Progressive Partition Miner: An Efficient Algorithm for Mining General Temporal Association Rules

Chang-Hung Lee, Ming-Syan Chen, *Senior Member*, *IEEE*, and Cheng-Ru Lin

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 and 2) lack of an equitable support counting basis for each item. To remedy this, we propose an innovative algorithm *Progressive-Partition-Miner* (abbreviated 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. 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 to effectively reduce the number of database scans. Explicitly, the execution time of *PPM* is, in orders of magnitude, smaller than those required by other competitive schemes that are directly extended from existing methods. The correctness of *PPM* is proven and some of its theoretical properties are derived. Sensitivity analysis of various parameters is conducted to provide many insights into Algorithm *PPM*.

Index Terms—Data mining, general temporal association rule, exhibition period, publication database.

1 Introduction

The discovery of association relationships among a huge database has been known to be useful in selective marketing, decision analysis, and business management [7], [16]. 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). Let $\mathcal{I} = \{x_1, x_2, \ldots, x_m\}$ be a set of items. A set $X \subseteq \mathcal{I}$ with $k = |X|$ is called a k-itemset or simply an itemset. Let a database D be a set of transactions, where each transaction T is a set of items such that $T \subseteq \mathcal{I}$. A transaction T is said to support X if and only if $X \subseteq T$. Conventionally, an association rule is an implication of the form $X \Longrightarrow Y$, meaning that the presence of the set X implies the presence of another set Y, where $X \subset \mathcal{I}$, $Y \subset \mathcal{I}$, and $X \bigcap Y = \phi$. The rule $X \Longrightarrow Y$ holds in the transaction set D with $confidence$ c if $c\%$ of transactions in D that contain X also contain Y. The rule $X \Longrightarrow Y$ has $supports$ in the transaction set D if $s\%$ of transactions in D contain $X \bigcup Y$.

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 [2] work in two steps: 1) generate all frequent itemsets that satisfy min_supp and 2) generate all association rules that satisfy min_conf using the frequent itemsets. This problem can be reduced to the problem of finding all frequent itemsets for the same support threshold.

Since the early work in [2], 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 [4], [24], partitioning [20], [27], and sampling [31];
2. TreeProjection [1] and FP-growth algorithms [15];
3. incremental updating [10], [18] and parallel algorithms [3], [23];
4. mining of generalized and multilevel rules [14], [28];
5. mining of quantitative rules [29];
6. mining of multidimensional rules [34];
7. constraint-based rule mining [32] and multiple minimum supports issues [21];
8. associations among correlated or infrequent items [12]; and
9. temporal database discovery [5], [6], [9], [8], [22], [30].

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 [16], 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

• *The authors are with the 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.

Manuscript received 25 Sept. 2001; revised 23 July 2002; accepted 5 Aug. 2002.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 115072.

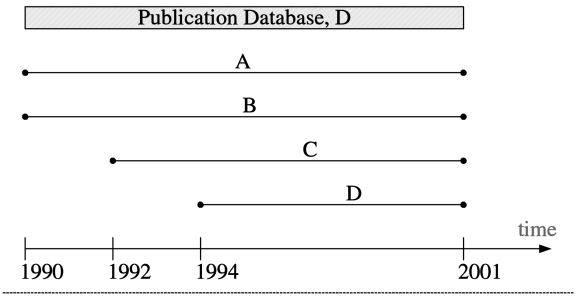


Fig. 1. An illustrative publication database where different items may have different publication dates.

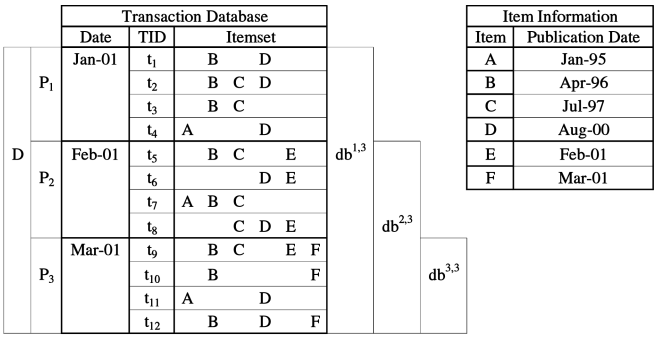


Fig. 2. An illustrative transaction database and the corresponding item information.

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 and 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 two illustrative examples below.

Example 1.1. Consider an illustrative database of publications in a bookstore, as shown in Fig. 1. Note that items A and B are exhibited from 1990 to 2001. However, item C is exhibited from 1992 to 2001 and item D is from 1994 to 2001. Even though each transaction item has a unique exhibition period, conventional mining algorithms, without further provision, tend to ignore such differences and determine frequent association rules with the same counting basis $\mathcal{D}$.

Example 1.2. In a bookstore transaction database, as shown in Fig. 2, the minimum transaction support and confidence are assumed to be $min\_supp = 30\%$ and $min\_conf = 75\%$, respectively. A set of time series 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, and 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 Fig. 2 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* (abbreviated 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 Fig. 2, a *relative* minimum support, denoted by $S^R_X = \lceil |\mathcal{D}_X| * min\_supp \rceil$, 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.

Example 1.3. Let us follow Example 1.2 and the given minimum support and confidence thresholds. According to this newly identified support threshold S_X^R, we have temporal association rules as follows:

1. $(C \Longrightarrow E)^{2,3}$ with relative support 37.5 percent and confidence 75 percent,
2. $(E \Longrightarrow C)^{2,3}$ with relative support 37.5 percent and confidence 75 percent,
3. $(B \Longrightarrow F)^{3,3}$ with relative support 75 percent and confidence 100 percent, and
4. $(F \Longrightarrow B)^{3,3}$ with relative support 75 percent and confidence 100 percent.

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 scans. Experimental results show that *PPM* produces a significantly smaller amount of candidate 2-itemsets than $Apriori^+$, i.e., an extended version of the Apriori 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 [5], [6], [8], [9], [30] if the exhibition period (t, n) of association rule $(X \Longrightarrow Y)^{t,n}$ is applied to a nonmaximal exhibition period of $X \Longrightarrow Y$, such as (j, n), where $j > t$. Consider for example the database in Fig. 2, 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 before [5], [6], [8], [9], [30], showing that the model we consider can be viewed as a general framework of prior studies. This is the very reason we use the term "general temporal association rule" in this paper.

We mention in passing that the works in [20], [27] are essentially based on a partition-based heuristic, i.e., *if X is a frequent itemset in database D, which is divided into n partitions $p_1, p_2, \ldots, p_n$, then X must be a frequent itemset in at least one of the n partitions*. However, these works were not applicable to handling the exhibition period of transaction items on mining association rules. In addition, 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 [15]. However, in our opinion, FP-growth algorithms do not have obvious extensions to deal with this publication database problem, nor do those constraint-based rule mining methods that allow users to focus the search for rules by providing metarules [32]. Further, some methodologies were proposed to explore the problem of discovering temporal association relationships in the partial of database retrieved [5], [6], [8], [9], [11], [13], [17], [19], [26], [30], [33], 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. It is worth mentioning that, in this paper, we assume each item has the same cut-off date of the item exhibition period, i.e., the "n" of (t, n). This is different from the prior definition of "life span" in temporal association rule mining works [5] which may have different ending times of item exhibition periods. As will be seen later, the problem formulation with the same ending period enables us to derive very efficient and effective mining algorithms for temporal association rules.

