Twain: Two-End Association Miner with Precise Frequent Exhibition Periods

JEN-WEI HUANG
National Taiwan University
BI-RU DAI
National Taiwan University of Science and Technology
and
MING-SYAN CHEN
National Taiwan University

8

We investigate the general model of mining associations in a temporal database, where the exhibition periods of items are allowed to be different from one to another. The database is divided into partitions according to the time granularity imposed. Such temporal association rules allow us to observe short-term but interesting patterns that are absent when the whole range of the database is evaluated altogether. Prior work may omit some temporal association rules and thus have limited practicability. To remedy this and to give more precise frequent exhibition periods of frequent temporal itemsets, we devise an efficient algorithm *Twain* (standing for *TWo end AssocIation miNer*.) *Twain* not only generates frequent patterns with more precise frequent exhibition periods, but also discovers more interesting frequent patterns. *Twain* employs Start time and End time of each item to provide precise frequent exhibition period while progressively handling itemsets from one partition to another. Along with one scan of the database, *Twain* can generate frequent 2-itemsets directly according to the cumulative filtering threshold. Then, *Twain* adopts the scan reduction technique to generate all frequent k-itemsets ($k > 2$) from the generated frequent 2-itemsets. Theoretical properties of *Twain* are derived as well in this article. The experimental results show that *Twain* outperforms the prior works in the quality of frequent patterns, execution time, I/O cost, CPU overhead and scalability.

Categories and Subject Descriptors: H2.8 [**Database Management**]: Database Application—*data mining*

General Terms: Algorithms

Additional Key Words and Phrases: Association, temporal

This work was supported in part by the National Science Council of Taiwan, R.O.C., under Contracts NSC93-2752-E-002-006-PAE.
Authors' address: National Taiwan University, Taipei, Taiwan, Correspondence email: mschen@cc.ee.ntu.edu.tw.

ACM Reference Format:
Huang, J.-W., Dai, B.-R., and Chen, M.-S. 2007. Twain: Two-end association miner with precise frequent exhibition periods. ACM Trans. Knowl. Discov. Data. 1, 2, Article 8 (August 2007), 33 pages. DOI = 10.1145/1267066.1267069 http://doi.acm.org/10.1145/1267066.1267069

1. INTRODUCTION

Along with the explosion of data from various applications, a great deal of research work has been done in the data mining area to discover interesting but unknown knowledge from large databases. Data mining techniques include association rules mining, classification, clustering, mining time series, and sequential pattern mining, to name a few [Han and Kamber 2000]. Among others, association rules mining receives significant attention. The problem of mining association rule is to discover the implication relation among items such that the presence of some items implies the presence of other items in the same transaction. Mathematically, let $\mathcal{I} = \{x_1, x_2, ..., x_m\}$ be a set of items and T be a set of transactions, where each transaction t is a set of items such that $t \subseteq \mathcal{I}$. Let X be a set of items, called an *itemset*. A transaction t is said to contain X if and only if $X \subseteq t$. An association rule is an implication of the form $X \Longrightarrow Y$, where $X \subset \mathcal{I}$, $Y \subset \mathcal{I}$ and $X \bigcap Y = \phi$. The rule $X \Longrightarrow Y$ holds in the transaction set T with *confidence* $c\%$ if $c\%$ of transactions in T that contain X also contain Y. The rule $X \Longrightarrow Y$ has *support* $s\%$ in the transaction set T if $s\%$ of transactions in T contain $X \bigcup Y$. A typical example of an association rule is that 90% of customer transactions that purchase milk and bread also purchase eggs. In the pioneering work, Apriori [Agrawal et al. 1993], the problem of mining association rules was decomposed into two steps: (1) generate all frequent itemsets that satisfy the minimum support; (2) generate all association rules that satisfy the minimum confidence from those frequent itemsets. Since the second step is straightforward, the problem of mining association rules can be reduced to the problem of finding all frequent itemsets that satisfy the minimum support.

Some fast algorithms based on the level-wise Apriori framework [Agrawal and Srikant 1994; Park et al. 1997], partitioning [Lin and Dunham 1998; Mueller 1995; Savasere et al. 1995], and sampling [Toivonen 1996] are proposed to remedy the performance bottleneck of Apriori. Additionally, other novel mining techniques, including TreeProjection [Agarwal et al. 2000], FP-growth algorithms Han and Pei 2000; Han et al. 2000a, 2000b; Pei et al. 2001], and Hidden Markov Model [Besemann and Denton 2005] also provide significant improvement. Moreover, several variants of mining algorithms are presented to increase more mining capabilities, such as incremental updating [Ayad et al. 2001; Lee et al. 2001b], mining of generalized and multi-level rules [Han and Fu 1995; Srikant and Agrawal 1995], mining of quantitative rules [Ke et al. 2006; Srikant and Agrawal 1996], mining of multi-dimensional rules [Ng and Han 1994; Yang et al. 2001], constraint-based rule mining [Kifer et al. 2002; Lakshmanan et al. 1998; Lakshmanan et al. 1999; Muhonen and Toivonen 2005; Pei and Han 2000; Steinbach et al. 2005; Szymon and Jaroszewicz 2006], mining with multiple minimum supports [Liu et al. 1999; Wang et al. 2000],

mining associations among correlated or infrequent items [Chen et al. 2004; Cohen et al. 2001; Xuan-Hieu et al. 2005], and interestingness measurement [Blanchard et al. 2005].

Traditional association rule mining algorithms treat all transactions at different timestamps as the same in time domain. They use the same support counting basis for all transactions and deal with the whole transaction database at the same time. These methods may not lead to interesting mining results because of the lack of time granularities imposed. When the occurrence time of each transaction is taken into consideration, a transaction database can be divided into partitions according to the time granularity. We would like to find interesting association rules with time constraints in such a temporal database. This problem is referred to as temporal association rule mining [Ale and Rossi 2000; Bettini et al. 1998; Chen and Petr 2000; Chen et al. 1998; Harms and Deogun 2004; Jiang and Gruenwald 2006; Lee et al. 2003, 2001a; Tansel and Ayan 1998]. In contrast, sequential pattern mining algorithms focus on time relationship of itemsets in a sequence database. Each sequence consists of a list of ordered itemsets. The itemsets in a sequential pattern should come in a certain order and form a subsequence. Then, sequential pattern mining is to find subsequences whose occurrence frequencies are no less than the threshold from the set of sequences. However, sequential pattern mining still discards the timestamp of each itemset in a sequence.

Although previous algorithms provide various solutions for fast generation of temporal association rules, they cannot be effectively applied to the temporal transaction database where the exhibition periods of the items are different from one to another. The exhibition period of an item is the time duration from the partition when this item starts to appear in the transaction database to the partition when this item no longer exists. That is, the exhibition period is the time duration when the item is purchased. In fact, items in a real transaction database usually have different exhibition periods. We illustrate this problem by the following examples.

Example 1.1. Consider the transaction database as shown in Figure 1. The database contains a set of transactions from January to April and is divided into four partitions according to the time granularity. The exhibition period of each item is given in the right of Figure 1. Assume that the minimum support and the minimum confidence are $min\_supp = 30\%$ and $min\_conf = 75\%$, respectively. Traditional mining algorithms, which use the same support counting basis, total number of transactions in the database, will only generate frequent itemsets $\{A\}$, $\{B\}$, $\{C\}$ and $\{F\}$. Thus, no rule will be discovered in this case.

We can see that the limitation of conventional mining algorithms lies in the absence of a relative support counting basis for each itemset in Example 1.1. The itemsets with longer exhibition periods (e.g., A, B, C and F) are more likely to be frequent than those with shorter exhibition periods (e.g., D and E). Consequently, the association rules generated by conventional algorithms are those consisting of long-term items (e.g., milk and bread are frequently purchased together, which is, however, of less interest to us). On the contrary,

			Transaction Database	
		Date	TID	Itemset
D	P1	January	T01	A C F
			T02	A D F
			T03	C F
			T04	A D
	P2	February	T05	B C D F
			T06	C F
			T07	B C D
			T08	A B
	P3	March	T09	C E F
			T10	A B F
			T11	E F
			T12	B
	P4	April	T13	E
			T14	A B E
			T15	A B
			T16	A B

Item	start	end
A	January	April
B	February	April
C	January	March
D	January	February
E	March	April
F	January	March

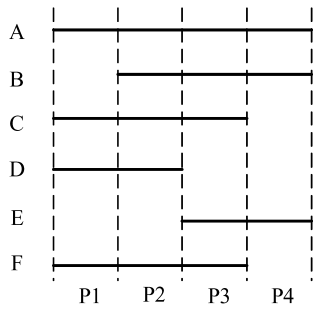


Fig. 1. An illustrative database where the items have individual exhibition periods.

some short-term itemsets, such as those composed of seasonal elements, which may be really "frequent" and interesting in their exhibition periods, are less likely to be identified as frequent. As will be shown later in Example 1.2, such general temporal association rules do exist in the illustrative database when the individual exhibition periods of items are taken into consideration, while being absent when the whole range of the database is evaluated altogether.

To address this issue, we explore in our prior work [Chang et al. 2002] a new model of mining general temporal association rules where the items are allowed to have different exhibition periods, and their supports is made in accordance with their exhibition periods. Explicitly, we introduce the notion of *maximal common exhibition period* (abbreviated as *MCP*) and define the *relative support* to provide a relative support counting basis for each itemset. The *MCP* of the itemset X, denoted by $[p, q]$, is defined as the period between the *latest-exhibition-start time* p and the *earliest-exhibition-end time* q of all items belonging to X. For example, for the itemset BE in Figure 1, its *MCP* is $[2, 4]$ since the *latest-exhibition-start time* of the items B and E is 2 and the *earliest-exhibition-end time* of the items B and E is 4. The relative support of the itemset X is given by $supp(X^{p,q}) = \frac{|\{t \in T^{p,q} | X \subseteq t\}|}{T^{p,q}}$ where $T^{p,q}$ indicates the partial database bounded by $[p, q]$, t is the a transaction, and $|\{t \in T^{p,q} | X \subseteq t\}|$ indicates the number of transactions that contain X in $T^{p,q}$. The general temporal association rule $(X \Longrightarrow Y)^{p,q}$ is termed to be frequent within its *MCP* $[p, q]$ if and only if its relative support is not smaller than the minimum support required (i.e., $supp((X \cup Y)^{p,q}) \geq min\_supp$), and its confidence is not smaller than the minimum confidence needed (i.e., $conf((X \Longrightarrow Y)^{p,q}) = \frac{supp((X \cup Y)^{p,q})}{supp(X^{p,q})} \geq min\_conf$).

