On Mining General Temporal Association Rules in a Publication Database

Chang-Hung Lee, Cheng-Ru Lin, and Ming-Syan Chen
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, ROC
E-mail: mschen@cc.ee.ntu.edu.tw, {chlee, owenlin}@arbor.ee.ntu.edu.tw

Abstract

In this paper, we explore a new problem of mining general temporal association rules in publication databases. In essence, a publication database is a set of transactions where each transaction T is a set of items of which each item contains an individual exhibition period. The current model of association rule mining is not able to handle the publication database due to the following fundamental problems, i.e., (1) lack of consideration of the exhibition period of each individual item; (2) lack of an equitable support counting basis for each item. To remedy this, we propose an innovative algorithm Progressive-Partition-Miner (abbreviatedly as PPM) to discover general temporal association rules in a publication database. The basic idea of PPM is to first partition the publication database in light of exhibition periods of items and then progressively accumulate the occurrence count of each candidate 2-itemset based on the intrinsic partitioning characteristics. Algorithm PPM is also designed to employ a filtering threshold in each partition to early prune out those cumulatively infrequent 2-itemsets. Explicitly, the execution time of PPM is, in orders of magnitude, smaller than those required by the schemes which are directly extended from existing methods.

1 Introduction

The discovery of association relationship among a huge database has been known to be useful in selective marketing, decision analysis, and business management [4, 9]. 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 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).

Since the early work in [1], several efficient algorithms to mine association rules have been developed in recent years. These studies cover a broad spectrum of topics including: (1) fast algorithms based on the level-wise Apriori framework [2, 13] and partitioning [11]; (2) FP-growth algorithms [8]; (3) incremental updating [6, 10]; (4) mining of generalized and multi-level rules [7, 14]; (5) mining of quantitative rules [15]; (6) mining of multi-dimensional rules [18]; (7) constraint-based rule mining [16] and multiple minimum supports issues [12, 17]; and (8) temporal association rule discovery [3, 5].

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 [9], we note that these mining methods cannot effectively be applied to the mining of a *publication-like* database which is of increasing popularity recently. In essence, a publication database is a set of transactions where each transaction T is a set of items of which each item contains an individual exhibition period. The current model of association rule mining is not able to handle the publication database due to the following fundamental problems, i.e., (1) lack of consideration of the *exhibition period* of each individual item; (2) lack of an equitable support counting basis for each item. Note that the traditional mining process takes the same *task-relevant tuples*, i.e., the size of transaction set D, as a counting basis. Recall that the task of support specification is to specify the minimum transaction support for each itemset. However, since different items have different exhibition periods in a publication database, only considering the occurrence count of each item might not lead to a fair measurement. This problem can be further explained by the illustrative example below.

Example 1.1: In a bookstore transaction database as shown in Figure 1, the minimum transaction support and confidence are assumed to be $min\_supp = 30\%$ and $min\_conf = 75\%$, respectively. A set of time series

0-7695-1119-8/01 $17.00 © 2001 IEEE

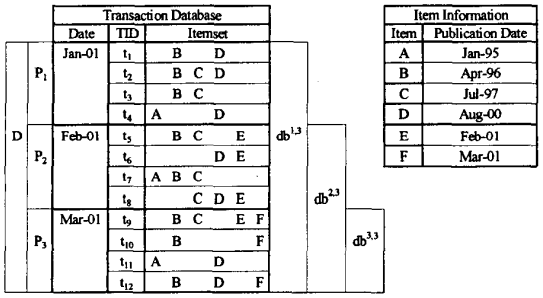

Figure 1. An illustrative transaction database and the corresponding item information

database indicates the transaction records from January 2001 to March 2001. The publication date of each transaction item is also given. Based on the traditional mining techniques, the *absolute support threshold* is denoted as $S^A = \lceil 12 * 0.3 \rceil = 4$ 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 the amounts of their occurrences in this transaction database are respectively larger than the absolute value of support threshold. Thus, only rule $C \Longrightarrow B$ is termed as a frequent association rule with support $s = 41.67\%$ and confidence $c = 83.33\%$. However, some phenomena are observed when we take the "*item information*" in Figure 1 into consideration.