On the other hand, some techniques were devised to use multiple minimum supports for frequent itemsets generation [21]. However, it remains an open issue for how these techniques 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. In this paper, we not only explore the new model of general temporal association rules in a publication database, but also propose an efficient *Progressive Partition Miner* methodology to perform the mining for this problem as well as conduct the corresponding performance studies. These features distinguish this paper from others.

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

2 Problem Description

To facilitate our presentation, some definitions and symbols used are presented in Section 2.1. For further looking into the proposed problem of mining temporal association rules, the traversing of the search space is examined in Section 2.2. In addition, to assess the performance of *PPM*, we also present in Section 2.2 the concept of an extended version of the Apriori algorithm, called $Apriori^+$, which will be employed in Section 4 later for performance comparison.

2.1 Preliminaries

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 Fig. 2. 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 Fig. 2, 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 Fig. 2, 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.,

$$supp\Big(X^{MCP(X)}\Big) = \frac{|\{T \in db^{MCP(X)} | X \subseteq T\}|}{|db^{MCP(X)}|}.$$

The support of a rule $(X \Longrightarrow Y)^{MCP(XY)}$ is defined as

$$supp\Big((X \Longrightarrow Y)^{MCP(XY)}\Big) = supp\Big((X \bigcup Y)^{MCP(XY)}\Big).$$

The confidence of this rule is defined as

$$conf\Big((X \Longrightarrow Y)^{MCP(XY)}\Big) = \frac{supp\Big((X \bigcup Y)^{MCP(XY)}\Big)}{supp(X^{MCP(XY)})}.$$

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*

$$conf\Big((X \Longrightarrow Y)^{MCP(XY)}\Big) = c$$

and

$$supp\Big((X \Longrightarrow Y)^{MCP(XY)}\Big) = s$$

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*

$$supp\Big((X \Longrightarrow Y)^{MCP(XY)}\Big) > min\_supp$$

and

$$conf\Big((X \Longrightarrow Y)^{MCP(XY)}\Big) > min\_conf.$$

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 subitemsets (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 percent and confidence 75 percent, in Example 1.3. 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 subitemsets, 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 subitemsets 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 subitemsets $X_i^{MCP(X_k)}$ $(1 \le i < k)$ is also frequent in $db^{MCP(X_k)}$.*

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

TABLE 1
Meanings of Symbols Used

$db^{i,n}$	The partial database of $\mathcal{D}$ formed by a continuous region from P_i to P_n
$\|db^{i,n}\|$	Number of transactions in $db^{i,n}$
$X^{i,n}$	A temporal itemset in partial database $db^{i,n}$
$MCP(\ )$	The maximal common exhibition period of an itemset
$(X \Longrightarrow Y)^{MCP(XY)}$	A general temporal association rule
$supp((X \Longrightarrow Y)^{t,n})$	The support of $X \Longrightarrow Y$ in partial database $db^{t,n}$
$conf((X \Longrightarrow Y)^{t,n})$	The support of $X \Longrightarrow Y$ in partial database $db^{t,n}$
min_supp	Minimum support threshold required
min_conf	Minimum confidence threshold required
min_leng	Minimum length of exhibition period required
TI	A maximal temporal itemset
SI	A corresponding temporal sub-itemset of TI

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 subitemsets 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*.

In addition, it is noted that users are likely to be interested in association rules whose exhibition periods are longer than a certain period. In view of this, we introduce a parameter of the minimum length of the exhibition period, denoted by *min_leng*, as a constraint in rule generation to reflect such a users' requirement in the exhibition period. In other words, for each general temporal association rule $(X \Longrightarrow Y)^{MCP(XY)}$ produced, the value of $MCP(XY)$ should be larger than *min_leng* required, i.e., $MCP(XY) > min\_leng$.

2.2 Traversing the Search Space

As explained, we have to find all maximal temporal itemsets that satisfy min_supp first and then to calculate the occurrences of their corresponding subitemsets 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., 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 $MCP(XY) = MCP(Y) = (t + 1, n)$.

It is worth mentioning that one may deal with the problem we consider with two naive procedures. The first one is to process the conventional mining algorithms in all kinds of combinatorial subdatabases, e.g., $db^{1,3}$, $db^{2,3}$, and $db^{3,3}$ in the foregoing example in Fig. 2, separately. However, due to the huge search space involved, looking at all subsets of $\mathcal{I}$, i.e., $db^{t,n}$ for $1 \leq t \leq n$, is too costly for this approach to be practically used.

Further, 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 subitemsets of $X_k^{t,n}$ at level k with different exhibition periods, i.e., $X_k^{t,n}$, $X_k^{t+1,n}$, $X_k^{t+2,n}, \ldots, X_k^{n,n}$, 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 antimonotone 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.

We next describe the search scenario of $Apriori+$. For the special case $\mathcal{I} = \{A^{1,n}, B^{1,n}, X^{2,n}, Y^{3,n}\}$, we visualize the search space that forms a lattice in Fig. 3. The frequent itemsets are located in the upper part of the figure whereas the infrequent ones are located in the lower part. Assume that the bold border separates the frequent itemsets from the infrequent ones. The basic principle of $Apriori+$ is to employ this border to efficiently prune the search space. As soon as the border line is found, we are able to restrict ourselves on determining the support values of the itemsets above the border and to ignore the itemsets below. However, it should be noted that a linearly growing number of temporal items still implies an exponential growing number of itemsets to be considered. In fact, as will be validated by experimental results later, the increase of candidates often causes a huge increase of execution time and a drastic performance degradation, meaning that without utilizing the partitioning and progressive support counting techniques we propose, a direct extension to priori work is not able to handle the general temporal association rule mining efficiently.

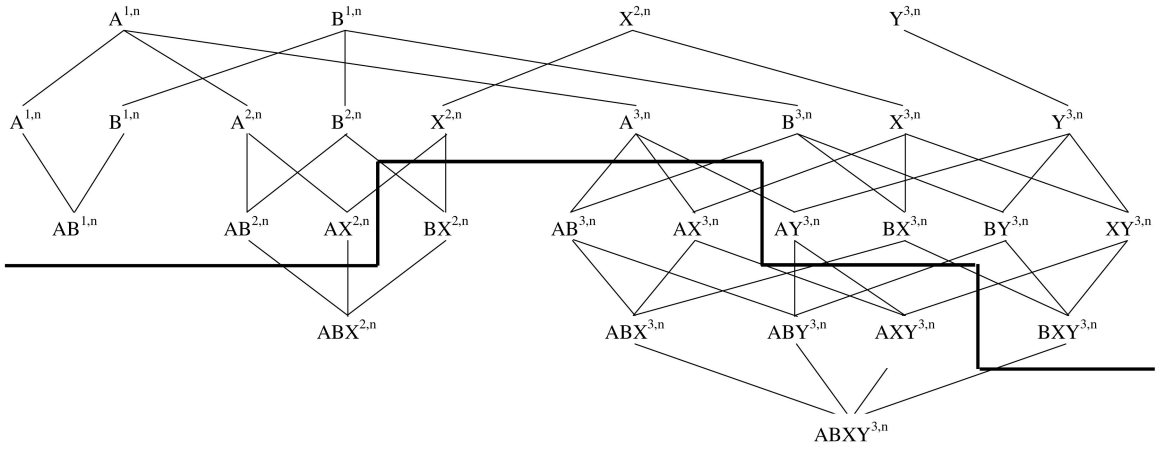


Fig. 3. Traversing the search space for existing algorithms, e.g., Apriori-like algorithms.

