83.3%
F => B	42.8%	100.0%

Fig. 5. The weighted association rule generation from frequent itemsets

4 Conclusion

In this paper, we explored a new model of mining weighted association rules, i.e., $(X \Rightarrow Y)^W$, in a transaction database and developed algorithm PWM to generate the weighted association rules as well as conducted related performance studies. In algorithm PWM, the importance of each transaction period was first reflected by a proper weight assigned by the user. Then, PWM partitioned the time-variant database in light of weighted periods of transactions and performed weighted mining. Algorithm PWM was designed to progressively accumulate the itemset counts based on the intrinsic partitioning characteristics and employed a filtering threshold in each partition to early prune out those cumulatively infrequent 2-itemsets. With this design, algorithm PWM was able to efficiently produce weighted association rules for applications where different time periods were assigned with different weights, leading to more interesting results.

Acknowledgement

The authors are supported in part by the Ministry of Education Project No. 89E-FA06-2-4-7 and the National Science Council, Project No. NSC 91-2213-E-002-034 and NSC 91-2213-E-002-045, Taiwan, Republic of China.

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. *Proc. of ACM SIGMOD*, pages 207–216, May 1993.
2. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. *Proc. of the 20th International Conference on Very Large Data Bases*, pages 478–499, September 1994.
3. J. M. Ale and G. Rossi. An Approach to Discovering Temporal Association Rules. *ACM Symposium on Applied Computing*, 2000.
4. A. M. Ayad, N. M. El-Makky, and Y. Taha. Incremental mining of constrained association rules. *Proc. of the First SIAM Conference on Data Mining*, 2001.
5. C.-Y. Chang, M.-S. Chen, and C.-H. Lee. Mining General Temporal Association Rules for Items with Different Exhibition Periods. *Proc. of the IEEE 2nd Intern'l Conf. on Data Mining (ICDM-2002)*, December.
6. M.-S. Chen, J. Han, and P. S. Yu. Data Mining: An Overview from Database Perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):866–883, December 1996.

7. M.-S. Chen, J.-S. Park, and P. S. Yu. Efficient Data Mining for Path Traversal Patterns. *IEEE Transactions on Knowledge and Data Engineering*, 10(2):209–221, April 1998.
8. X. Chen and I. Petr. Discovering Temporal Association Rules: Algorithms, Language and System. *Proc. of 2000 Int. Conf. on Data Engineering*, 2000.
9. D. Cheung, J. Han, V. Ng, and C. Y. Wong. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. *Proc. of 1996 Int'l Conf. on Data Engineering*, pages 106–114, February 1996.
10. J. Han and J. Pei. Mining Frequent Patterns by Pattern-Growth: Methodology and Implications. *ACM SIGKDD Explorations (Special Issue on Scaleble Data Mining Algorithms)*, December 2000.
11. J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining – a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, July 2000.
12. D. Kifer, C. Bucila, J. Gehrke, and W. White. Dualminer: A dual-pruning algorithm for itemsets with constraints. *Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.
13. L. V. S. Lakshmanan, R. Ng, J. Han, and A. Pang. Optimization of Constrained Frequent Set Queries with 2-Variable Constraints. *Proc. of 1999 ACM-SIGMOD Conf. on Management of Data*, pages 157–168, June 1999.
14. C.-H. Lee, C.-R. Lin, and M.-S. Chen. On Mining General Temporal Association Rules in a Publication Database. *Proc. of 2001 IEEE International Conference on Data Mining*, November 2001.
15. C.-H. Lee, C.-R. Lin, and M.-S. Chen. Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining. *Proc. of the Tenth ACM Intern'l Conf. on Information and Knowledge Management*, November 2001.
16. J.-L. Lin and M. H. Dunham. Mining Association Rules: Anti-Skew Algorithms. *Proc. of 1998 Int'l Conf. on Data Engineering*, pages 486–493, 1998.
17. B. Liu, W. Hsu, and Y. Ma. Mining Association Rules with Multiple Minimum Supports. *Proc. of 1999 Int. Conf. on Knowledge Discovery and Data Mining*, August 1999.
18. J.-S. Park, M.-S. Chen, and P. S. Yu. Using a Hash-Based Method with Transaction Trimming for Mining Association Rules. *IEEE Transactions on Knowledge and Data Engineering*, 9(5):813–825, October 1997.
19. J. Pei and J. Han. Can We Push More Constraints into Frequent Pattern Mining? *Proc. of 2000 Int. Conf. on Knowledge Discovery and Data Mining*, August 2000.
20. J. Pei, J. Han, and L.V.S. Lakshmanan. Mining Frequent Itemsets with Convertible Constraints. *Proc. of 2001 Int. Conf. on Data Engineering*, 2001.
21. R. Srikant and R. Agrawal. Mining Generalized Association Rules. *Proc. of the 21th International Conference on Very Large Data Bases*, pages 407–419, September 1995.
22. K. Wang, Y. He, and J. Han. Mining Frequent Itemsets Using Support Constraints. *Proc. of 2000 Int. Conf. on Very Large Data Bases*, September 2000.
23. W. Wang, J. Yang, and P. S. Yu. Efficient mining of weighted association rules (WAR). *Proc. of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000.
24. C. Yang, U. Fayyad, and P. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. *Proc. of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.