Example 1.2. Based on the definitions above, the frequent general temporal association rules in Example 1.1 can be identified as follows:

(1) $(A \Longrightarrow B)^{2,4}$ with $supp(AB^{2,4}) = \frac{5}{12}$ and $conf((A \Longrightarrow B)^{2,4}) = \frac{supp(AB^{2,4})}{supp(A^{2,4})} = \frac{5}{5}$

(2) $(D \Longrightarrow B)^{2,2}$ with $supp(BD^{2,2}) = \frac{2}{4}$ and $conf((D \Longrightarrow B)^{2,2}) = \frac{supp(BD^{2,2})}{supp(D^{2,2})} = \frac{2}{2}$

(3) $(C \Longrightarrow F)^{1,3}$ with $supp(CF^{1,3}) = \frac{5}{12}$ and $conf((C \Longrightarrow F)^{1,3}) = \frac{supp(CF^{1,3})}{supp(C^{1,3})} = \frac{5}{6}$

(4) $(E \Longrightarrow F)^{3,3}$ with $supp(EF^{3,3}) = \frac{2}{4}$ and $conf((E \Longrightarrow F)^{3,3}) = \frac{supp(EF^{3,3})}{supp(E^{3,3})} = \frac{2}{2}$

(5) $(D \Longrightarrow BC)^{2,2}$ with $supp(BCD^{2,2}) = \frac{2}{4}$ and
$conf((D \Longrightarrow BC)^{2,2}) = \frac{supp(BCD^{2,2})}{supp(D^{2,2})} = \frac{2}{2}$

(6) $(BC \Longrightarrow D)^{2,2}$ with $supp(BCD^{2,2}) = \frac{2}{4}$ and
$conf((BC \Longrightarrow D)^{2,2}) = \frac{supp(BCD^{2,2})}{supp(BC^{2,2})} = \frac{2}{2}$

(7) $(BD \Longrightarrow C)^{2,2}$ with $supp(BCD^{2,2}) = \frac{2}{4}$ and
$conf((BD \Longrightarrow C)^{2,2}) = \frac{supp(BCD^{2,2})}{supp(BD^{2,2})} = \frac{2}{2}$

(8) $(CD \Longrightarrow B)^{2,2}$ with $supp(BCD^{2,2}) = \frac{2}{4}$ and
$conf((CD \Longrightarrow B)^{2,2}) = \frac{supp(BCD^{2,2})}{supp(CD^{2,2})} = \frac{2}{2}$.

Recall that the downward closure property which all prior Apriori-based algorithms relied upon is no longer valid in this new model. The downward closure property, which was first presented by Agrawal and Srikant [1994], shows that all sub-itemsets of a frequent itemset are also frequent. In this way, all Apriori-based algorithms are able to prune the searching space substantially. However, this property is not valid in the new model, the searching space will explode dramatically. We show the invalidation of the property by the following example.

Example 1.3. From the illustrative database Figure 1, we can find that although the itemset *BCD* is frequent in its maximal common exhibition period, the itemsets *BC*, *BD* and *CD* are not all frequent in their corresponding maximal common exhibition periods. Explicitly, the itemset *BC* is infrequent in its maximal common exhibition period [2, 3] since its relative support is only 25% ($< 30\%$). Similarly, the itemset *CD* is infrequent in its maximal common exhibition period [1, 2]. Hence, to determine if an itemset is frequent, the downward closure property is no longer valid.

We first describe an algorithm *SPF* (standing for *Segmented Progressive Filter)* reported in Chang et al. [2002] to address this issue. In essence, *SPF* consists of two major procedures, that is, *Segmentation (*abbreviated as *ProcSG)* and *Progressively Filtering* (abbreviated as *ProcPF)*. The basic idea behind *SPF* is to first divide the database into partitions according to the time granularity imposed. Then, in light of the exhibition period of each item, *SPF* employs *ProcSG* to segment the database into subdatabases in such a way that items in each sub-database will have *either* the common starting time *or* the common ending time. Note that such a segmentation will allow us of counting the itemsets in each subdatabase either forward or backward (in time) efficiently. Then, for each subdatabase, *SPF* utilizes *ProcPF* to progressively filter candidate 2-itemsets

with cumulative filtering thresholds from one partition to another. Since infrequent 2-itemsets are hence filtered out in the early processed partitions, the resulting candidate 2-itemsets will be very close to the frequent 2-itemsets. This feature allows us of adopting the scan reduction technique by generating all candidate k-itemsets ($k > 2$) from candidate 2-itemsets directly [Park et al. 1997].

Although *SPF* can find frequent itemsets with their corresponding *MCPs* the *MCPs* found by *SPF* after the generation of all candidates are usually too rough to show the real frequent exhibition periods of the itemsets. Itemsets may not be always frequent during the whole exhibition period. Furthermore, some rules may not be found by *SPF* due to the limitation of separating the original database into sub-databases. These limitations of *SPF* are illustrated as follows.

(a) *Overestimating the Common Exhibition Period of a Frequent Itemset.* According to Figure 1, the frequent itemset $AB^{2,4}$ with $supp(AB^{2,4}) = \frac{5}{12} > 30\%$ is identified by algorithm *SPF*. However, we can observe from the database that itemset *AB* is only frequent during the period [4, 4] rather than the whole range of [2, 4], which is an overestimated frequent exhibition period.

(b) *Underestimating the Occurrence Frequency of a Frequent Itemset.* Although itemset $AD^{1,1}$ in Figure 1 with $supp(AD^{1,1}) = \frac{2}{4} > 30\%$ is frequent, it cannot be detected by algorithm *SPF*. According to the segmentation procedure of algorithm *SPF*, partitions P_1 and P_2 are considered as a sub-database together, where itemset $AD^{1,2}$ with $supp(AD^{1,2}) = \frac{2}{8} < 30\%$ is not regarded as frequent. Therefore, the frequent itemset $AD^{1,1}$ is consequently omitted by algorithm *SPF* because the support of itemset *AD* in period [1, 1] is underestimated.

In addition to the lack of deriving accurate frequent exhibition periods of frequent temporal itemsets, *SPF* has to find all candidate 2-itemsets without their exhibition periods in the *ProcPF* phase and then derive their *MCPs* after the generation of all candidate k-itemsets. This process is time consuming and can be done at the same time when we discover all candidate 2-itemsets.

To remedy these limitations and to provide precise general temporal association rules with more precise frequent exhibition periods, we introduce the notions of *frequent common exhibition period* (abbreviated as *FCP*) and *maximal frequent common exhibition period* (abbreviated as *MFCP*) in this work. The *FCP* of an itemset X is the common exhibition period of the items belonging to X such that the itemset X starts and continues to be frequent in this period. On the other hand, the *MFCPs* of an itemset X are the distinguishing periods of the union of *FCPs*. For example, in Figure 1, the *FCP* of the itemset *AB* is [4, 4] and the *FCPs* of the itemsets *CF* are [1, 1], [1, 2] and [1, 3]. The reason is that the itemset *AB* starts to be frequent in P_4, and the itemset *CF* starts to be frequent in P_1 and continues to be frequent in P_2 and P_3. Accordingly, the *MFCP* of the itemset *AB* is [4, 4] and the *MFCP* of the itemset *CF* is [1, 3]. Note that identifying the *MFCP* of the frequent temporal itemset is critical in practical applications. Recall that *MCP* of a frequent itemset X covers the whole range of its exhibition period, which includes the segments that itemset

X starts to appear but is not frequent yet and the segments that itemset X becomes infrequent. The practical use of *MCP* is, however, not clear because of the mixture of frequent and infrequent segments in the whole *MCP*. However, *MFCP* allows us to detect the starting point of an itemset to be frequent and to observe the period that this itemset continues to be popular. It is noted that the *MFCP* of an itemset is not necessarily unique. If the occurrence frequency of an itemset oscillates between frequent and infrequent several times, there can be many *MFCPs* of the itemset corresponding to different frequent exhibition periods respectively. As such, every frequent exhibition period of the itemsets can be detected correctly. Therefore, the decision making systems can be supported by providing frequent temporal patterns with the periods which target on the triggering points and the continuous intervals of being popular.

In order to find the more precise frequent temporal itemsets, we propose an efficient algorithm *Twain* (standing for *TWo end AssocIation miNer*) in this work. *Twain* is able to expose real frequent exhibition periods of the itemsets specifically during the candidate generation process. *Twain* progressively cumulates the occurrence frequencies of all candidate 2-itemsets in each partition of the database without the separation of the database and uses a progressive filtering technique to filter out infrequent candidate 2-itemsets from one partition to another. By utilizing *Start time* and *End time* of each candidate itemset, *Twain* can reduce the amount of candidate 2-itemsets very efficiently. In fact, *Twain* generates frequent 2-itemsets immediately after the first scan of the database. As such, *Twain* reduces a huge amount of examining time for checking all candidate 2-itemsets. *Twain* also adopts the scan reduction technique after generating candidate k-itemsets ($k > 2$) from frequent 2-itemsets. In accordance with *Start time* and *End time*, *Twain* can generate the frequent temporal itemsets more precisely than *SPF*. In addition, without the limitation of separating the original database, the frequent patterns whose supports are under-estimated by *SPF* can be discovered by *Twain*. Furthermore, another advantageous feature of *Twain* is that *Twain* possesses the incremental mining capability. More explicitly, *Twain* is able to deal with the incremental database efficiently without re-mining the whole database with the help of *Start time* and *End time.*

The main contribution provided by *Twain* can be listed as follows.

(1) To the best of our knowledge, *Twain* is the first work for two-end association rule mining, which is not restricted by the limitation of having common *Start time* or common *End time* of the items in a temporal database. Without the need of segmenting the database, *Twain* is able to obtain precise frequent exhibition periods of frequent temporal itemsets, and derive precise temporal association rules. Explicitly, *Twain* can find many interesting temporal general association rules which may be overlooked by *SPF*.

(2) *Twain* is efficient in that it is able to generate frequent 2-itemsets directly without scanning the database twice. As validated by our empirical works, this property allows *Twain* to find candidate k-itemsets ($k > 2$) very efficiently.

(3) *Twain* possesses the incremental capability to find frequent 2-itemsets without re-mining the whole database. Therefore, when *Twain* deals with incoming transactions, *Twain* is of graceful performance.

Several related theoretical properties are also derived in this article. The experiment results prove that *Twain* can overcome the drawbacks of *SPF* and generate frequent patterns with better quality no matter what the synthetic datasets or the real datasets are used. Sensitivity analysis on various parameters of the synthetic database is also conducted to provide many insights into *Twain*. The advantages of *Twain* become even more prominent as the size of the database increases. This is indeed an important feature for these two algorithms to be practically used for the mining of a time series database in the real world. Furthermore, *Twain* performs especially better when handling items with short exhibition periods. These results justify the practical importance of *Twain* for the mining of a time series database in the real world. In summary, the experimental results show that *Twain* outperforms other schemes which are extended from prior methods in the quality of frequent patterns, execution time, I/O cost, CPU overhead and scalability.