3 Mining General Temporal Association Rules

We present an illustrative example for the operations of *PPM* in Section 3.1. A detailed description of Algorithm *PPM* is given in Section 3.2. The correctness of Algorithm *PPM* is proven in Section 3.3.

3.1 An Illustrative Example of Algorithm *PPM*

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.

The operation of algorithm PPM can be best understood by an illustrative example described below and its corresponding flowchart is depicted in Fig. 4. Recall the transaction database shown in Fig. 2, 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 four transactions, i.e., partition P_1, 2-itemsets $\{BD, BC, CD, AD\}$ are sequentially generated, as shown in Fig. 5. 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 Fig. 5, 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 counts of potential candidate 2-itemsets are recorded (of type α and type β). From Fig. 5, it is noted that since there are also four transactions in P_2, the filtering threshold of those itemsets carried out from the previous phase (that

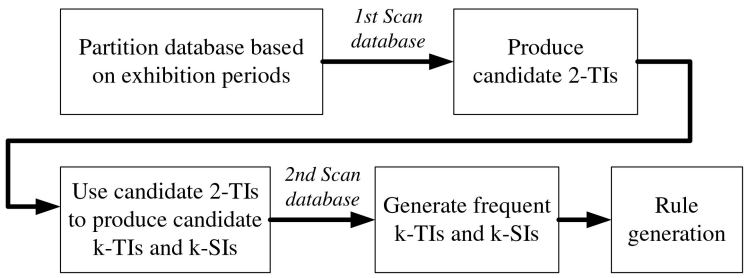


Fig. 4. The flowchart of PPM.

P_1			
	C_2	start	count
○	BD	1	2
○	BC	1	2
	CD	1	1
	AD	1	1

$P_1 + P_2$			
	C_2	start	count
	BD	1	2
○	BC	1	4
	BE	2	1
○	CE	2	2
○	DE	2	2
	AB	2	1
	AC	2	1
	CD	2	1

$P_1 + P_2 + P_3$			
	C_2	start	count
○	BC	1	5
○	CE	2	3
	DE	2	2
	BE	3	1
○	BF	3	3
	CF	3	1
	EF	3	1
	AD	3	1
	BD	3	1
	DF	3	1

After 1st scan database D, we have candidate itemsets (relative support = 30%) as follows:
$\{B^{1,3}\}, \{B^{3,3}\}, \{C^{1,3}\}, \{C^{2,3}\}, \{E^{2,3}\}, \{F^{3,3}\}, \{BC^{1,3}\}, \{BF^{3,3}\}, \{CE^{2,3}\}$

Candidate Itemsets		count	S_X^R
C_1	$\{B^{1,3}\}$	8	4
	$\{B^{3,3}\}$	3	3
	$\{C^{1,3}\}$	6	4
	$\{C^{2,3}\}$	4	3
	$\{E^{2,3}\}$	4	3
	$\{F^{3,3}\}$	3	2
C_2	$\{BC^{1,3}\}$	5	4
	$\{BF^{3,3}\}$	3	2
	$\{CE^{2,3}\}$	3	3

Pruning ⟹

Frequent Itemsets		count
L_1	$\{B^{1,3}\}$	8
	$\{B^{3,3}\}$	3
	$\{C^{1,3}\}$	6
	$\{C^{2,3}\}$	4
	$\{E^{2,3}\}$	4
	$\{F^{3,3}\}$	3
L_2	$\{BC^{1,3}\}$	5
	$\{BF^{3,3}\}$	3
	$\{CE^{2,3}\}$	3

After 2nd scan database D, we have frequent itemsets (relative support = 30%) as follows:
$\{B^{1,3}\}, \{B^{3,3}\}, \{C^{1,3}\}, \{C^{2,3}\}, \{E^{2,3}\}, \{F^{3,3}\}, \{BC^{1,3}\}, \{BF^{3,3}\}, \{CE^{2,3}\}$

Rules	Support	Confidence
$(B \Rightarrow C)^{1,3}$	41.67%	62.50%
$(C \Rightarrow B)^{1,3}$	41.67%	83.33%
$(B \Rightarrow F)^{3,3}$	75.00%	100.00%
$(F \Rightarrow B)^{3,3}$	75.00%	100.00%
$(C \Rightarrow E)^{2,3}$	37.50%	75.00%
$(E \Rightarrow C)^{2,3}$	37.50%	75.00%

Pruning ⟹

Rules	Support	Confidence
$(C \Rightarrow B)^{1,3}$	41.67%	83.33%
$(B \Rightarrow F)^{3,3}$	75.00%	100.00%
$(F \Rightarrow B)^{3,3}$	75.00%	100.00%
$(C \Rightarrow E)^{2,3}$	37.50%	75.00%
$(E \Rightarrow C)^{2,3}$	37.50%	75.00%

Fig. 5. Frequent temporal itemsets generation for mining general temporal association rules by *PPM*.

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 three 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 Fig. 5. 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 three 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 three candidate 2-itemsets are generated by *PPM*. The correctness of Algorithm *PPM* will be formally proven later.

After generating C_2 from the first scan of database $db^{1,3}$, we employ the scan reduction technique [24] and use C_2 to generate C_k ($k = 2, 3, \ldots, m$), 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 ($k = 1, 2, \ldots, m$) 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 Fig. 5, 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,

nine 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 Fig. 5, 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 Fig. 5, 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 $(X \Rightarrow Y)^{i,n}$ holds if $conf((X \Rightarrow Y)^{i,n}) \geq min\_conf$.

3.2 Algorithm of *PPM*

Initially, a publication database $\mathcal{D}$ is partitioned into n partitions based on the exhibition periods of items and PS, i.e., progressive screen, is empty. Let C_2 be the set of progressive candidate 2-itemsets generated by database $\mathcal{D}$. Three parameters, i.e., n, min_supp, and min_leng, are taken as the input values into Algorithm *PPM*. As mentioned above, the minimum support threshold required is denoted as min_supp. In the process of general temporal association rule generation, we employ the parameter min_leng to be a filtering threshold for frequent itemsets to satisfy the minimal length required for the exhibition period. The procedure of Algorithm *PPM* is outlined below, where Algorithm *PPM* is decomposed into five subprocedures for ease of description.