1. **An early publication intrinsically possesses a higher likelihood to be determined as a frequent itemset.** For example, the sales volume of an early product, such as A, B, C or D, is likely to be larger than that of a newly exhibited product, e.g., E or F, since an early product has a longer exhibition period. As a result, the association rules we usually get will be those with long-term products such as "milk and bread are frequently purchased together", which, while being correct by the definition, is of less interest to us in the association rule mining. In contrast, some more recent products, such as new books, which are really "frequent" and interesting in their exhibition periods are less likely to be identified as frequent ones if a traditional mining process is employed.

2. **Some discovered rules may be expired from users' interest.** Considering the generated rule $C \Longrightarrow B$, both B and C were published from the very early dates of this mining transaction database. This information is very likely to have been explored in the previous mining database, such as the one from January 1996 to December 1997. Such mining results could be of less interest to our on-going mining works. For example, most researchers tend to pay more attention to the latest published papers.

Note that one straightforward approach to addressing the above issues is to lower the value of the minimum support threshold required. However, this naive approach will cause another problem, i.e., those interesting rules with smaller supports may be overshadowed by lots of less important information with higher supports. As a consequence, we introduce the notion of *exhibition period* for each transaction item in this paper and develop an algorithm, *Progressive Partition Miner* (abbreviatedly as *PPM*), to address this problem. It is worth mentioning that the application domain of this study is not limited to the mining of a publication database. Other application domains include bookstore transaction databases, video and audio rental store records, stock market data, and transactions in electronic commerce, to name a few.

Explicitly, we explore in this paper the mining of *general temporal association rules*, i.e., $(X \Longrightarrow Y)^{t,n}$, where t is the *latest-exhibition-start time* of both itemsets X and Y, and n denotes the *end* time of the publication database. In other words, (t, n) is the *maximal common exhibition period* of itemsets X and Y. An association rule $X \Longrightarrow Y$ is termed to be a frequent general temporal association rule $(X \Longrightarrow Y)^{t,n}$ if and only if its probability is larger than minimum support required, i.e., $P(X^{t,n} \cup Y^{t,n}) > min\_supp$, and the conditional probability $P(Y^{t,n}|X^{t,n})$ is larger than minimum confidence needed, i.e., $P(Y^{t,n}|X^{t,n}) > min\_conf$. Instead of using the *absolute support threshold* $S^A = \lceil |\mathcal{D}| * min\_supp \rceil$ as a minimum support threshold for each item in Figure 1, a *relative* minimum support, denoted by $S_X^R = \lceil |\mathcal{D}_X| * min\_supp \rceil$ where $|\mathcal{D}_X|$ indicates the amount of partial transactions in the exhibition period of itemset X, is given to deal with the mining of temporal association rules.

To deal with the mining of general temporal association rule $(X \Longrightarrow Y)^{t,n}$, an efficient algorithm, *Progressive Partition Miner*, is devised. The basic idea of *PPM* is to first partition the publication database in light of exhibition periods of items and then progressively accumulate the occurrence count of each candidate 2-itemset based on the intrinsic partitioning characteristics. Algorithm *PPM* 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 *PPM* is very close to the number of frequent 2-itemsets allows us to employ the scan reduction technique by generating C_ks from C_2 directly to effectively reduce the number of database scan. Experimental results show that *PPM* produces a significantly smaller amount of candidate 2-itemsets than *Apriori*$^+$, i.e., an extended version of *Apri-*

ori algorithm. In fact, the number of the candidate itemsets C_ks generated by *PPM* approaches to its theoretical minimum, i.e., the number of frequent k-itemsets, as the value of the minimal support increases. Explicitly, the execution time of *PPM* is, in orders of magnitude, smaller than those required by *Apriori*$^+$. Sensitivity analysis on various parameters of the database is also conducted to provide many insights into algorithm *PPM*. The advantage of *PPM* over *Apriori*$^+$ becomes even more prominent as the size of the database increases. This is indeed an important feature for *PPM* to be practically used for the mining of a time series database in the real world.

It is worth mentioning that the problem of mining *general* temporal association rules will be degenerated to the one of mining temporal association rules explored in prior works [3, 5] if the exhibition period (t, n) of association rule $(X \Longrightarrow Y)^{t,n}$ is applied to a non-maximal exhibition period of $X \Longrightarrow Y$, such as (j, n) where $j > t$. Consider for example the database in Figure 1 where $(C \Longrightarrow B)^{1,3}$ and $(C \Longrightarrow E)^{2,3}$ are two *general temporal* association rules in database $\mathcal{D}$ while the temporal subset of $(C \Longrightarrow B)^{1,3}$, e.g., $(C \Longrightarrow B)^{2,3}$, can also be a temporal association rule as defined in [3, 5], showing that the model we consider can be viewed as a general framework of prior studies. This is the reason we use the term "general temporal association rule" in this paper.