The rest of this article is organized as follows. Section 2 describes the preliminaries for the general model of mining temporal association rules. The proposed algorithm *Twain* is presented in Section 3 with its correctness proved. The performance analysis of *Twain* is empirically evaluated in Section 4. Finally, we conclude this article with Section 5.

2. PRELIMINARIES

To facilitate our later discussion in this article, we define the problem of mining general temporal association rules in Section 2.1. Related works are surveyed in Section 2.2. In Section 2.3, an extended version of algorithm Apriori, $Apriori^{IP}$, is considered to deal with the problem of mining precise general temporal association rule. Then, we describe our prior work, *SPF*, briefly in Section 2.4. We will include $Apriori^{IP}$ and *SPF* in our experiments as yardsticks for the performance comparison in Section 4.

2.1 Problem Description

In order to deal with the items having different exhibition periods in a transaction database, it is assumed that, without loss of generality, a certain time granularity, for example, *week, month, quarter* or *year*, is imposed by the application database. Let n be the number of partitions generated when the database T is divided by the time granularity. In the model considered, $T^{p,q}$ $(1 \leq p \leq q \leq n)$ denotes the portion of the transaction database formed by a continuous region from the partition P_p to the partition P_q, and $X^{p,q}$ denotes the *temporal itemset* whose items are commonly exhibited from the partition P_p to the partition P_q.

Example 2.1. Consider the database in Figure 1, which is divided into four partitions, P_1, P_2, P_3 and P_4, by the time granularity of *month* since the database records the transactions from January to April. According to the above denotations, $T^{2,3}$ denotes the partial database consisting of partitions P_2 and P_3, and $BC^{2,3}$ denotes the temporal itemset whose items (i.e., B and C) are commonly exhibited from the partition P_2 to the partition P_3.

Definition 1. The itemset $X^{p,q}$ is called a *temporal itemset*, where P_p is the common *Start time* and P_q is the common *End time* of all items belonging to X. $[p, q]$ is the *common exhibition period* of all items in X.

As such, we can define the maximal temporal itemset $X^{p,q}$ and the corresponding temporal sub-itemsets, as follows.

Definition 2. The temporal itemset $X^{p,q}$ is called a *maximal temporal itemset (TI)* if P_p is the *latest Start time* and P_q is the *earliest End time* of all items belonging to X. In this situation, $[p, q]$ is referred to as the *maximal common exhibition period (MCP)* of the itemset X, denoted by $MCP(X)$.

Definition 3. The temporal itemset $Y^{p,q}$ is called a *temporal sub-itemset (SI)* of the maximal temporal itemset $X^{p,q}$ if $Y \subset X$.

Based on Definition 2, the temporal itemset $BCD^{2,2}$ in Figure 1 is deemed a maximal temporal itemset since the latest *Start time* and the earliest *End time* of the items B, C and D are both P_2. In addition, the temporal itemsets $BC^{2,2}$, $BD^{2,2}$ and $CD^{2,2}$ are the corresponding temporal sub-itemsets of $BCD^{2,2}$ in accordance with Definition 2. Note that $BD^{2,2}$ is also a maximal temporal itemset itself, but $BC^{2,2}$ is not since the earliest *End time* of the items B and C is P_3 rather than P_2.

In the conventional problem of mining association rules, the support of the itemset X is determined by $supp(X) = \frac{|\{t \in T^{1,n} | X \subseteq t\}|}{|T^{1,n}|}$, where t is a transaction containing a set of items. This is referred to as the *absolute* support in this article. However, as explained in Section 1, we shall provide a relative support counting basis for each temporal itemset. To this end, we define the relative support for a temporal itemset below.

Definition 4. The *relative support* of the temporal itemset $X^{p,q}$ is defined as :

$$supp(X^{p,q}) = \frac{|\{t \in T^{p,q} | X \subseteq t\}|}{|T^{p,q}|},$$

which is the fraction of the transactions supporting the itemset X in $T^{p,q}$.

With the equitable support counting basis defined in Definition 4, we can then determine whether a maximal temporal itemset is frequent by the following definition.

Definition 5. The maximal temporal itemset $X^{MCP(X)}$ is termed to be *frequent* iff $supp(X^{MCP(X)}) \geq min\_supp$ where min_supp is the minimum support required.

Property 1: All temporal sub-itemsets of a frequent maximal temporal itemset are frequent.

As will be explained later, Property 1 is very important for us to determine the confidence of a general temporal association rule defined below.

Definition 6. The rule $(Y \Longrightarrow Z)^{MCP(X)}$ derived from the maximal temporal itemset $X^{MCP(X)}$ is called a *general temporal association rule* if $Y \subset X$ and $Z = X - Y$.

Definition 7. The *confidence* of the general temporal association rule $(Y \Longrightarrow Z)^{MCP(Y \cup Z)}$ is defined as :

$$conf((Y \Longrightarrow Z)^{MCP(Y \cup Z)}) = \frac{supp((Y \bigcup Z)^{MCP(Y \cup Z)})}{supp(Y^{MCP(Y \cup Z)})}.$$

Note that the calculation of the confidence of a general temporal association rule not only depends on the relative support of the corresponding maximal temporal itemset but also relies on the relative supports of the corresponding temporal sub-itemsets. Property 1 ensures that the relative supports of the corresponding temporal sub-itemsets can be obtained without extra database scans since all temporal sub-itemsets of the frequent maximal temporal itemset are also frequent.

Consequently, given a pair of *min_conf* and *min_supp* required, we can define the frequent general temporal association rule below.

Definition 8. The general temporal association rule $(Y \Longrightarrow Z)^{MCP(Y \cup Z)}$ is termed to be frequent iff $supp((Y \cup Z)^{MCP(Y \cup Z)}) \geq min\_supp$ and $conf((Y \Longrightarrow Z)^{MCP(Y \cup Z)}) \geq min\_conf$.

With these definitions, we can discover general temporal association rules by frequent temporal itemsets with their MCPs. However, to give more precise frequent exhibition period of frequent temporal itemsets, we must define the following notations as well.

Definition 9. A *frequent common exhibition period* (*FCP*) of an itemset X, whose *MCP* is $[p, q]$, is denoted by $[s, u]$ $(p \leq s \leq u \leq q)$ such that the following conditions are followed:

(1) $\forall r$, where $s \leq r \leq u$, $X^{s,r}$ is frequent.
(1) If $s > p$, $X^{s-1,s}$ is not frequent.

The *FCP* of an itemset X is the common exhibition period of the items belonging to X such that the itemset X starts and continues to be frequent in this period.

Definition 10. An *maximal frequent common exhibition period* (*MFCP*) of an itemset X is the *FCP* of the itemset X which is not covered by any other *FCPs* of the itemset X.

Definition 11. The association rule $(Y \Longrightarrow Z)^{MFCP(X)}$ derived from the frequent temporal itemset $X^{MFCP(X)}$ is called a *precise general temporal association rule* if $Y \subset X$ and $Z = X - Y$.

Table I. A List of the Symbols Used

Symbol	Description
P_p	p-th partition of the database according to the time granularity
$\|P_p\|$	number of transactions in P_p
$T^{p,q}$	partial database formed by the continuous region from P_p to P_q
$\|T^{p,q}\|$	number of transactions in $T^{p,q}$
$X^{p,q}$	a temporal itemset existing from partition P_p to P_q
$MCP(X)$	maximal common exhibition period of the itemset X
$(X \Longrightarrow Y)^{MCP(X \cup Y)}$	general temporal association rule
$MFCP(X)$	maximal frequent common exhibition period of the itemset X
$(X \Longrightarrow Y)^{MFCP(X \cup Y)}$	precise general temporal association rule
min_supp	minimum support required
min_conf	minimum confidence required
TI	maximal temporal itemset
SI	temporal sub-itemset

Definition 12. The *confidence* of the precise general temporal association rule $(Y \Longrightarrow Z)^{MFCP(Y \cup Z)}$ is defined as:

$$conf((Y \Longrightarrow Z)^{MFCP(Y \cup Z)}) = \frac{supp((Y \bigcup Z)^{MFCP(Y \cup Z)})}{supp(Y^{MFCP(Y \cup Z)})}.$$

Similarly, the problem of mining precise general temporal association can be decomposed into two steps: (1) Generating all frequent temporal itemsets with their relative supports and their MFCPs; (2) Deriving all precise general temporal association rules that satisfy *min_conf* from these frequent temporal itemsets. Note that once the frequent temporal itemsets with their supports are obtained, deriving the precise general temporal association rules is straightforward. Therefore, in the rest of this article, we concentrate our discussion on the algorithms for mining frequent temporal itemsets.

For better readability, a list of the symbols used is given in Table I.

2.2 Related Works

There are some prior works exploring the problem of finding temporal association rules, that is, discovering association rules from a given subset of database specified by time [Ale and Rossi 2000; Bettini et al. 1998; Chen and Petr 2000; Chen et al. 1998; Harms and Deogun 2004; Jiang and Gruenwald 2006; Lee et al. 2001a; Tansel and Ayan 1998]. The temporal transaction database is divided into several partitions according to the time granularity imposed. The temporal association rule mining is to find frequent association rules within partitions. Among these, Lee et al. [2001a] focused on mining general temporal association rules in a publication database, where the starting exhibition times of items are allowed to be different but their ending exhibition times are required to be the same. However, these works are still not able to mine general temporal association rules in the database in which both the starting exhibition time and the ending exhibition time of items are different. The work in Chang et al. [2002] devised a model of mining general temporal association rules and proposed an algorithm *SPF* to deal with this problem. In this paper, we present a new algorithm, *Twain*, to discover precise general temporal association rules.

2.3 *Apriori*^{IP}

Conventional Apriori-based algorithms for mining association rules are all based on the downward closure property to generate candidate k-itemsets from frequent $(k-1)$-itemsets where $k > 2$. However, as pointed out in Section 1, the downward closure is no longer valid in the new model of mining general temporal association rules defined in Section 2.1. Consequently, it is necessary to develop new solutions to this problem. The ordinary approach is to employ the conventional mining algorithms to mine association rules separately from all kinds of combinatorial subdatabases, for example, $T^{1,1}$, $T^{1,2}$, $T^{1,3}$, $T^{1,4}$, $T^{2,2}$, $T^{2,3}$, $T^{2,4}$, $T^{3,3}$, $T^{3,4}$, and $T^{4,4}$ in Figure 1. However, this approach is too costly to be practically used. Note that for a database which consists of n partitions, the total number of all kinds of combinatorial subdatabases is equal to $n*(n+1)/2$, meaning that $n*(n+1)/2$ runs of the mining algorithm are needed. Thus, we forsook this approach immediately. Recall that, in Property 1, all temporal subitemsets of a frequent maximal temporal itemset are also frequent. In light of this, we first transform each item to the temporal 1-itemsets with all possible exhibition periods. Then, based on the anti-monotone Apriori-like heuristic, we can generate candidate temporal k-itemsets from frequent temporal $(k-1)$-itemsets efficiently until no candidate temporal itemset can be generated any more. This approach is referred to as *Apriori*^{IP} and will be implemented and comparatively analyzed with *Twain* proposed in this paper.