Algorithm *PPM (n, min_supp, min_leng)*:
Progressive Partition Miner
Initial Subprocedure: The database D is partitioned into n partitions and set $PS = \emptyset$

1. $|db^{1,n}| = \sum_{h=1,n} |P_h|$; // $db^{1,n}$ is partitioned into n partitions
2. $PS = \emptyset$;

Subprocedure I: Generate 2nd level candidate TIs with progressive screen

3. begin for $h = 1$ to n // 1st scan of $db^{1,n}$
4. begin for each 2-itemset $X_2^{t,n} \in P_h$ where $n - t > min\_leng$
5. if ($X_2 \notin PS$)
6. $X_2.count = N_{p_h}(X_2)$;
7. $X_2.start = h$;
8. if ($X_2.count \geq min\_supp * |P_h|$)
9. $PS = PS \cup X_2$;
10. if ($X_2 \in PS$)
11. $X_2.count = X_2.count + N_{p_h}(X_2)$;
12. if ($X_2.count < \lceil min\_supp * \sum_{m=X_2.start,h} |P_m| \rceil$)
13. $PS = PS - X_2$;
14. end
15. end
16. select C_2 from X_2 where $X_2 \in PS$;
17. $PS = \emptyset$;

Subprocedure II: Generate candidate TIs and SIs with the scheme of database scan reduction

18. begin while ($C_k \neq \emptyset$ & $k \geq 2$)
19. $C_{k+1} = C_k \star C_k$; // where $\star$ indicate the operation of convolution
20. $k = k + 1$;
21. end
22. $X_k^{t,n} = \{X_k^{t,n} \subseteq X_k \mid X_k \in C_k\}$; // Candidate TIs generation
23. $SI(X_k^{t,n}) = \{X_j^{t,n} \subseteq subset\ of\ X_k^{t,n} | j < k\}$; // Candidate SIs of TIs generation
24. $PS = PS \cup SI(X_k^{t,n})$;
25. select $X_k^{t,n}$ into C_k where $X_k^{t,n} \in PS$;

Subprocedure III: Generate all frequent TIs and SIs with the 2nd scan of database $\mathcal{D}$

26. begin for $h = 1$ to n
27. for each itemset $X_k^{t,n} \in C_k$
28. $X_k^{t,n}.count = X_k^{t,n}.count + N_{p_h}(X_k^{t,n})$;
29. end
30. for each itemset $X_k^{t,n} \in C_k$
31. if ($X_k^{t,n}.count \geq \lceil min\_supp * |db^{t,n}| \rceil$)
32. $L_k = L_k \cup X_k^{t,n}$;
33. end

Subprocedure IV: Prune out the redundant frequent SIs from L_k

34. for each SI itemset $X_k^{t,n} \in L_k$
35. if ($\nexists$ TI itemset $X_j^{t,n} \subseteq L_j \mid j > k$)
36. $L_k = L_k - X_k^{t,n}$;
37. end
38. return L_k;

Initially, the database $db^{1,n}$ is partitioned into n partitions by executing the *Subprocedure I* (in Step 1), and PS, i.e., progressive screen, is empty (in Step 2). In essence, *Subprocedure I* 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 PS is a superset of the set of all frequent 2-itemsets in $\mathcal{D}$. Algorithm *PPM* constructs PS incrementally by adding candidate 2-itemset to PS and starts counting the number of occurrences for each candidate 2-itemset X_2 in PS whenever X_2 is added to PS. If the cumulative occurrences of a candidate 2-itemset X_2 does not meet the partial minimum support required, X_2 is removed from the progressive screen PS. From Step 3 to Step 15 of *Subprocedure I*, Algorithm *PPM* processes one partition at a time for all partitions. When processing partition P_i, each potential candidate 2-itemset X_2 is read and saved to PS, where its exhibition period, i.e., $n - t$, should be larger than the minimum constraint exhibition period min_leng required. The number of occurrences of an itemset X_2 and its starting partition which keeps its first occurrence in PS are recorded in $X_2.count$ and $X_2.start$, respectively. As such, in the end of processing $db^{1,h}$, only an itemset, whose $X_2.count \geq \lceil min\_supp * \sum_{m=X_2.start,h} |P_m| \rceil$, will be kept in PS. Note that a large amount of infrequent TI candidates will be further reduced with the early pruning technique by this progressive portioning processing. Next, in Step 16, we select C_2 from $X_2 \in PS$ and set $PS = \emptyset$ in Step 17.

In *Subprocedure II*, with the scan reduction scheme [24], C_2 produced by the first scan of database is employed to generate C_ks ($k \geq 3$) in main memory from Step 18 to Step 21. Recall

that $X_k^{t,n}$ is a *maximal temporal k-itemset* in a partial database $db^{t,n}$. In Step 22, all candidate TIs, i.e., $X_k^{t,n}$s, are generated from $X_k \in C_k$ with considering the maximal common exhibition period of itemset X_k, where $MCP(X_k) = (t, n)$. After that, from Step 23 to Step 25, we generate all corresponding temporal subitemsets of $X_k^{t,n}$, i.e., $SI(X_k^{t,n})$, to join into PS.

Then, from Step 26 to Step 33 of *Subprocedure III*, we begin the second database scan to calculate the support of each itemset in PS and to find out which candidate itemsets are really frequent TIs and SIs in database $\mathcal{D}$. As a result, those itemsets whose $X_k^{t,n}.count \geq \lceil min\_supp * |db^{t,n}| \rceil$ are the frequent temporal itemsets L_ks.

Finally, in *Subprocedure IV*, we have to prune out those redundant frequent SIs whose TI itemsets are not frequent in database $\mathcal{D}$ from the L_ks. As will be proven in Section 3.3, the output of Algorithm *PPM* consists of frequent itemsets L_ks of database $\mathcal{D}$. According to these output L_ks in Step 38, all kinds of general temporal association rules implied in database $\mathcal{D}$ can be generated in a straightforward method.

Note that *PPM* is able to filter out false candidate itemsets in P_i with a hash table. Same as in [24], 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 *PPM* can be further reduced. As will be validated by our experimental studies, *PPM* provides very efficient solutions for mining general temporal association rules. This feature is, as described earlier, very important for mining the publication-like databases whose data are being exhibited from different starting times.

In addition, the progressive screen produced in each processing phase constitutes the key component to realize the mining of general temporal association rules. Note that Algorithm *PPM* proposed has several important advantages, including 1) with judiciously employing progressive knowledge in the previous phases, *PPM* is able to reduce the amount of candidate itemsets efficiently, which in turn reduces the CPU and memory overhead, and 2) owing to the small number of candidate sets generated, the scan reduction technique can be applied efficiently. As a result, only two scan of the time series database is required.

3.3 Correctness of *PPM*

We now prove the correctness of Algorithm *PPM*. Let $N_{p_h}(X)$ be the number of transactions in partition P_h that contain itemset X, and $|P_h|$ is the number of transactions in partition P_h. Also, let $db^{i,j}$ denote the part of the transaction database formed by a continuous region from partition P_i to partition P_j, and $|db^{i,j}| = \sum_{h=i,j} |P_h|$. We can then define the region ratio of an itemset as follows:

Definition 5. *A region ratio of an itemset X for the transaction database $db^{i,j}$, denoted by $r_{i,j}(X)$, is*

$$r_{i,j}(X) = \frac{\sum_{h=i,j} N_{p_h}(X)}{|db^{i,j}|}.$$

In essence, the region ratio of an itemset is the support of that itemset if only the part of transaction database $db^{i,j}$ is considered.