We mention in passing that the Frequent Pattern growth (FP-growth), which constructs a highly compact data structure (an FP-tree) to compress the original transaction database, is a method of mining frequent itemsets without candidate generation [8]. However, in our opinion, FP-growth algorithms do not have obvious extensions to deal with this publication database problem. Further, some methodologies were proposed to explore the problem of discovering temporal association relationship in the partial of database retrieved [3, 5], i.e., to determine association rules from a given subset of database specified by time. These works, however, do not consider the individual exhibition period of each transaction item, and are thus not applicable to solving the mining problems in a publication database. On the other hand, some techniques were devised to use multiple minimum supports for frequent itemsets generation [12, 17]. However, it remains unclear for how the techniques in [12, 17] to be coupled with the corresponding minimum confidence thresholds when general temporal association rules we consider in this paper in a publication database are being generated.

The rest of this paper is organized as follows. Problem description is given in Section 2. Algorithm *PPM* is described in Section 3. Performance studies on various schemes are conducted in Section 4. This paper concludes with Section 5.

2 Problem Description

Let n be the number of partitions with a time granularity, e.g., *business-week, month, quarter, year*, to name a few, in database $\mathcal{D}$. In the model considered, $db^{t,n}$ denotes the part of the transaction database formed by a continuous region from partition P_t to partition P_n, and $|db^{t,n}| = \sum_{h=t,n} |P_h|$ where $db^{t,n} \subseteq \mathcal{D}$. An item $x^{x.start,n}$ is termed as a *temporal item* of x, meaning that $P_{x.start}$ is the starting partition of x and n is the partition number of the last database partition retrieved.

Example 2.1: Consider the database in Figure 1. Since database $\mathcal{D}$ records the transaction data from January 2001 to March 2001, database $\mathcal{D}$ is intrinsically segmented into three partitions P_1, P_2 and P_3 in accordance with the "month" granularity. As a consequence, a partial database $db^{2,3} \subseteq \mathcal{D}$ consists of partitions P_2 and P_3. A temporal item $E^{2,3}$ denotes that the exhibition period of $E^{2,3}$ is from the beginning time of partition P_2 to the end time of partition P_3.

As such, we can define a maximal temporal itemset $X^{t,n}$ as follows.

Definition 1: An itemset $X^{t,n}$ is called a *maximal temporal itemset* in a partial database $db^{t,n}$ if t is the *latest starting partition number* of all items belonging to X in database $\mathcal{D}$ and n is the partition number of the last partition in $db^{t,n}$ retrieved.

For example, as shown in Figure 1, itemset $DE^{2,3}$ is deemed a maximal temporal itemset whereas $CD^{2,3}$ is not. In view of this, the exhibition period of an itemset is expressed in terms of *Maximal Common exhibition Period* (MCP) of the items that appear in the itemset. Let $MCP(x)$ denote the MCP value of item x. The MCP value of an itemset X is the shortest MCP among the items in itemset X. Consider three items C, E and F in Figure 1 for example. Their exhibition periods are as follows: $MCP(C) = (1, 3)$, $MCP(E) = (2, 3)$ and $MCP(F) = (3, 3)$. Since itemset CEF is termed to be $CEF^{3,n} = (CEF)^{3,n}$ with considering the exhibition of CEF, we have $MCP(CEF) = (3, 3)$.

In addition, $|db^{t,n}|$ is the number of transactions in the partial database $db^{t,n}$. The fraction of transaction T supporting an itemset X with respect to partial database $db^{t,n}$ is called the support of $X^{t,n}$, i.e., $supp(X^{MCP(X)}) = \frac{|\{T \in db^{MCP(X)} | X \subseteq T\}|}{|db^{MCP(X)}|}$. The support of a rule $(X \Longrightarrow Y)^{MCP(XY)}$ is defined as $supp((X \Longrightarrow Y)^{MCP(XY)}) = supp((X \bigcup Y)^{MCP(XY)})$. The confidence of this rule is defined as $conf((X \Longrightarrow Y)^{MCP(XY)}) = \frac{supp((X \bigcup Y)^{MCP(XY)})}{supp(X^{MCP(XY)})}$. Consequently, a general temporal association rule $(X \Longrightarrow Y)^{MCP(XY)}$

which holds in the transaction set D can be defined as follows.