Example 2.2. According to the database in Figure 1, we have 6 items A, B, C, D, E and F with their exhibition periods [1, 4], [2, 4], [1, 3], [1, 2], [3, 4] and [1, 3]. Suppose that the minimum support and confidence required are 30% and 75%, respectively, that is, $min\_supp = 30\%$ and $min\_conf = 75\%$. *Apriori*^{IP} first transforms them into temporal 1-itemsets with all possible exhibition periods, for example, B is transformed into $B^{2,2}$, $B^{2,3}$, $B^{2,4}$, $B^{3,3}$, $B^{3,4}$ and $B^{4,4}$. Then, *Apriori*^{IP} uses the frequent ones to create candidate temporal 2-itemsets. In this example, we can eliminate $A^{2,2}$, $A^{2,3}$, $A^{3,3}$ and $C^{3,3}$ as well as the consequent candidate temporal k-itemset $(k > 1)$ formed by these temporal 1-itemsets. After *Apriori*^{IP} generates candidate temporal k-itemsets from frequent temporal $(k-1)$-itemsets until no candidate temporal itemset can be generated any more, the resulting frequent patterns are $\{AD^{1,1}, AF^{1,1}, CF^{1,1}, CF^{1,2}, CF^{1,3}, CF^{2,2}, CF^{2,3}, CF^{3,3}, BC^{2,2}, BD^{2,2}, CD^{2,2}, EF^{3,3}, AB^{4,4}, BCD^{2,2}\}$ by scanning the database. Since $CF^{1,1}$, $CF^{1,2}$, $CF^{2,2}$, $CF^{2,3}$ and $CF^{3,3}$ are covered by $CF^{1,3}$. The final frequent patterns are $\{AD^{1,1}, AF^{1,1}, CF^{1,3}, BC^{2,2}, BD^{2,2}, CD^{2,2}, EF^{3,3}, AB^{4,4}, BCD^{2,2}\}$. As will be shown later, all of the patterns discovered by *Apriori*^{IP} can be covered by the proposed algorithm *Twain*.

Note that although such a pruning technique can reduce the search space for *Apriori*^{IP}, a linearly growing number of the items in the database still implies an exponentially growing number of temporal itemsets to be considered. In fact, as will be validated by our experimental results later, the increase of the number of the candidates often causes a huge increase of execution time and a drastic performance degradation.

2.4 *SPF*

With the preliminaries described in Section 2.1, we describe the algorithm, *SPF*, for mining general temporal association rules reported in Chang et al. [2002]. The major challenge of mining general temporal association rules is that the exhibition periods of the items in the transaction database are allowed to be different from one another. In such a circumstance, it is very difficult to efficiently generate candidate itemsets since the downward closure property is no longer valid as explained in Section 1.

In essence, *SPF* consists of two major procedures, that is, *Segmentation* (abbreviated as *ProcSG*) and *Progressively Filtering* (abbreviated as *ProcPF)*. In light of the exhibition period of each item, *SPF* employs *ProcSG* to segment the database into subdatabases in such a way that items in each subdatabase will have *either* the common *Start time or* the common *End time*. For each subdatabase, *SPF* utilizes *ProcPF* to progressively filter candidate 2-itemsets with cumulative filtering thresholds from one partition to another. Specifically, *ProcPF* generates all 2-itemsets and counts their occurrences in the first partition. For those 2-itemsets whose numbers of occurrences are not smaller than the filtering threshold (i.e., $min\_supp * |P_1|$), they are viewed as candidate 2-itemsets and will be brought to the next partition for further processing. Then, *ProcPF* will generate new 2-itemsets in the second partition and count their occurrences as well. Meanwhile, for those candidate 2-itemsets brought from the previous partition, their numbers of occurrences will be cumulated from the previous partition to the current one. Note that the filtering threshold (i.e., $min\_supp * (|P_1| + |P_2|)$) for these itemsets will also be cumulated. Similarly, those 2-itemsets whose numbers of occurrences are not smaller than their corresponding filtering thresholds will be brought to the next partition for further processing until there is no partition to be processed any more. After all subdatabases are processed, *SPF* unions all candidate 2-itemsets generated in each subdatabase. Since infrequent 2-itemsets will be filtered out in the early processed partitions, the resulting candidate 2-itemsets will be very close to the frequent 2-itemsets. This feature allows us of adopting the scan reduction technique [Park et al. 1997] by generating all candidate k-itemsets ($k > 2$) from candidate 2-itemsets directly. After all candidate itemsets are generated, they are transformed to TIs, and the corresponding SIs are generated based on these TIs. Finally, the frequent TIs and SIs with their supports can be obtained by scanning the whole database once.

Example 2.3. The operation of *SPF* can be best understood by an illustrative example as shown in Figure 1. Suppose that the minimum support and confidence required are 30% and 75%, respectively. *SPF* first segments the database into several sub-databases by *ProcSG*. In our example, the transaction database, $T^{1,4}$, is segmented into two sub-databases, $T^{1,2}$ and $T^{3,4}$as shown in Figure 2. The scanning direction of $T^{1,2}$ is from the left to the right whereas the scanning direction of $T^{3,4}$ is from the right to the left.

After the database is segmented into subdatabases where the items in each subdatabase have either the same starting or ending partition., *SPF* employs *ProcPF* to progressively filter the candidate 2-itemsets in each of these two

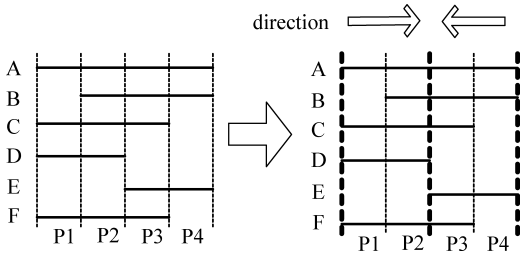
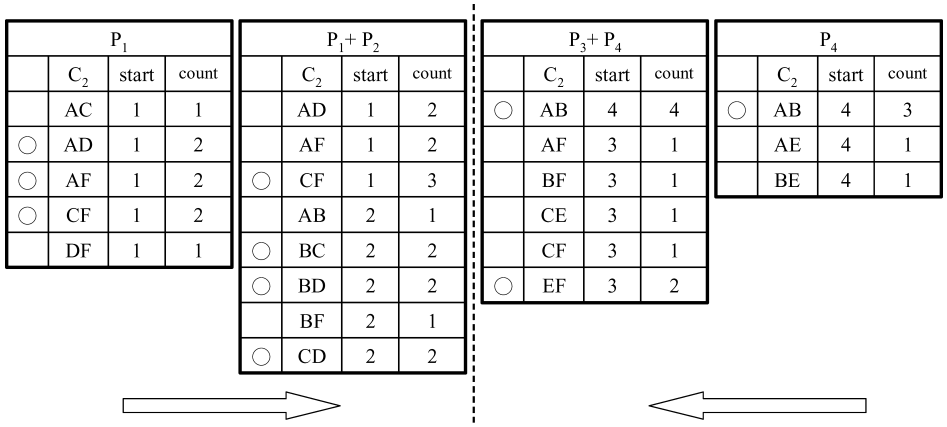


Fig. 2. Segmenting the illustrative database.



P_1

	C_2	start	count
	AC	1	1
◯	AD	1	2
◯	AF	1	2
◯	CF	1	2
	DF	1	1

$P_1 + P_2$

	C_2	start	count
	AD	1	2
	AF	1	2
◯	CF	1	3
	AB	2	1
◯	BC	2	2
◯	BD	2	2
	BF	2	1
◯	CD	2	2

$P_3 + P_4$

	C_2	start	count
◯	AB	4	4
	AF	3	1
	BF	3	1
	CE	3	1
	CF	3	1
◯	EF	3	2

P_4

	C_2	start	count
◯	AB	4	3
	AE	4	1
	BE	4	1

After first database scan, we have :

$C_2 = \{AB, BC, BD, CD, CF, EF\}$ $C_3 = \{BCD\}$.

$TI = \{AB^{2,4}, BC^{2,3}, BD^{2,2}, CD^{1,2}, CF^{1,3}, EF^{3,3}, BCD^{2,2}\}$.

$SI = \{A^{2,4}, B^{2,2}, B^{2,3}, B^{2,4}, C^{1,2}, C^{1,3}, C^{2,2}, C^{2,3}, D^{1,2}, D^{2,2}, E^{3,3}, F^{1,3}, F^{3,3}, BC^{2,2}, BD^{2,2}, CD^{2,2}\}$

After the second scan, we have :

Frequent $TI = \{AB^{2,4}, BD^{2,2}, CF^{1,3}, EF^{3,3}, BCD^{2,2}\}$.