Lemma 1. *A 2-itemset X_2 remains in the PS after the processing of partition P_j if and only if there exists an i such that for any integer t in the interval $[i, j]$, $r_{i,t}(X_2) \geq min\_supp$, where min_supp is the minimal support required.*

Proof of Lemma 1. We shall prove the "if" condition first. Consider the following two cases. First, suppose the 2-itemset X_2 is not in the progressive candidate set *before* the processing of partition P_i. Since $r_{i,i}(X_2) \geq min\_supp$, itemset X_2 will be selected as a type β candidate itemset by *PPM* after the processing of partition P_i. On the other hand, if the itemset X_2 is already in the progressive candidate set before the processing of partition P_i, itemset X_2 will remain as a type α candidate itemset by *PPM*. Clearly, for the above two cases, itemset X_2 will remain in PS throughout the processing from P_i to P_j since for any integer t in the interval $[i, j]$, $r_{i,t}(X_2) \geq min\_supp$.

We now prove the "only if" condition, i.e., if X_2 remains in PS after the processing of partition P_j, then there exists an i such that for any t in the interval $[i, j]$, $r_{i,t}(X_2) \geq min\_supp$. Note that itemset X_2 can be either type α or type β candidate itemset in the PS after the processing of partition P_j. Suppose X_2 is a type β candidate itemset there, then this implication follows by setting $j = i$ since $r_{i,i}(X_2) \geq min\_supp$. On the other hand, suppose that X_2 is a type α candidate itemset after the processing of P_j, which means itemset X_2 has become a type β candidate itemset in a previous phase. Then, we shall trace backward the type of itemset X_2 from partition P_j (i.e., looking over P_j, P_{j-1}, P_{j-2}, etc.) until the partition that records itemset X_2 as a type β candidate itemset is *first* encountered. (It should be noted that there could be two *discontinuous* regions that record itemset X_2 in the PS, which means that an itemset may get on and off the progressive candidate set through the processing of partitions. This, in turn, means that an itemset may appear as a type β candidate itemset more than once.) By referring the partition identified above as partition P_i, we have, for any t in the interval $[i, j]$, $r_{i,t}(X_2) \geq min\_supp$, completing the proof of this lemma. □

Lemma 1 leads to Lemma 2.

Lemma 2. *An itemset X_2 remains in PS after the processing of partition P_j if and only if there exists an i such that $r_{i,j}(X_2) \geq min\_supp$, where min_supp is the minimal support required.*

Proof of Lemma 2. It can be seen that the proof of "only if" condition follows directly from Lemma 1. We now prove the "if" condition of this lemma. If there exists an i such that $r_{i,j}(X_2) \geq min\_supp$, then we let u be the largest v such that $r_{i,v}(X_2) < min\_supp$. If such a u does not exist, it follows from Lemma 1 that itemset X_2 will remain in PS after the processing of partition P_j. If such a u exists, we have $r_{u+1,j}(X_2) \geq min\_supp$ since $r_{i,u}(X_2) < min\_supp$ and $r_{i,j}(X_2) \geq min\_supp$. It again follows from Lemma 1 that itemset X_2 will remain in PS after the processing of partition P_j. This lemma follows: □

Lemma 2 leads to the following theorem that states the completeness of candidates 2-itemsets generated by the first scan of transaction database $db^{1,n}$ with Algorithm *PPM*.

Theorem 1. *If there exits a frequent itemset $X_2^{t,n}$ in the transaction database $db^{t,n}$ such that $r_{t,n}(X_2) \geq min\_supp$,*

then X_2 will be in the progressive candidate set of itemsets produced by Algorithm PPM.

Proof of Theorem 1. Let n be the number of partitions of the transaction database. Since the itemset $X_2^{t,n}$ is a frequent itemset, we have $r_{t,n}(X_2) \geq min\_supp$, which is in essence a special case of Lemma 2 for $i = t$ and $j = n$, proving this theorem. □

Furthermore, we let $C^{i,j}$, $i \leq j$, be the set of progressive candidate itemsets generated by Algorithm *PPM* with respect to database $db^{i,j}$ after the processing of P_j. We then have the following lemma.

Lemma 3. *For $i \leq t \leq j$, then $C^{t,j} \subset C^{i,j}$.*

Proof of Lemma 3. Assume that there exists a 2-itemset $X_2 \in C^{t,j}$. From the"only if" implication of Lemma 2, it follows that there exists an h such that $r_{h,j}(X_2) \geq s$, where $t \leq h \leq j$. Since $i \leq t \leq j$, we have $i \leq h \leq j$. Then, according to the "if" implication of Lemma 2, itemset X_2 is also in $C^{i,j}$, i.e., $X_2 \in C^{i,j}$. The fact that $C^{t,j} \subset C^{i,j}$ follows. □

Theorem 1 and Lemma 3 lead to the following theorem which states the correctness of Algorithm *PPM*.

Theorem 2. *If there exists a frequent k-itemset $X_k^{t,n}$ in the transaction database $db^{t,n}$ such that $r_{t,n}(X_k) \geq s$, then $X_k^{t,n}$ will be produced by Algorithm PPM.*

Proof of Theorem 2. Since itemset $X_k^{t,n}$ is frequent, we have $r_{t,n}(X_k) \geq min\_supp$. As mentioned above, all of its subitemsets $X_h^{t,n}$s $(h < k)$ will be frequent with $r_{t,n}(X_h) \geq s$. Specifically, $X_2^{t,n}$s are in essence special cases of $X_h^{t,n}$s with $h = 2$. Consequently, according to the implication of Theorem 1, X_2s will be in the progressive candidate set of itemsets, i.e., PS, produced by Algorithm *PPM*. As such, based on an antimonotone Apriori-like heuristic, *i.e., if any length k itemset $X_k^{i,n}$ is not frequent in the database, its length $(k+1)$ superitemset $X_{k+1}^{i,n}$ will never be frequent*, the superitemset $X_k^{t,n}$ of $X_2^{t,n}$ will be produced by Algorithm *PPM*, proving this theorem. □

Further, if there exists a frequent TI 3-itemset $ABC^{t,n}$, for example, in the transaction database $db^{t,n}$, meaning that $r_{t,n}(ABC) \geq min\_supp$, then we have $r_{t,n}(AB) \geq min\_supp$, $r_{t,n}(AC) \geq min\_supp$, and $r_{t,n}(BC) \geq min\_supp$. According to Theorem 1, we learn that all SIs of $ABC^{t,n}$, i.e., $AB^{t,n}$, $AC^{t,n}$, $BC^{t,n}$, $A^{t,n}$, $B^{t,n}$, and $C^{t,n}$, will be in the progressive candidate set of itemsets produced by Algorithm *PPM*. Consequently, Theorem 2 states the correctness of Algorithm *PPM*.

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 transaction data resides in the NTFS file system and is stored on a 30GB IDE 3.5" drive with a measured sequential throughput of 10MB/second. The simulation program was coded in C++. 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. Section 4.3 shows the I/O cost and CPU overhead for *PPM* and *Apriori*$^+$. Results on scaleup experiments are presented in Section 4.4.

TABLE 2
Meanings of Various Parameters

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

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 [4], [24]. Explicitly, we generated several different transaction databases from a set of potentially frequent itemsets to evaluate the performance of *PPM*. These transactions mimic the publication items in a publication database. Note that the efficiency of Algorithm *PPM* has been evaluated by some real databases, such as bookstore transaction databases and grocery sales data. However, we show the experimental results from synthetic transaction data so as to obtain results of different workload parameters. 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*.