Definition 2: An association rule $(X \Longrightarrow Y)^{MCP(XY)}$ is called a *general temporal association rule* in the transaction set D with $conf((X \Longrightarrow Y)^{MCP(XY)}) = c$ and $supp((X \Longrightarrow Y)^{MCP(XY)}) = s$ if $c\%$ of transactions in $db^{MCP(XY)}$ that contain X also contain Y and $s\%$ of transactions in $db^{MCP(XY)}$ contain $X \bigcup Y$ while $X \bigcap Y = \phi$.

For a given pair of *min_conf* and *min_supp* as the minimum thresholds required in the maximal common exhibition period of each association rule, the problem of mining general temporal association rules is to determine all frequent general temporal association rules, e.g., $(X \Longrightarrow Y)^{MCP(XY)} \in db^{MCP(XY)}$ which transaction itemsets X and Y have "relative" support and confidence greater than the corresponding thresholds. Thus, we have the following definition to identify the frequent general temporal association rules.

Definition 3: A general temporal association rule $(X \Longrightarrow Y)^{MCP(XY)}$ is termed to be *frequent* if and only if $supp((X \Longrightarrow Y)^{MCP(XY)}) > min\_supp$ and $conf((X \Longrightarrow Y)^{MCP(XY)}) > min\_conf$.

Consequently, this rule mining of general temporal association can also be decomposed into to three steps:
(1) generate all frequent maximal temporal itemsets (TIs) with their support values.
(2) generate the support values of all corresponding temporal sub-itemsets (SIs) of frequent TIs.
(3) generate all temporal association rules that satisfy min_conf using the frequent TIs and/or SIs.

Example 2.2: Recall the illustrative general temporal association rules, e.g., $(C \Longrightarrow E)^{2,3}$ with relative support 37.5% and confidence 75%, of the bookstore transaction database as shown in the Figure 1. In accordance with Definition 3, the implication $(C \Longrightarrow E)^{2,3}$ is termed as a general temporal association rule if and only if $supp((C \Longrightarrow E)^{2,3}) > min\_supp$ and $conf((C \Longrightarrow E)^{2,3}) > min\_conf$. Consequently, we have to determine if $supp(CE^{2,3}) > min\_supp$ and $supp(C^{2,3}) > min\_supp$ for discovering the newly identified association rule $(C \Longrightarrow E)^{2,3}$. It is worth mentioning that though $CE^{2,3}$ has to be a maximal temporal itemset, called TI, $C^{2,3}$ may not be a TI. We call $C^{2,3}$ is one of corresponding temporal sub-itemsets, i.e., SI, of itemset $CE^{2,3}$.

For better readability, a list of symbols used is given in Table 1. Then, the definition of a *frequent* maximal temporal itemset and the property of its corresponding sub-itemsets are given below.

Definition 4: A maximal temporal itemset $X^{MCP(X)}$ is termed to be frequent when the occurrence frequency of $X^{MCP(X)}$ is larger than the value of min_supp required, i.e., $supp(X^{MCP(X)}) > min\_supp$, in transaction set $db^{MCP(X)}$.

Property 1: When a maximal temporal k-itemset $X_k^{MCP(X_k)}$ is frequent in data set $db^{MCP(X_k)}$, each of its corresponding sub-itemset $X_i^{MCP(X_k)}$ $(1 \leq i < k)$ is also frequent in $db^{MCP(X_k)}$.

$\|db^{i,n}\|$	Number of transactions in $db^{i,n}$
$X^{i,n}$	A temporal itemset in partial database $db^{i,n}$
TI	A maximal temporal itemset
SI	A corresponding temporal sub-itemset of TI

Table 1: Meanings of symbols used

Once, $\mathcal{F} = \{ X^{MCP(X)} \subseteq \mathcal{I} \mid X^{MCP(X)}$ is $frequent\}$, the set of all frequent TIs and SIs together with their support values is known, deriving the desired association rules is straightforward. For every $X^{MCP(X)} \in \mathcal{F}$, check the confidence of all rules $(X \Longrightarrow Y)^{MCP(XY)}$ and drop those that do not satisfy $s(XY^{MCP(XY)})/s(X^{MCP(XY)}) \geq min\_conf$. This problem can also be reduced to the problem of finding all frequent maximal temporal itemsets first and then generating their corresponding frequent sub-itemsets for the same support threshold. Therefore, in the rest of this paper we concentrate our discussion on the algorithms for mining frequent TIs and SIs. In fact, the process steps of generating frequent TIs and SIs can be further merged to one step in our proposed algorithm PPM.