Fig. 3. Progressively Filtering candidate 2-itemsets in each subdatabase.

sub-databases. The execution of *ProcPF* in subdatabase $T^{1,2}$ is shown in the upper-left part of Figure 3. Five 2-itemsets are first considered in the partition P_1, and their numbers of occurrences and the partition in which they are first considered (i.e., P_1) are also recorded. The corresponding filtering threshold for each of them is equal to $2 = \lceil 4 * 0.3 \rceil$ since there are four transactions in P_1. As a result, only the 2-itemsets *AD*, *AF* and *CF* are viewed as the candidate 2-itemsets and brought to the next partition for further processing. In the partition P_2, the 2-itemsets *AC*, *BC*, *BD*, *BF* and *CD* are newly generated, and their numbers of occurrences and the partition in which they are first considered (i.e., P_2) are also recorded. Since *AD*, *AF* and *CF* are brought from the previous partition, their numbers of occurrences are cumulated. So are the corresponding filtering thresholds for them (i.e., $3 = \lceil 0.3 * (4 + 4) \rceil$). Thus, only the 2-itemsets *CF*, *BC*, *BD* and *CD* are viewed as candidate 2-itemsets after processing P_2. The execution of *ProcPF* in the subdatabase $T^{3,4}$ works similarly

and is shown in the upper-right part of Figure 3. Note that the scanning direction in the subdatabase $T^{3,4}$ is from the right to the left, that is, from P_4 to P_3. After scanning the subdatabases $T^{1,2}$ and $T^{3,4}$, the resulting candidate set C_2 becomes {*AB*, *BC*, *BD*, *CD*, *CF*, *EF*}. Using the scan reduction technique, we generate all candidate k-itemsets from candidate $(k-1)$-itemsets where $k > 2$. In our case, only one candidate 3-itemset *BCD* is generated. Then, these candidates are transformed to the TIs (i.e., {$AB^{2,4}$, $BC^{2,3}$, $BD^{2,3}$, $CD^{1,2}$, $CF^{1,3}$, $EF^{3,3}$, $BCD^{2,2}$}), and the corresponding SIs (i.e., {$A^{2,4}$, $B^{2,2}$, $B^{2,3}$, $B^{2,4}$, $C^{1,2}$, $C^{1,3}$, $C^{2,2}$, $C^{2,3}$, $D^{1,2}$, $D^{2,2}$, $E^{3,3}$, $F^{1,3}$, $F^{3,3}$, $BC^{2,2}$, $BD^{2,2}$, $CD^{2,2}$}) are generated as shown in the bottom of Figure 3. By scanning the entire database again, we can have the relative supports of all temporal itemsets in $TI \cup SI$ and then determine the frequent TIs. From the frequent TIs determined (i.e., {$AB^{2,4}$, $BD^{2,2}$, $CF^{1,3}$, $EF^{3,3}$, $BCD^{2,2}$}), we can thus derive all frequent general temporal association rules as shown in Example 1.2.

3. *TWAIN* FOR PRECISE GENERAL TEMPORAL ASSOCIATION

To give more precise frequent exhibition periods of frequent temporal itemsets, we present an efficient algorithm, *Twain*, for mining precise general temporal association rules in this section. We give the detailed description of algorithm *Twain* in Section 3.1. Then, an illustrative example of *Twain* is shown in Section 3.2. The correctness of *Twain* is proved in Section 3.3. Finally, we show the incremental ability of *Twain* in Section 3.4.

3.1 Algorithm *Twain*

In order to deal with the items having different exhibition periods on transaction databases and to discover precise general temporal association rules, we propose an efficient algorithm *Twain*, which stands for *TWo end AssocIation miNer*. The transaction database is first divided into partitions according to the time granularity imposed as shown in Figure 1. Each partition of the database contains several transactions. The basic idea of *Twain* is to employ a filtering threshold from one partition to another of the database and utilize *Start time* and *End time* of each itemset to early prune away unqualified candidate 2-itemsets. Explicitly, *Twain* generates candidate 2-itemsets with their *MFCPs* while processing each partition of the database. It is noted that two types of 2-itemsets are possible to be generated when *Twain* processes each partition.

(1) *Type*-1. The itemsets that were not in the candidate set of the previous partition but are newly generated by taking the current partition into consideration.
(2) *Type*-2. The itemsets that conformed to the *End time* constraint and were carried from the pervious partitions.

The *End time* constraint means that the *End time* of *MCP* of an itemset must be larger than the current partition. The way how *Twain* generates these candidate 2-itemsets is explained as follows.

Twain is designed to progressively filter out infrequent candidate 2-itemsets from one partition to another. Specifically, it generates all possible combination

of 2-itemsets in each transaction and counts their occurrence frequencies in the current partition. These itemsets are viewed as Type-1 itemsets. For those 2-itemsets whose occurrence frequencies are not smaller than the filtering threshold (i.e., $min\_supp * |P_i|$), they are regarded as candidate 2-itemsets. Among these candidate 2-itemsets, if their *End times* of *MCPs* are larger than the current partition, they will be brought to the next partition for further processing. Otherwise, they are left in the current partition and are regarded as frequent 2-itemsets. Specifically, those candidate 2-itemsets that are brought from the previous partition are regarded as Type-2 itemsets, and their occurrence frequencies will be cumulated in the current partition. Note that the filtering threshold (i.e., $min\_supp * (|P_i| + |P_{i+1}|)$) for Type-2 itemsets will also be cumulated. Similar to the operations in the previous partition, the itemsets, including both Type-1 and Type-2 itemsets, whose cumulative occurrence frequencies are not smaller than their corresponding filtering thresholds are regarded as candidate 2-itemsets. Among them, the itemsets whose *End times* of *MCPs* are larger than the current partition are brought to the next partition for further processing. Otherwise, they are left in the current partition and are regarded as frequent 2-itemsets.

Twain continuously generates candidate 2-itemsets in the current partition and examines the candidate 2-itemsets coming from the previous partitions until there is no partition to be processed any more. After the examination of every partition in the database, *Twain* generates directly a set of frequent 2-itemsets which contain the itemsets left in each partition and the itemsets passing through some partitions. These itemsets are all qualified by the relative filtering threshold and their *MFCPs* are recorded as they are generated. Then, by using the set of frequent 2-itemsets, *Twain* generates all candidate k-itemsets ($k > 2$) from $(k-1)$-itemsets at the same time. Consequently, *Twain* can employ the scan reduction technique [Park et al. 1997] to compute the occurrence frequencies of all candidate k-itemsets by scanning the whole database once.

The main procedure to progressively filter out infrequent 2-itemsets is shown in Procedure *ProcPM* as follows.

Procedure ProcPM(n, min_supp**)**

```
Begin
1.  FI = ∅; PS = ∅; CL = ∅;
2.  for (h = 1 to n) {
3.    PS = CL; CL = ∅;
4.    for (each 2-itemset X2 in Ph) {
5.      if (X2 ∉ PS) {
6.        X2.count = NPh(X2);
7.        X2.start = h; X2.end = h;
8.        if (X2.count ≥ min_supp * |Ph|)
9.          if (MCP(X2).end < h + 1)
10.           FI = FI ∪ X2;
11.         else
12.           CL = CL ∪ X2;
13.   } else {
```

14. $X_2.count = X_2.count + N_{P_h}(X_2)$;
15. **if** $(X_2.count < \lceil min\_supp * \sum_{m=X_2.start,h} |P_m| \rceil)$
16. $X_2.count = X_2.count - N_{P_h}(X_2)$; $FI = FI \cup X_2$;
17. **else if** $(MCP(X_2).end < h + 1)$
18. $X_2.end = h$; $FI = FI \cup X_2$;
19. **else**
20. $X_2.end = h$; $CL = CL \cup X_2$;
21. $PS = PS - X_2$;
22. }**end if**
23. }**end for**
24. **for** (each 2-itemset X_2 in PS) {
25. **if** $(X_2.count < \lceil min\_supp * \sum_{m=X_2.start,h} |P_m| \rceil)$
26. $FI = FI \cup X_2$;
27. **else if** $(MCP(X_2).end < h + 1)$
28. $X_2.end = h$; $FI = FI \cup X_2$;
29. **else**
30. $X_2.end = h$; $CL = CL \cup X_2$;
31. $PS = PS - X_2$;
32. }**end for**
33.}**end for**
34.**return** FI;
End

As shown in *ProcPM*, operations in line 1 initialize the set of frequent 2-itemset FI, the progressive screen PS, and the check list to be carried to the next partition CL. The **for** loop, from line 2 to line 33, finds out all frequent 2-itemsets with their occurrence frequencies, *Start times*, and *End times* in each partition, and employs the corresponding filtering threshold (i.e., $min\_supp * \sum_{m=X.start,h} |P_m|$) to filter out infrequent ones. Among these lines, the **for** loop from line 4 to line 23 checks every possible combination of 2-itemset in each transaction, and the **for** loop from line 24 to line 32 examines the rest itemsets in PS that do not appear in this partition. Note that $N_{P_h}(X_2)$ in line 6 means the number transactions containing X_2 in partition P_h. After *ProcPM* processes all partitions, the cumulative set of frequent 2-itemsets, FI, is returned in line 34.

By applying *ProcPM*, *Twain* is able to find all frequent 2-itemsets and uses them to generate candidate k-itemsets ($k > 2$) from frequent 2-itemsets. The detailed description of algorithm *Twain* is shown as follows.

Algorithm *Twain*(n, min_supp)

Input : number of partitions in the database, n
user defined minimum support, min_supp
Output : all frequent temporal itemsets with their *MFCP*s
Begin
1. $C_2 = \emptyset$; $CS = \emptyset$;
2. $C_2 = ProcPM(n, min\_supp)$;
3. $k = 2$;
4. **while** $(C_k \neq \emptyset)$ {
5. $C_{k+1} = C_k \star C_k$;
6. $CS = CS \cup C_{k+1}$;

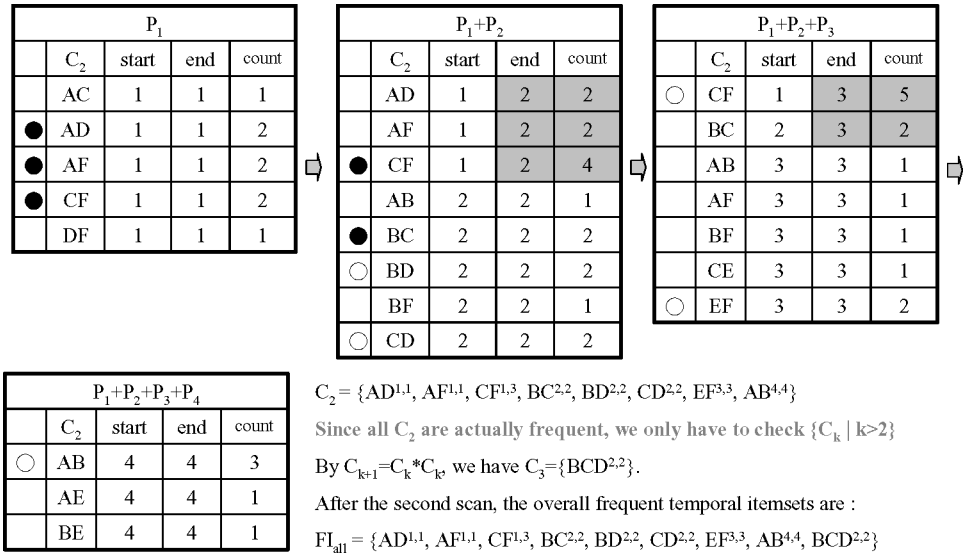


Fig. 4. Progressively Filtering candidate 2-itemsets by *ProcPM*.

7. $k = k + 1$;
8. }**end while**
9. scan database once to find frequent k-itemsets ($k > 2$) from CS;

End

As listed in algorithm *Twain*, line 1 initializes all temporary sets to be empty. Line 2 employs *ProcPM* to find frequent 2-itemsets. Then, the **while** loop from line 4 to line 8 generates all candidate k-itemsets from $(k-1)$-itemsets. Finally, the database is scanned once to determine all frequent general temporal itemsets in line 9 using the scan reduction technique [Park et al. 1997].