As mentioned before, the *Apriori*$^+$ algorithm is an extended version of Apriori-like algorithms to deal with the mining problem in publication databases. As will be shown by our experimental results, with the progressive partition technique that carries cumulative information selectively, the execution time of *PPM* is, in orders of magnitude, smaller than that required by *Apriori*$^+$.

4.2 Experiment One: Relative Performance

We first conducted several experiments to evaluate the relative performance of *Apriori*$^+$ and *PPM*. As shown in Fig. 6, the experimental results are consistent for various values of n, $|L|$ and N on data set T10-I4-D100, e.g., T10-I4-D100(N20-L4-n20). For interest of space, we only report the results on $|L| = 2,000$ and $N = 10,000$ in the following experiments. In addition, the number of partitions in the database is selected as $n = 10$. Fig. 6 shows the relative execution times for both two algorithms as the minimum support threshold is decreased from 1 percent support to 0.1 percent support. When the support threshold is high, there are only a limited number of frequent itemsets

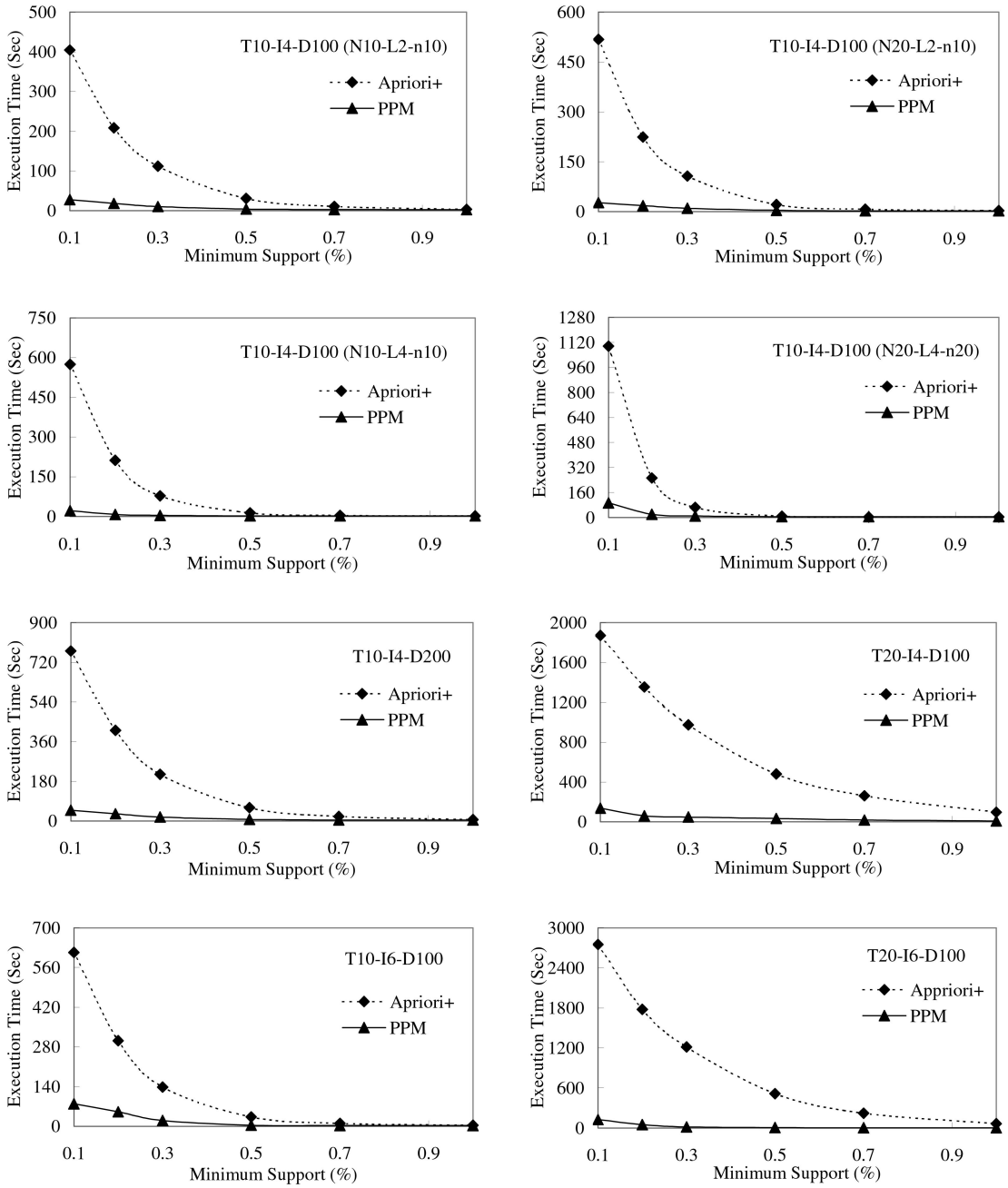


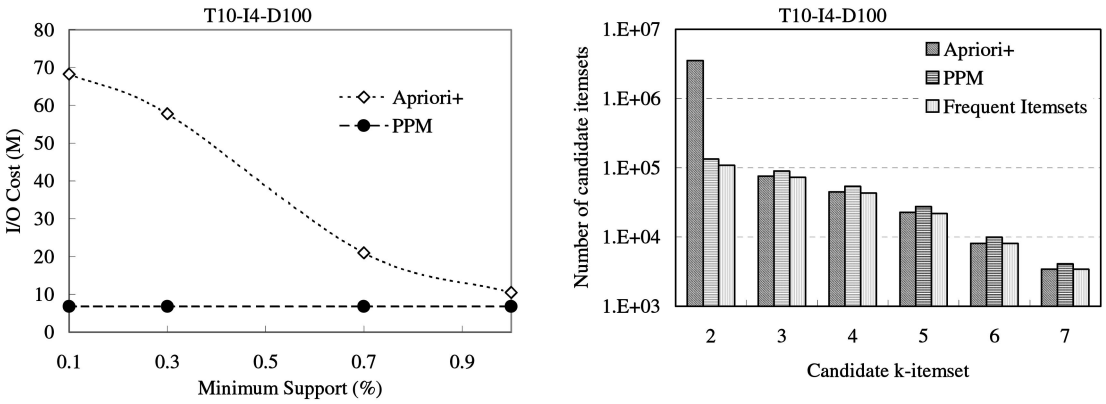
Fig. 6. Relative performance studies.

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. In fact, *PPM* outperforms *Apriori* in both CPU and I/O costs, which are evaluated next.