As explained, we have to find all maximal temporal itemsets that satisfy min_supp first and then to calculate the occurrences of their corresponding sub-itemsets for producing all temporal association rules hidden in database $\mathcal{D}$. However, if we use an existing algorithm to find all frequent TIs for this new problem, the downward closure property, which Apriori-based algorithms are based on, no longer holds. In addition, the candidate generation process is not intuitive at all. Note that, even though itemset $X^{t,n}$ is not a frequent itemset, it does not imply that $X^{t+1,n}$, i.e., a temporal sub-itemset of $X^{t,n}$, is not a frequent itemset. In other words, even knowing $X^{t,n}$ is not frequent in $db^{t,n}$ where $MCP(X) = (t, n)$, we are not able to assert whether $XY^{t+1,n}$ is frequent or not when $MCP(Y) = (t + 1, n)$. Specifically, to determine whether a general temporal association rule $(X \Longrightarrow Y)^{t+1,n}$ is frequent, we have to find out the support values of $X^{t+1,n}$ and $XY^{t+1,n}$ where $MCP(XY) = MCP(Y) = (t + 1, n)$.

Example 2.3: Consider $MCP(x_1) = (1, n)$, $MCP(x_2) = (2, n)$ and $MCP(x_3) = (3, n)$. If we find that item x_1 is not frequent at exhibition period $(1, n)$, then it does not satisfy min_supp requirement at level 1. Under a conventional

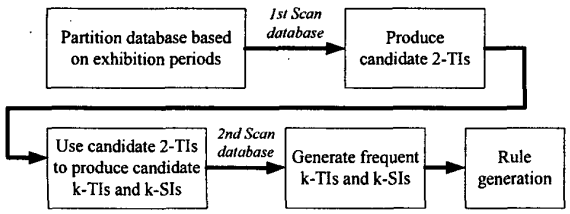

Figure 2. The flowchart of PPM

Apriori-based association rule mining algorithm, this itemset is discarded since it will not be frequent. The potentially frequent itemsets x_1x_2 and x_1x_3 will then not be generated at level 2 for consideration. Clearly, this disposition is incorrect in mining general temporal association rules since x_1 is still possible to be frequent at $(2, n)$ and $(3, n)$, indicating that the downward property is not valid in mining general temporal association rules.

It is worth mentioning that since the downward level-wise property, which holds for Apriori-like algorithms, is not valid in this general temporal association rule mining problem, the second method is to expand each transaction item to be its combination with different exhibition periods. For instance, all temporal sub-itemsets of $X_k^{t,n}$ at level k with different exhibition periods, i.e., $X_k^{t,n}$, $X_k^{t+1,n}$, $X_k^{t+2,n}$, ..., $X_k^{n,n}$, are taken as "*temporal candidate k-itemsets*" for producing any possible combination of general temporal association rules. Using this approach, the problem of mining temporal association rules can be implemented on an anti-monotone Apriori-like heuristic. As in most previous works, the essential idea is to iteratively generate the set of candidate itemsets of length $(k+1)$, i.e., $X_{k+1}^{r,n}$, from the set of frequent itemsets of length k, i.e., $X_k^{r,n}$, (for $k \geq 1$), and to check their corresponding occurrence frequencies in the database $db^{r,n}$. This is the basic concept of an extended version of Apriori-based algorithm, called *Apriori+*, whose performance will be comparatively evaluated with algorithm *PPM* in our experimental studies later.

3 General Temporal Association Rules

An overview of progressive partition miner is given in Section 3.1. We present an illustrative example of algorithm *PPM* in Section 3.2.

3.1 An overview of Progressive Partition Miner

As explained above, a naive adoption of conventional methods to mine general temporal association rules will be prohibitively expensive. To remedy this, by partitioning a transaction database into several partitions, algorithm *PPM* is devised to employ a filtering threshold 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. Explicitly, a progressive candidate set of itemsets is composed of the following two types of candidate itemsets, i.e., (1) the candidate itemsets that were carried over from the previous progressive candidate set in the previous phase and remain as candidate itemsets after the current partition is included into consideration (Such candidate itemsets are called type α candidate itemsets); and (2) the candidate itemsets that were not in the progressive candidate set in the previous phase but are newly selected after only taking the current data partition into account (Such candidate itemsets are called type β candidate itemsets). Under *PPM*, 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 *PPM* outputs a *progressive screen*, denoted by PS, which consists of a progressive candidate set of itemsets, their occurrence counts and the corresponding partial supports required.