3.2 On Detailed Operations of *Twain*

An illustrative example is given in this section to explain the detailed operations of *Twain*.

Example 3.1. The example database is the same as the one shown in Figure 1. Suppose that the minimum support and confidence required are 30% and 75%, that is, $min\_supp = 30\%$ and $min\_conf = 75\%$. Note that the shadowed parts in Figure 4 are the values being updated in that partition.

As shown in Figure 4, *Twain* employs *ProcPM* to progressively filter out infrequent candidate 2-itemsets. Initially, five 2-itemsets, *AC*, *AD*, *AF*, *CF* and *DF*, are first considered in the partition P_1. According to the descriptions in section 3.1, they are all Type-1 itemsets. The occurrence frequencies of these itemsets are recorded. It is noted that the *Start time* and the *End time* of the itemset that is first examined are set to be the current partition. Therefore, the

Start times and the *End times* of these itemsets are all set to be 1 (i.e., P_1). The corresponding filtering threshold for each itemset is equal to $\lceil min\_supp * |P_1| \rceil = \lceil 0.3 * 4 \rceil = 2$ since there are four transactions in P_1. Consequently, only the 2-itemsets *AD*, *AF* and *CF* are regarded as the candidate 2-itemsets, as marked by circles in Figure 4. Because all of their *End times* of *MCP* are larger than the current partition (i.e., P_1), they are brought to the next partition for further processing, and are marked by black circles in Figure 4. In the partition P_2, since *AD*, *AF* and *CF* are carried from the previous partition, they are viewed as Type-2 itemsets and their occurrence frequencies are cumulated. So are the corresponding filtering thresholds for them (i.e.,$\lceil min\_supp * (|P_1| + |P_2|) \rceil = \lceil 0.3 * (4 + 4) \rceil = 3$). Note that their *End times* are updated as P_2. On the other hand, the 2-itemsets *AB*, *BC*, *BD*, *BF* and *CD* are newly generated in this partition. Their occurrence frequencies and the partition in which they are considered (i.e., P_2) are recorded. The corresponding filtering thresholds for them are equal to $\lceil 0.3 * 4 \rceil = 2$. Thus, only the 2-itemsets *CF*, *BC*, *BD* and *CD* are regarded as candidate 2-itemsets after processing P_2, as marked by circles in Figure 4. Since *CF*s and *BC*s *End times* of *MCP* are larger than the current partition (i.e., P_2), itemsets *CF* and *BC* are carried to the next partition for further processing, as marked by black circles in Figure 4. In contrast, itemsets *BD* and *CD* are left in the set of frequent itemset, *FI*, and do not have to be carried to the next partition. After *ProcPM* scans the remaining partitions P_3 and P_4, the resulting set of frequent 2-itemsets, *FI*, becomes $\{AD^{1,1}, AF^{1,1}, CF^{1,3}, BC^{2,2}, BD^{2,2}, CD^{2,2}, EF^{3,3}, AB^{4,4}\}$.

Then, we generate all candidate k-itemsets from candidate $(k-1)$-itemsets for $k > 2$. In this example, only one candidate 3-itemset $BCD^{2,2}$ is generated. Finally, since all C_2 are actually frequent itemsets, we only have to check the occurrence frequencies of $\{C_k | k > 2\}$. We use the scan reduction technique by scanning the entire database again to compute the relative supports of all these temporal candidates k-itemsets ($k > 2$). The resulting frequent itemsets together with frequent 2-itemsets already in *FI* are shown in the bottom of Figure 4. Therefore, the overall frequent itemsets $\{AD^{1,1}, AF^{1,1}, CF^{1,3}, BC^{2,2}, BD^{2,2}, CD^{2,2}, EF^{3,3}, AB^{4,4}, BCD^{2,2}\}$ are obtained. We can thus derive all precise general temporal association rules as follows.

(1) $(D \Longrightarrow A)^{1,1}$ with $supp(AD^{1,1}) = \frac{2}{4}$ and $conf((D \Longrightarrow A)^{1,1}) = \frac{supp(AD^{1,1})}{supp(D^{1,1})} = \frac{2}{2}$

(2) $(D \Longrightarrow B)^{2,2}$ with $supp(BD^{2,2}) = \frac{2}{4}$ and $conf((D \Longrightarrow B)^{2,2}) = \frac{supp(BD^{2,2})}{supp(D^{2,2})} = \frac{2}{2}$

(3) $(D \Longrightarrow C)^{2,2}$ with $suppp(CD^{2,2}) = \frac{2}{4}$ and $conf((D \Longrightarrow C)^{2,2}) = \frac{supp(CD^{2,2})}{supp(D^{2,2})} = \frac{2}{2}$

(4) $(C \Longrightarrow F)^{1,3}$ with $supp(CF^{1,3}) = \frac{5}{12}$ and $conf((C \Longrightarrow F)^{1,3}) = \frac{supp(CF^{1,3})}{supp(C^{1,3})} = \frac{5}{6}$

(5) $(E \Longrightarrow F)^{3,3}$ with $supp(EF^{3,3}) = \frac{2}{4}$ and $conf((E \Longrightarrow F)^{3,3}) = \frac{supp(EF^{3,3})}{supp(E^{3,3})} = \frac{2}{2}$

(6) $(A \Longrightarrow B)^{4,4}$ with $supp(AB^{4,4}) = \frac{3}{4}$ and $conf((A \Longrightarrow B)^{4,4}) = \frac{supp(AB^{4,4})}{supp(A^{4,4})} = \frac{3}{3}$

(7) $(D \Longrightarrow BC)^{2,2}$ with $supp(BCD^{2,2}) = \frac{2}{4}$ and
$conf((D \Longrightarrow BC)^{2,2}) = \frac{supp(BCD^{2,2})}{supp(D^{2,2})} = \frac{2}{2}$

(8) $(BC \Longrightarrow D)^{2,2}$ with $supp(BCD^{2,2}) = \frac{2}{4}$ and
$conf((BC \Longrightarrow D)^{2,2}) = \frac{supp(BCD^{2,2})}{supp(BC^{2,2})} = \frac{2}{2}$

(9) $(BD \Longrightarrow C)^{2,2}$ with $supp(BCD^{2,2}) = \frac{2}{4}$ and
$conf((BD \Longrightarrow C)^{2,2}) = \frac{supp(BCD^{2,2})}{supp(BD^{2,2})} = \frac{2}{2}$

(10) $(CD \Longrightarrow B)^{2,2}$ with $supp(BCD^{2,2}) = \frac{2}{4}$ and
$conf((CD \Longrightarrow B)^{2,2}) = \frac{supp(BCD^{2,2})}{supp(CD^{2,2})} = \frac{2}{2}$.

3.3 Correctness of Algorithm *Twain*

As mentioned in the previous sections, algorithm *SPF* has two drawbacks, that is, (1) *overestimating the common exhibition period of a frequent itemset* and (2) *underestimating the occurrence frequency of a frequent itemset*. As shown in Example 3.1, *Twain* is able to identify the *MFCP* $[4, 4]$ of AB instead of the whole range of *MCP* $[2, 4]$ generated by *SPF*. Thus, *Twain* can defeat the first limitation. Moreover, *Twain* is able to conquer the second limitation and identify the frequent itemset $AD^{1,1}$ while *SPF* cannot find such kind of frequent itemsets. We show that how *Twain* overcomes the drawbacks of *SPF* by the following lamma:

LEMMA 1. *FCPs describe more precise frequent periods of an itemset X than the MCP of X.*

PROOF. In accordance with Definition 9 in Section 2.1, an *FCP* of an itemset X, whose *MCP* is $[p, q]$, is denoted by $[s, u]$ $(p \leq s \leq u \leq q)$ such that the following conditions are followed: (1) $\forall\, r$, where $s \leq r \leq u$, $X^{s,r}$ is frequent. (2) If $s > p$, $X^{s-1,s}$ is not frequent. That is, an *FCP* is a subperiod of an *MCP* and all *FCPs* in an *MCP* identify the frequent subperiods in which the itemset X starts and continues to be frequent. Therefore, *FCPs* exclude the periods in which the itemset X is not frequent in its *MCP*. This defeats the overestimating problem of *SPF*. In addition, each frequent period is included in one of *FCPs.* As a result, the underestimating problem of *SPF* can be solved. □

THEOREM 1. *Twain overcomes the drawbacks of SPF by using MFCPs of itemsets.*

PROOF. In accordance with Definition 10 in Section 2.2, an *MFCP* of an itemset X is the *FCP* of the itemset X that is not covered by any other *FCPs* of the itemset X. That is, an *MFCP* is the longest period among all *FCPs* of the itemset X. Thus, *MFCP* describes more precise frequent period of an itemset than *MCP* as well. Accordingly, by using *MFCPs* of itemsets, *Twain* can overcome the drawbacks of *SPF*. □

Therefore, both drawbacks of algorithm *SPF* can be successfully overcome by taking the concept of *MFCP* into consideration. In addition, $Apriori^{IP}$ transforms each item to the temporal 1-itemsets with all possible exhibition periods and generates temporal k-itemsets by frequent temporal $(k-1)$-itemsets. The

frequent patterns found by *Apriori*^{IP} must be the combination of frequent 2-itemsets with all possible exhibition periods. Moreover, *Twain* is able to find frequent 2-itemsets with their *MFCPs* and all frequent exhibition periods will be covered by *MFCPs*. Since *Twain* uses these frequent 2-itemsets to generate temporal k-itemsets ($k > 2$), all the patterns discovered by *Apriori*^{IP} can be covered by *Twain*.

We then prove the correctness of *Twain* by showing that $X_k^{s,u}$ will be produced by *Twain* if and only if $[s, u]$ is the *MFCP* of X_k.

LEMMA 2. *Assume that* $[i, j]$ *is the MFCP of the temporal itemset* $X_2^{i,j}$ $(1 \leq i \leq j \leq n)$ *in* $T^{1,n}$. *If the* $MCP(X_2).end > j$ *and* $j < n$, *then* $supp(X_2^{i,j+1}) < min\_supp$.

PROOF. We prove the lemma by contradiction. Assume that $supp(X_2^{i,j+1}) \geq min\_supp$. Based on the definition of *MFCP*, an *MFCP* is also an *FCP*. That is, $[i, j]$ is an *FCP* and the following conditions are held.

(1) $\forall k$, where $i \leq k \leq j$, $supp(X_2^{i,k}) \geq min\_supp$.
(2) If $i > 1$, then $supp(X_2^{i-1,i}) < min\_supp$.