4.3 Experiment Two: Evaluation of I/O Cost and CPU Overhead

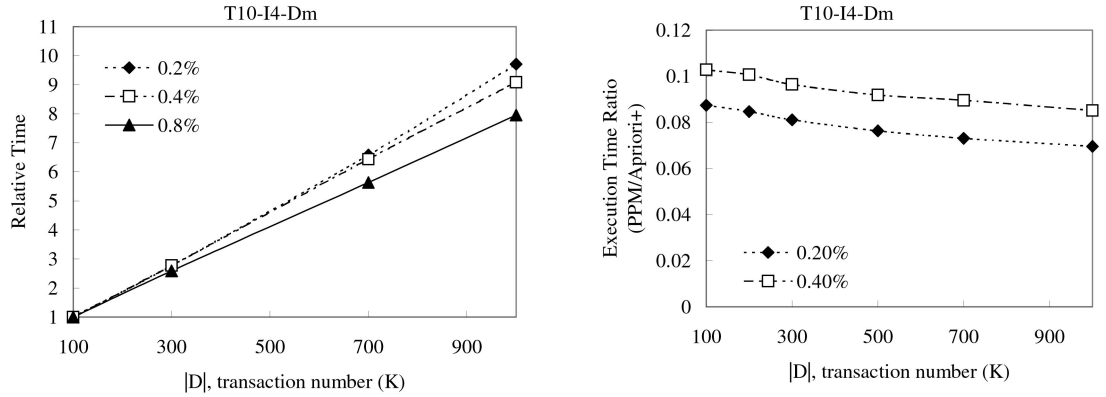
To evaluate the corresponding of I/O cost, same as in [25], we assume that each sequential read of a byte of data consumes one unit of I/O cost and each random read of a byte of data consumes two units of I/O cost. Fig. 7a shows the number of database scans and the I/O costs of $Apriori^+$ and *PPM* over the data set T10-I4-D100. As shown in Fig. 7a, *PPM* outperforms $Apriori^+$. Note that the large amount of database scans is the performance bottleneck when the database size does not fit into main memory. In view of that, *PPM* is advantageous since only two scan, of the publication database is required, which is independent of the variance in minimum supports.

As explained before, *PPM* substantially reduces the number of candidate itemsets generated. The effect is particularly important for the candidate 2-itemsets. The experimental results in Fig. 7b show the candidate itemsets generated by $Apriori^+$ and *PPM* across the whole processing on the data set T10-I4-D100 with minimum support threshold $min\_supp = 0.2$ percent. As shown in Fig. 7b, *PPM* leads to a



(a) (b)

Fig. 7. I/O cost and CPU overhead performance. (a) I/O cost performance over data set T10-I4-D100. (b) Number of candiate itemsets on T10-I4-D100.



(a) (b)

Fig. 8. Scale-up performance of PPM and the execution time ratio between PPM and $Apriori^+$. (a) Scale-up performance in various value of $|D|$. (b) Execution time ratio in various value of $|D|$.

96 percent candidate reduction rate in C_2 when being compared to $Apriori^+$. This feature of *PPM* enables it to efficiently reduce the CPU and memory overhead. Note that the number of candidate 2-itemsets produced by *PPM* approaches to its theoretical minimum, i.e., the number of large 2-itemsets. Recall that the C_3 in $Apriori^+$ has to be obtained by L_2 due to the large size of their C_2. As shown in Fig. 7b, the value of $|C_k|$ ($k \geq 3$) is only slightly larger than that of $Apriori^+$, even though *PPM* only employs C_2 to generate C_ks, thus fully exploiting the benefit of scan reduction.

4.4 Experiment Three: Scaleup Performance

In this experiment, we examine the scaleup performance of Algorithm *PPM*. The scale-up results for different selected data sets are obtained. Fig. 8 shows the scale-up performance of Algorithm *PPM* as the values of $|D|$ increase. Three different minimum supports are considered. We obtained the results for the data set 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 data set in the Fig. 8a. Note that, as shown in Fig. 8a the execution time only slightly increases with the growth of the database size, showing good scalability of *PPM*.

To further understand the impact of $|D|$ to the relative performance of algorithms *PPM* and $Apriori^+$ algorithms, we conduct the scale-up experiments for both *PPM* and $Apriori^+$ with two minimum support thresholds 0.2 percent and 0.4 percent. The results are shown in Fig. 8b where the value in y-axis corresponds to the ratio of the execution time of *PPM* to that of $Apriori^+$. Fig. 8b shows the referenced ratio obtained from a publication-like database over data sets of T10-I4-Dm. The execution-time-ratio of *PPM* to $Apriori^+$ decreases when the amount of database $|D|$ grows larger, meaning that the advantage of *PPM* over $Apriori^+$ increases as the database size increases.

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* not only significantly reduced I/O and CPU cost by the concepts of

progressive counting and scan reduction techniques, but also effectively controlled memory utilization by proper partitioning. 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. The correctness of *PPM* is proven and some of its theoretical properties are derived. Extensive simulations have been performed to evaluate performance of Algorithm *PPM*. Sensitivity analysis of various parameters was conducted to provide many insights into Algorithm *PPM*. It was noted that the improvement achieved by *PPM* increases as the size of the database increases.

ACKNOWLEDGMENTS

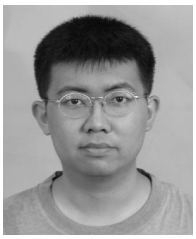
The authors are supported in part by the Ministry of Education project no. 89-E-FA06-2-4-7 and the National Science Council, project nos. NSC 91-2213-E-002-034 and NSC 91-2213-E002-045, Taiwan, Republic of China.