3.2 Algorithm of PPM

The operation of algorithm PPM can be best understood by an illustrative example described below and its corresponding flowchart is depicted in Figure 2. Recall the transaction database shown in Figure 1 where the transaction database $db^{1,3}$ is assumed to be segmented into three partitions P_1, P_2 and P_3, which correspond to the three time granularities from January 2001 to March 2001. Suppose that $min\_supp = 30\%$ and $min\_conf = 75\%$. Each partition is scanned sequentially for the generation of candidate 2-itemsets in the first scan of the database $db^{1,3}$. 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 3. 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 occurrences of c since c was added to C_2. Since there are four transactions in P_1, the partial minimal support is $\lceil 4 * 0.3 \rceil = 2$. Such a partial minimal support is called the *filtering threshold* in this paper. Itemsets whose occurrence counts are below the filtering threshold are removed. Then, as shown in Figure 3, only $\{BD, BC\}$, marked by " ◯ ", remain as candidate itemsets (of type β in this phase since they are newly generated) whose information is then carried over to the next phase P_2 of processing.

Similarly, after scanning partition P_2, the occurrence

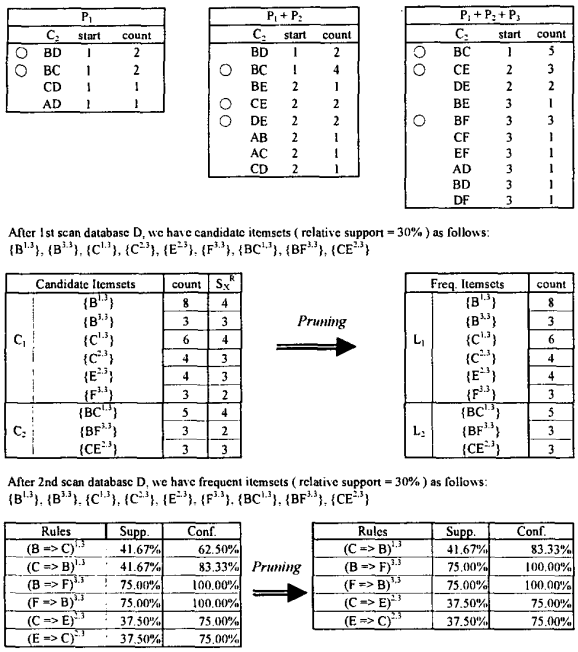

Figure 3. Frequent temporal itemsets generation for mining general temporal association rules by *PPM*

counts of potential candidate 2-itemsets are recorded (of type α and type β). From Figure 3, 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 (that become type α candidate itemsets in this phase) is $\lceil (4+4) * 0.3 \rceil = 3$ and that of newly identified candidate itemsets (i.e., type β candidate itemsets) is $\lceil 4 * 0.3 \rceil = 2$. It can be seen that we have 3 candidate itemsets in C_2 after the processing of partition P_2, and one of them is of type α and two of them are of type β.

Finally, partition P_3 is processed by algorithm *PPM*. The resulting candidate 2-itemsets are $C_2 = \{BC, CE, BF\}$ as shown in Figure 3. 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., $2 < 3$. However, we do have one new itemset, i.e., BF, which joins the C_2 as a type β candidate itemset. Consequently, we have 3 candidate 2-itemsets generated by *PPM*, and two of them are of type α and one of them is of type β. Note that only 3 candidate 2-itemsets are generated by *PPM*.

After generating C_2 from the first scan of database $db^{1,3}$, we employ the scan reduction technique [13] and use C_2 to generate C_k ($k = 2, 3, ..., m$), where C_m is the candidate *last*-itemsets. Instead of generating C_3 from $L_2 \star L_2$, a C_2 generated by *PPM* can be used to generate the candidate 3-itemsets and its sequential C'_{k-1} can be utilized to generate C'_k. 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, since the $|C_2|$ generated by *PPM* 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 where $k \geq 3$. Thus, $C'_k = \{BC, CE, BF\}$ and all C'_k can be stored in main memory. Then, we can find L_ks ($k = 1, 2, ..., m$) together when the second scan of the database $db^{1,3}$ is performed. Note that those generated itemsets $C'_k = \{BC, CE, BF\}$ are termed to be the candidate maximal temporal itemsets (TIs), i.e., $BC^{1,3}, CE^{2,3}$ and $BF^{3,3}$, with a maximal exhibition period of each candidate.