Therefore, according to the above conditions and $supp(X_2^{i,j+1}) \geq min\_supp$, it follows from the definition of *FCP* that $[i, j + 1]$ is also an *FCP*. Thus, the *MFCP* of the temporal itemset $X_2^{i,j}$ is $[i, j + 1]$ rather than $[i, j]$. This incurs a contradiction and thus proves the lemma. □

LEMMA 3. *After ProcPM scans all partitions in* $T^{1,n}$, *the 2-itemset* $X_2^{i,j}$ $(1 \leq i \leq j \leq n)$ *will be in FI if all of the following conditions are held:*

(1) $\forall k$, *where* $i \leq k \leq j$, $supp(X_2^{i,k}) \geq min\_supp$.
(2) *If* $i > 1$, *then* $supp(X_2^{i-1,i}) < min\_supp$.
(3) *If* $MCP(X_2).end > j$ *and* $j < n$, *then* $supp(X_2^{i,j+1}) < min\_supp$.

PROOF. In accordance with the procedure *ProcPM*, the itemset $X_2^{i,j}$ is inserted into *FI* due to one of the following three scenarios:

(a) X_2 is a Type-1 candidate itemset in P_j, which is a newly generated 2-itemset appearing in P_j, and is inserted in *FI* after *ProcPM* processes P_j. In accordance with line 8 of *ProcPM*, $supp(X_2^{j,j}) \geq min\_supp$. The case $i = j$ obeys condition (1). Since X_2 is not carried from the previous partition, condition (2) is held. Finally, condition (3) is also held according to line 9 of *ProcPM*.

(b) X_2 is a Type-2 candidate itemset in P_j, which was already a candidate 2-itemset in *PS*, and is inserted in *FI* after *ProcPM* processes P_j. That is, X_2 was once inserted into *CL* as a new candidate 2-itemset in some partition, say in P_a $(a < j)$, and remains in *PS* during the period $[a, j-1]$. In accordance with the **for** loop from line 24 to line 32 of *ProcPM*, we have $supp(X_2^{a,t}) \geq min\_supp$ for each t $(a \leq t \leq j-1)$. Note that in P_a, X_2 is a Type-1 candidate itemset. The case $i = a$ validates conditions (1) and (2) except $supp(X_2^{i,j}) \geq min\_supp$. If X_2 is inserted into *FI* due to line 18 or line 28 of *ProcPM*, then $supp(X_2^{i,j}) \geq min\_supp$.

Therefore, condition (1) is held. Consequently, according to line 17 or line 27 of *ProcPM*, condition (3) is held.

(c) X_2 is a Type-2 candidate itemset in P_{j+1}, which was already a candidate 2-itemset in *PS*, and is inserted in *FI* after *ProcPM* processes P_{j+1}. That is, X_2 was once inserted into *CL* as a new candidate 2-itemset in some partition, say in P_a ($a < j+1$), and remains in *PS* during the period $[a, j+1]$. In accordance with the **for** loop from line 24 to line 32 of *ProcPM*, we have $supp(X_2^{a,t}) \geq min\_supp$ for each t ($a \leq t \leq j$). Note that in P_a, X_2 is a Type-1 candidate itemset. The case $i = a$ validates conditions (1) and (2). Then, X_2 is inserted into *FI* due to either line 16 or line 26 of *ProcPM*. Therefore, in accordance with line 15 or line 25 of *ProcPM*, condition (3) is held.

Based on the above scenarios, Lemma 3 is proved. □

THEOREM 2. *If $[i, j]$ is the MFCP of the temporal itemset $X_2^{i,j}$ ($1 \leq i \leq j \leq n$) in $T^{1,n}$, then $X_2^{i,j}$ will be in FI returned by ProcPM.*

PROOF. Note that $[i, j]$ is the *MFCP* of the temporal itemset $X_2^{i,j}$ ($1 \leq i \leq j \leq n$) in $T^{1,n}$. In accordance with Lemma 2 and the definition of *FCP*, the following conditions are held:

(1) $\forall\, k$, where $i \leq k \leq j$, $supp(X_2^{i,k}) \geq min\_supp$.

(2) If $i > 1$, then $supp(X_2^{i-1,i}) < min\_supp$.

(3) If $MCP(X_2).end > j$ and $j < n$, then $supp(X_2^{i,j+1}) < min\_supp$.

It follows from Lemma 3 that $X_2^{i,j}$ will be in *FI* returned by *ProcPM* after *ProcPM* scans all partitions in $T^{1,n}$. □

THEOREM 3. *If the $[s, u]$ is the MFCP of temporal k-itemset $X_k^{s,u}$ ($k \geq 2$) in $T^{1,n}$, then the itemset $X_k^{s,u}$ will be generated by algorithm Twain in C_k.*

PROOF. If $X_k^{s,u}$ is frequent in $T^{1,n}$, by Property 1 in Section 2.1, every temporal sub-2-itemset $Y_2^{s,u}$ of $X_k^{s,u}$, is also frequent in $T^{1,n}$. As derived in Theorem 2, every C_2 with its *MFCP* will be generated in *FI* returned by *ProcPM*.

(a) If every temporal sub-2-itemset $Y_2^{s,u}$ of $X_k^{s,u}$ is in *FI*, X_k will be generated in the candidate k-itemset C_k with its *MFCP* by algorithm *Twain* based on the anti-monotone Apriori-like heuristic.

(b) If some of temporal sub-2-itemset $Z_2^{s,u}$ of $X_k^{s,u}$ is not in *FI*, there must be another $Z_2^{m,n}$ ($m \leq s \leq t \leq n$), where $[m, n]$ is the *MFCP* of Z_2, in *FI* based on Theorem 2. Therefore, X_k will be generated in the candidate k-itemset C_k with its *MFCP* by algorithm *Twain* too.

The above cases prove this theorem. □

THEOREM 4. *If $X_k^{s,u}$ is produced by Twain, then $[s, u]$ is the MFCP of X_k.*

PROOF. In accordance with lines 10, 16, 18, 26 and 28 of *ProcPM*, the 2-itemsets being examined must be in the ranges of their *FCPs* and will be inserted to *FI* only when their *MFCPs* are reached. Therefore, the *ProcPM* returns all frequent temporal 2-itemsets with their *MFCPs*. Then, in accordance with

the **while** loop from line 4 to line 8 of algorithm *Twain*, all the candidate k-itemsets are generated by their sub-itemsets with their corresponding *MFCPs*. After line 9 and line 10 of algorithm *Twain*, all the frequent temporal k-itemsets with their *MFCPs* are returned by *Twain*. □

Finally, we show the theorem for proving the correctness of algorithm *Twain*.

THEOREM 5. *$X_k^{s,u}$ will be produced by Twain if and only if $[s, u]$ is the MFCP of X_k.*

PROOF. Based on Theorem 2, the "if" condition is proved. In addition, Theorem 4 proves the "only if" condition. Therefore, all frequent itemsets with their *MFCPs* will be discovered by algorithm *Twain*. □

3.4 Incremental Ability of *Twain*

Twain is in essence a progressive algorithm to find precise general temporal association rules. It can also be applied to deal with the incremental database where there are new transactions arriving. As shown in Section 3.1, the data structure maintained by *ProcPM* changes its state from one partition to another to maintain the latest candidate 2-itemsets. Note that the candidate 2-itemsets generated in each partition are examined and those whose occurrence frequencies are larger than the relative supports are passed to the next partition. Therefore, in the scenario that there are new transactions arriving at the database and resulting in a new partition, *ProcPM* can pass the latest candidate 2-itemsets of the original database to the new partition. Then, *ProcPM* handles the transactions in the new partition and generates new candidate 2-itemsets in this partition. After new partitions are examined, *ProcPM* is able to update the latest candidate 2-itemsets without re-processing the old partitions in the original database. Thus *ProcPM* can incrementally generate candidate 2-itemsets. Since *ProcPM* can directly generate frequent 2-itemsets, by applying the scan reduction technique, *Twain* is able to find all frequent k-itemsets efficiently.

4. EXPERIMENTAL RESULTS

To assess the performance of *Twain*, we conducted several experiments on generating frequent temporal itemsets from synthetic databases. The simulation program is coded in C++ and the experiments are run on a computer with Pentium III 866MHz CPU and 512MB RAM. We use both synthetic datasets and a real dataset to demonstrate the behavior of *Twain*. As will be shown later, *Twain* not only generates frequent patterns with better quality, but also outperforms *Apriori*^{IP} and *SPF* in terms of execution time, I/O costs, CPU overheads, and scalability. In addition to generating more precise frequent patterns, *Twain* performs even more efficiently than *SPF* because *Twain* can generate frequent 2-itemsets directly. We describe the method used to generate synthetic databases and the source of real datasets in Section 4.1. Section 4.2 reveals the quality comparison of frequent patterns generated by *SPF* and *Twain*. The execution time of *Twain* is compared with prior algorithms in Section 4.3. Section 4.4 shows the I/O costs and CPU overheads for *Twain*. Results of scaleup

Table II. Summary of the Parameters Used

$\|T\|$	Ave. size of the transactions
$\|I\|$	Avg. size of the potentially frequent itemsets
$\|D\|$	Number of transactions in the database
N	Number of distinct items
$\|L\|$	Number of potentially frequent itemsets
$\|P\|$	Number of partitions

experiments are presented in Section 4.5. Finally, the incremental ability of *Twain* is examined in Section 4.6.

4.1 Simulation Model

We use the same scenario as in Chang et al. [2002] to generate synthetic datasets where the items in transactions are allowed to have different exhibition periods. The method to generate synthetic datasets is similar to the ones used in Agrawal and Srikant [1994] and Park et al. [1997]. However, in order to simulate various exhibition periods of the items in a realistic database, we equally divide the synthetic database into n partitions to imitate the phenomenon of the time granularity required. Then, an exhibition period $[p, q]$ $(1 \leq p \leq q \leq n)$ for each item in the synthetic database is randomly assigned. Finally, we scan the database once to remove the items that are not within their exhibition periods in each transaction. For example, the item X would be removed from the transaction XYZ in partition P_1 if the exhibition period of the item X is $[2, 4]$. In accordance with the above method, we generate several different synthetic datasets to evaluate the performance *Twain*. Each of the generated databases consists of $|D|$ transactions with $|T|$ items in average. The number of different items in each database is N. The average size of the potentially frequent itemsets is set to $|I|$, and the number of the potentially frequent itemsets is set to $|L|$. The mean correlation level between the potential frequent itemsets is set to 0.25 in our experiments. Table II provides a summary of some parameters used in our experiments. In addition, for the simplicity of presentation, we use the notation $Tx - Iy - Dz(Nm - Ln - Po)$ to represent a database in which $|T| = x$, $|I| = y$, $|D| = z$ thousands, $N = m$ thousands, $|L| = n$ thousands and $|P| = o$.