REFERENCES

[1] R. Agarwal, C. Aggarwal, and V.V.V. Prasad, "A Tree Projection Algorithm for Generation of Frequent Itemsets," *J. Parallel and Distributed Computing,* 2000.
[2] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proc. ACM SIGMOD,* pp. 207-216, May 1993.
[3] R. Agrawal and J.C. Shafer, "Parallel Mining of Association Rules: Design, Implementation, and Experience," *IEEE Trans. Knowledge and Data Eng.,* pp. 487-499, Dec. 1996.
[4] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," *Proc. 20th Int'l Conf. Very Large Data Bases,* pp. 478-499, Sept. 1994.
[5] J.M. Ale and G. Rossi, "An Approach to Discovering Temporal Association Rules," *Proc. ACM Symp. Applied Computing,* 2000.
[6] C. Bettini, X.S. Wang, and S. Jajodia, "Mining Temporal Relationships with Multiple Granularities in Time Sequences," *Bull. IEEE Computer Soc. Technical Committee on Data Eng.,* 1998.
[7] M.-S. Chen, J. Han, and P.S. Yu, "Data Mining: An Overview from Database Perspective," *IEEE Trans. Knowledge and Data Eng.,* vol. 8, no. 6, pp. 866-883, Dec. 1996.
[8] X. Chen and I. Petr, "Discovering Temporal Association Rules: Algorithms, Language, and System," *Proc. 2000 Int'l Conf. Data Eng.,* 2000.
[9] X. Chen, I. Petrounias, and H. Heathfield, "Discovery of Association Rules in Temporal Databases," *Proc. Issues and Applications of Database Technology,* 1998.
[10] D. Cheung, J. Han, V. Ng, and C.Y. Wong, "Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique," *Proc. 1996 Int'l Conf. Data Eng.,* pp. 106-114, Feb. 1996.
[11] G. Dong and J. Li, "Efficient Mining of Emerging Patterns: Discovering Trends and Differences," *Knowledge Discovery and Data Mining,* pp. 43-52, 1999.
[12] E. Cohen et al., "Finding Interesting Associations without Support Pruning," *IEEE Trans. Knowledge and Data Eng.,* vol. 13, no. 1, pp. 64-78, Jan./Feb. 2001.
[13] J. Han, G. Dong, and Y. Yin, "Efficient Mining of Partial Periodic Patterns in Time Series Database," *Proc. 15th Int'l Conf. Data Eng.,* pp. 106-115, Mar. 1999.
[14] J. Han and Y. Fu, "Discovery of Multiple-Level Association Rules from Large Databases," *Proc. 21st Int'l Conf. Very Large Data Bases,* pp. 420-431, Sept. 1995.
[15] J. Han and J. Pei, "Mining Frequent Patterns by Pattern-Growth: Methodology and Implications," *ACM SIGKDD Explorations,* Dec. 2000.
[16] J. Hipp, U. Guntzer, and G. Nakhaeizadeh, "Algorithms for Association Rule Mining—A General Survey and Comparison," *ACM SIGKDD Explorations,* vol. 2, no. 1, pp. 58-64, July 2000.
[17] E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzani, "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases," *Proc. ACM-SIGMOD Conf. Management of Data,* 2001.
[18] C.-H. Lee, C.-R. Lin, M.-S. Chen, "Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining," *Proc. ACM 10th Int'l Conf. Information and Knowledge Management,* Nov. 2001.
[19] Y. Li, P. Ning, X.S. Wang, and S. Jajodia, "Discovering Calendar-Based Temporal Association Rules," *TIME,* pp. 111-118, 2001.
[20] J.-L. Lin and M.H. Dunham, "Mining Association Rules: Anti-Skew Algorithms," *Proc. Int'l Conf. Data Eng.,* pp. 486-493, 1998.
[21] B. Liu, W. Hsu, and Y. Ma, "Mining Association Rules with Multiple Minimum Supports," *Proc. Int'l Conf. Knowledge Discovery and Data Mining,* Aug. 1999.
[22] C.C. Liu, J.L. Hsu, and A.L.P. Chen, "Efficient Theme and Non-Trivial Repeating Pattern Discovering in Music Databases," *Proc. IEEE Int'l Conf. Data Eng.,* 1999.
[23] J.-S. Park, M.-S. Chen, and P.S. Yu, "Mining Association Rules with Adjustable Accuracy," *Proc. ACM Sixth Int'l Conf. Information and Knowledge Management,* pp. 151-160, Nov. 1997.
[24] J.-S. Park, M.-S. Chen, and P.S. Yu, "Using a Hash-Based Method with Transaction Trimming for Mining Association Rules," *IEEE Trans. Knowledge and Data Eng.,* vol. 9, no. 5, pp. 813-825, Oct. 1997.
[25] J. Pei, J. Han, and L.V.S. Lakshmanan, "Mining Frequent Itemsets with Convertible Constraints," *Proc. Int'l Conf. Data Eng.,* 2001.
[26] J.F. Roddick and M. Spiliopoulou, "A Survey of Temporal Knowledge Discovery Paradigms and Methods," *IEEE Trans. Knowledge and Data Eng.,* pending publication, 2000.
[27] A. Savasere, E. Omiecinski, and S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases," *Proc. 21st Int'l Conf. Very Large Data Bases,* pp. 432-444, Sept. 1995.
[28] R. Srikant and R. Agrawal, "Mining Generalized Association Rules," *Proc. the 21st Int'l Conf. Very Large Data Bases,* pp. 407-419, Sept. 1995.
[29] R. Srikant and R. Agrawal, "Mining Quantitative Association Rules in Large Relational Tables," *Proc. ACM-SIGMOD Conf. Management of Data,* 1996.
[30] A.U. Tansel and N.F. Ayan, "Discovery of Association Rules in Temporal Databases," *Proc. AAAI Knowledge Discovery in Databases,* 1998.
[31] H. Toivonen, "Sampling Large Databases for Association Rules," *Proc. 22nd Very Large Data Base Conf.,* pp. 134-145, Sept. 1996.
[32] A.K.H. Tung, J. Han, L.V.S. Lakshmanan, and R.T. Ng, "Constraint-Based Clustering in Large Databases," *Proc. 2001 Int'l Conf. Database Theory,* Jan. 2001.
[33] R. Villafane, K.A. Hua, D. Tran, and B. Maulik, "Mining Interval Time Series," *Data Warehousing and Knowledge Discovery,* pp. 318-330, 1999.
[34] C. Yang, U. Fayyad, and P. Bradley, "Efficient Discovery of Error-Tolerant Frequent Itemsets in High Dimensions," *Proc. Seventh ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* 2001.

Chang-Hung Lee received the BS and PhD degrees in electrical engineering from the National Taiwan University, Taipei, in 1996 and 2002, respectively. Dr. Lee has published in ACM SIGKDD, IEEE ICDCS, IEEE ICDM, and SIAM SDM. He is currently a research staff member at BenQ Inc., Taipei, Taiwan, leading projects in wireless multimedia applications. His research interests include distributed database systems, data mining, mobile computing systems, wireless multimedia, and flat panel display.

Ming-Syan Chen received the BS degree in electrical engineering from the National Taiwan University, Taipei, and the MS and PhD degrees in computer, information, and control engineering from The University of Michigan, Ann Arbor, in 1985 and 1988, respectively. Dr. Chen is currently a professor in the Electrical Engineering Department, National Taiwan University, Taipei. He was a research staff member at IBM Thomas J. Watson Research Center, Yorktown Heights, New York, from 1988 to 1996. His research interests include database systems, data mining, mobile computing systems, and multimedia networking, and he has published more than 160 papers in his research areas. In addition to serving as program committee members in many conferences, Dr. Chen served as an associate editor of *IEEE Transactions on Knowledge and Data Engineering* on data mining and parallel database areas from 1997 to 2001, an editor of *Journal of Information Science and Engineering*, a distinguished visitor of IEEE Computer Society for Asia-Pacific from 1998 to 2000, and program chair of PAKDD-02 (Pacific Area Knowledge Discovery and Data Mining), program vice-chair of VLDB-2002 (Very Large Data Bases) and ICPP 2003, general chair of Real-Time Multimedia System Workshop in 2001, program chair of IEEE ICDCS Workshop on Knowledge Discovery and Data Mining in the World Wide Web in 2000, and program cochairs of the International Conference on Mobile Data Management in 2003, International Computer Symposium (ICS) on Computer Networks, Internet and Multimedia in 1998 and 2000, and ICS on Databases and Software Engineering in 2002. He was a keynote speaker on Web data mining at the International Computer Congress in Hong Kong, 1999, a tutorial speaker on Web data mining in DASFAA-1999 and on parallel databases in the 11th IEEE International Conference on Data Engineering in 1995 and also a guest coeditor for *IEEE Transactions on Knowledge and Data Engineering* on a special issue for data mining in December 1996. He holds, or has applied for, 18 US patents and seven ROC patents in the areas of data mining, Web applications, interactive video playout, video server design, and concurrency and coherency control protocols. He received the Outstanding Innovation Award from IBM Corporate in 1994 for his contribution to parallel transaction design and implementation for a major database product, and has received numerous awards for his research, teaching, inventions, and patent applications. Dr. Chen is a senior member of the IEEE, the IEEE Computer Society, and a member of ACM.



Cheng-Ru Lin received the BS degree in electrical engineering from the National Taiwan University, Taipei, in 1999. He is currently a PhD candidate in Electrical Engineering Department, National Taiwan University, Taipei and is expected to graduate in the summer of 2003. He has published in the *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Parallel and Distributed Systems*, ICDM, ACM SIGKDD, CIKM, and SIAM SDM, and also received the ACM SIGMOD Research Student Award. His research interests include distributed systems, databases, data clustering, and data mining.