Before we process the second scan of the database $db^{1,3}$ to generate L_ks, all candidate SIs of candidate TIs can be propagated based on Property 1, and then added into C'_k. For instance, as shown in Figure 3, both candidate 1-itemsets $B^{1,3}$ and $C^{1,3}$ are derived from $BC^{1,3}$. Moreover, since $BC^{1,3}$, for example, is a candidate 2-itemset, its subsets, i.e., $B^{1,3}$ and $C^{1,3}$, should potentially be candidate itemsets. As a result, 9 candidate itemsets, i.e., $\{B^{1,3}, B^{3,3}, C^{1,3}, C^{2,3}, E^{2,3}, F^{3,3}, BC^{1,3}, BF^{3,3}, CE^{2,3}\}$ as shown in Figure 3, are generated. Note that since there is no candidate TI k-itemset ($k \geq 2$) containing A or D in this example, $A^{i,3}$ and $D^{i,3}$ ($1 \leq i \leq 3$) are not necessary to be taken as SI itemsets for generating general temporal 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. Note that itemsets $BC^{1,3}$, $BF^{3,3}$ and $CE^{2,3}$ are termed as frequent TIs, while $B^{1,3}$, $B^{3,3}$, $C^{1,3}$, $C^{2,3}$, $E^{2,3}$ and $F^{3,3}$ are frequent SIs in this example.

As shown in Figure 3, after all frequent TI and SI itemsets are identified, the corresponding general temporal association rules can be derived in a straightforward manner. Explicitly, the general temporal association rule of $(X \Rightarrow Y)^{MCP(XY)}$ holds if $conf((X \Rightarrow Y)^{MCP(XY)}) \geq min\_conf$.

4 Experimental Studies

To assess the performance of algorithm *PPM*, we performed several experiments on a computer with a CPU clock rate of 450 MHz and 512 MB of main memory. The methods used to generate synthetic data are described in Section 4.1. The performance comparison of *PPM* and *Apriori*$^+$ is presented in Section 4.2. Results on scaleup experiments are presented in Section 4.3.

4.1 Generation of synthetic workload

For obtaining reliable experimental results, the method to generate synthetic transactions we employed in this study is similar to the ones used in prior works [2, 13]. These transactions mimic the publication items in a publication database. Each database consists of $|D|$ transactions, and on the average, each transaction has $|T|$ items. To simulate the characteristic of the exhibition period in each item, transaction items are uniformly distributed into database $\mathcal{D}$ with a random selection. In accordance with the exhibition periods of items, database $\mathcal{D}$ is divided into n partitions. Table 2 summarizes the meanings of various parameters used in the experiments. The mean of the correlation level is set to 0.25 for our experiments. Without loss of generality, we use the notation $Tx - Iy - Dm$ to represent a database in which $D = m$ thousands, $|T| = x$, and $|I| = y$. We compare relative performance of *Apriori*$^+$ and *PPM*.

$\|\mathcal{D}\|$	Number of transactions in the database
$\|T\|$	Average size of the transactions
$\|I\|$	Average size of the maximal frequent itemsets
$\|L\|$	Number of maximal potentially frequent itemsets
N	Number of items
$\|P_i\|$	Number of transactions in the partition database P_i

Table 2: Meanings of various parameters

4.2 Relative performance

We first conducted several experiments to evaluate the relative performance of *Apriori*$^+$ and *PPM*. Since the experimental results are consistent for various values of n, $|L|$ and N, for interest of space, we only report the results on $|L| = 2000$ and $N = 10000$ in the following experiments. Figure 4 shows the relative execution times for both two algorithms as the minimum support threshold is decreased from 1% support to 0.1% support. When the support threshold is high, there are only a limited number of frequent itemsets produced. However, as the support threshold decreases, the performance difference becomes prominent in that *PPM* significantly outperforms *Apriori*$^+$. Explicitly, *PPM* is in orders of magnitude faster than *Apriori*$^+$, and the margin grows as the minimum support threshold decreases.

4.3 Scaleup performance

In this experiment, we examine the scaleup performance of algorithm *PPM*. The scale-up results for different selected datasets are obtained. Figure 5 shows the scaleup performance of algorithm *PPM* as the values of |D| increase.

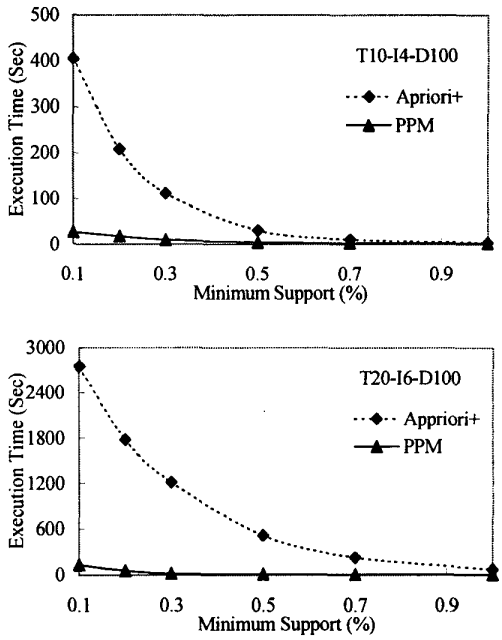

Figure 4. Relative performance studies

Three different minimum supports are considered. We obtained the results for the dataset $T10 - I4 - Dm$ when the number of customers increases from 100,000 to one million. The execution times are normalized with respect to the times for the 100,000 transactions dataset in the Figure 5. Note that, as shown in Figure 5 the execution time only slightly increases with the growth of the database size, showing good scalability of *PPM*.

5 Conclusion

In this paper, we not only explored a new model of mining general temporal association rules, i.e., $(X \Rightarrow Y)^{MCP(XY)}$, in a publication database but also developed algorithm *PPM* to generate the temporal association rules as well as conducted related performance studies. Under *PPM*, the cumulative information of mining previous partitions is selectively carried over toward the generation of candidate itemsets for the subsequent partitions. Algorithm *PPM* is particularly powerful for efficient mining for a publication-like transaction database, such as bookstore transaction databases, video rental store records, library-book rental records, and transactions in electronic commerce. One extension to our proposed model in this paper is to mine general temporal association rules with different

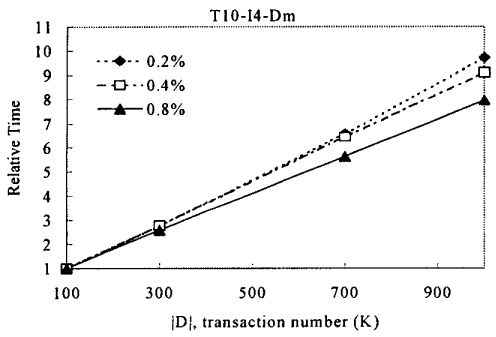

Figure 5. Scaleup performance of PPM

start and end points of items. This is an interesting yet challenging issue since the levelwise property does not hold in this situation, and will be a matter of future research.

6 Acknowledgment

The authors are supported in part by the Ministry of Education Project No. 89-E-FA06-2-4-7 and the National Science Council, Project No. NSC 89-2219-E-002-028 and NSC 89-2218-E-002-028, 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] 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.

[5] X. Chen and I. Petr. Discovering Temporal Association Rules: Algorithms, Language and System. *Proc. of 2000 Int. Conf. on Data Engineering*, 2000.

[6] 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.

[7] J. Han and Y. Fu. Discovery of Multiple-Level Association Rules from Large Databases. *Proc. of the 21th International Conference on Very Large Data Bases*, pages 420–431, September 1995.

[8] J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. *Proc. of 2000 ACM-SIGMOD Int. Conf. on Management of Data*, pages 486–493, May 2000.

[9] 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.

[10] C.-H. Lee, C.-R. Lin, and M.-S. Chen. Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining. *Proc. of the ACM 10th Intern'l Conf. on Information and Knowledge Management*, November 2001.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. *Proc. of 1996 ACM-SIGMOD Conf. on Management of Data*, 1996.

[16] A. K. H. Tung, J. Han, L. V. S. Lakshmanan, and R. T. Ng. Constraint-Based Clustering in Large Databases. *Proc. of 2001 Int. Conf. on Database Theory*, January 2001.

[17] 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.

[18] C. Yang, U. Fayyad, and P. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. *The Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.