The real dataset is from KDDCUP web site. We use BMS-POS as the testing workload, which contains 515597 transactions and 1633 different items. The dataset is divided into 12 and 24 partitions, which are denoted as BOS-POS-12 and BOS-POS-24 respectively, for the performance evaluation. We appreciate that Zheng and Blue Martini Software provided the dataset in KDD 2001 [Zheng et al. 2001].

4.2 Quality of Frequent Patterns

At the beginning, the quality of frequent patterns provided by *SPF* and *Twain* is investigated. We use both synthetic datasets and real datasets as testing workloads. We generate several synthetic datasets with parameters $T10 - I4 - D100$. As shown in Figure 5, the leftmost column represents

Dataset	SPF	Twain	Same	New	Precise	Same-R(%)	New-R(%)	Precise-R(%)	Improvement
N10-L2-P12-3	360	850	45	490	315	5.29	57.65	37.06	94.71
N10-L2-P12-5	98	224	9	126	89	4.02	56.25	39.73	95.98
N10-L2-P12-7	34	67	1	33	33	1.49	49.25	49.25	98.51
N10-L2-P12-9	14	29	1	15	13	3.45	51.72	44.83	96.55
N10-L4-P12-3	189	471	3	282	186	0.64	59.87	39.49	99.36
N10-L4-P12-5	48	101	0	53	48	0.00	52.48	47.52	100.00
N10-L4-P12-7	16	37	0	21	16	0.00	56.76	43.24	100.00
N10-L4-P12-9	6	15	0	9	6	0.00	60.00	40.00	100.00
N20-L2-P12-3	175	486	15	311	160	3.09	63.99	32.92	96.91
N20-L2-P12-5	46	101	1	55	45	0.99	54.46	44.55	99.01
N20-L2-P12-7	13	30	0	17	13	0.00	56.67	43.33	100.00
N20-L2-P12-9	3	12	0	9	3	0.00	75.00	25.00	100.00
N20-L4-P24-3	84	465	2	381	82	0.43	81.94	17.63	99.57
N20-L4-P24-5	18	85	1	67	17	1.18	78.82	20.00	98.82
N20-L4-P24-7	2	23	0	21	2	0.00	91.30	8.70	100.00
N20-L4-P24-9	0	7	0	7	0	0.00	100.00	0.00	100.00
BMS-POS-12-3	90	177	24	87	66	13.56	49.15	37.29	86.44
BMS-POS-12-5	29	50	3	21	26	6.00	42.00	52.00	94.00
BMS-POS-24-3	90	211	24	121	66	11.37	57.35	31.28	88.63
BMS-POS-24-5	29	61	3	32	26	4.92	52.46	42.62	95.08

Fig. 5. Comparison of frequent patterns generated by *SPF* and *Twain*.

different datasets. The synthetic datasets are denoted by $Nm - Ln - Po - s$, where N, L and P are the same definitions in Table II and s represents the support value per thousand. On the other hand, the real datasets are represented by $BMS - POS - P - S$, where P is the number of partition and S is the support value per hundred. The second and the third columns are the number of frequent patterns generated by *SPF* and *Twain*. There are three types of patterns generated by *Twain*. The numbers are listed in the fourth, fifth, and sixth columns. The "Same" column represents the number of patterns whose items and the frequent exhibition periods are exactly the same as the patterns generated by *SPF*. The "New" column represents the number of patterns whose items are not found by *SPF*. The "Precise" column represents the number of patterns whose items are the same as some patterns discovered by *SPF* but whose frequent exhibition periods are more precise than those by *SPF*. In addition, the following three columns are the ratio(%) of the number in the fourth, fifth, and sixth columns to the total number of patterns generated by *Twain*. Finally, the last column represents the ratio(%) of the number of patterns in "New" and "Precise" columns to the total number of patterns generated by *Twain*. For interested readers, the results of the patterns obtained by *Twain* and *SPF* in this experiment are shown in the following web site: http://arbor.ee.ntu.edu.tw/~jwhuang/twain-results/.

As shown in Figure 5, *Twain* can find some frequent patterns which *SPF* is not able to discover. For some patterns generated by *SPF*, *Twain* can obtain more precise frequent exhibition periods of them. The improvement of the

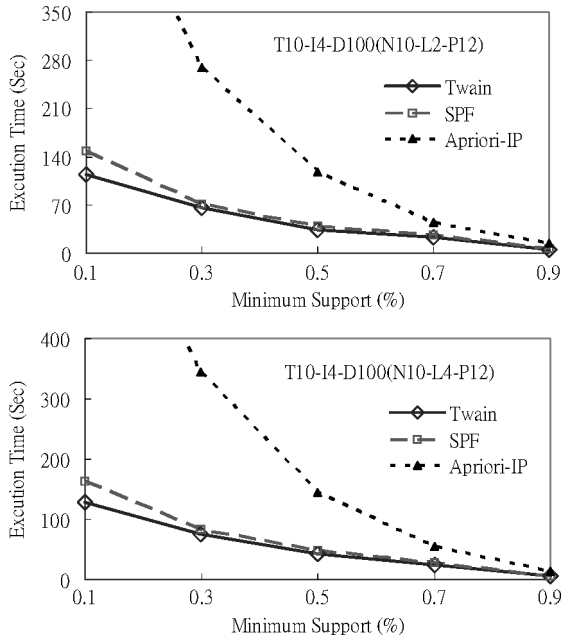


Fig. 6. The execution time under various minimum support.

quality is 97.40% in average for synthetic datasets. For real datasets, the improvement ratio reaches 89.18%. Therefore, *Twain* overcomes the drawbacks of *SPF* and discovers frequent patterns of significantly better quality.

4.3 Execution Time

Twain has shown significant improvement in the quality of the discovered patterns compared to existing methods. In fact, *Twain* also takes reasonably fewer computational steps than other algorithms. In the second experiment, several synthetic datasets are used to investigate the execution time of all algorithms by varying the minimum support. The experimental results on various datasets are shown in Figure 6 and Figure 7. Note that no matter what combination of different parameters is, *Twain* consistently outperforms *Apriori*^{IP} and *SPF* in terms of the execution time. Specifically, the execution time of *Twain* is in orders of magnitude smaller than that of *Apriori*^{IP}, and is also better than that of *SPF*. The margin even grows as the minimum support decreases.

The reason is that the number of candidate itemsets generated by *Apriori*^{IP} increases exponentially as the number of items or the number of partitions increases. In contrast, the number of candidates generated by *SPF* and *Twain* is in proportion to the number of items or the number of partitions in the database, which is about constant. Additionally, *Twain* can generate frequent 2-itemsets directly, whose number is much less than the number of candidate 2-itemsets generated by the other two algorithms. Furthermore, *Apriori*^{IP} needs to scan the database multiple times to determine frequent k-itemsets. However, by the technique of scan reduction, *Twain* only needs to scan the database twice.

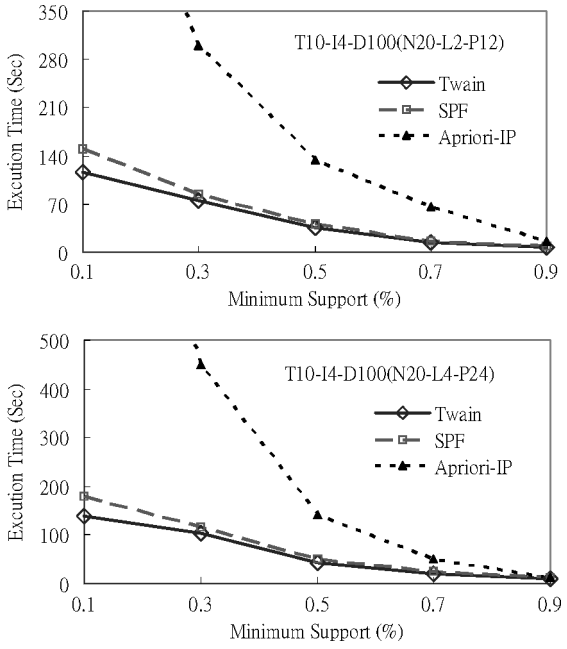


Fig. 7. The execution time under various minimum support.

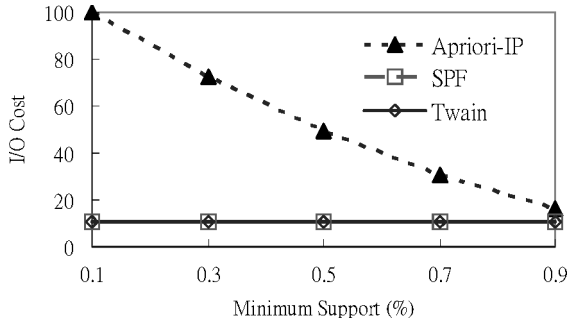


Fig. 8. I/O costs under various minimum supports.

Therefore, *Twain* can perform much better than *Apriori*^{IP} and is more efficient than *SPF* in terms of execution time.

4.4 I/O Costs and CPU Overheads

In this experiment, we examine I/O costs and the CPU overheads of all algorithms. As the method used in Pei et al. [2001], we assume that each sequential read of a byte consumes one unit of I/O cost and each random read of a byte of data consumes two units of I/O cost. The experimental result of I/O costs under various minimum supports is shown in Figure 8. The I/O costs of *SPF* and *Twain* remain the same as the minimum support decreases. Nevertheless, the cost of *Twain* is slightly less than that of *SPF*. However, the I/O cost of

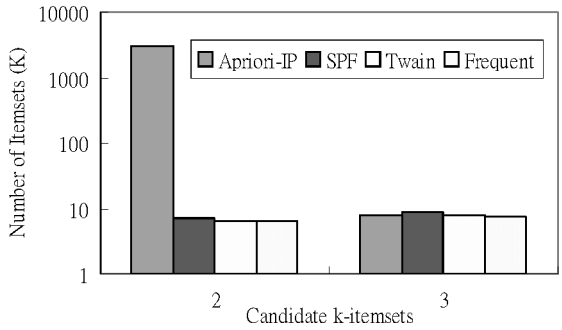


Fig. 9. Number of candidates generated.

Apriori^{IP} increases as the minimum support decreases. It is noted that the performance of I/O cost depends mainly on the number of database scans needed. As the minimum support decreases, the value of k (for k-itemset) increases. Recall that one database scan is needed in *Apriori*^{IP} whenever a determination of frequent k-itemsets is required. In contrast, I/O costs of *SPF* and *Twain*, due to the advantage of the scan reduction technique, are not affected as the minimum support varies.

Note that the scan reduction technique can only be used when the number of candidate 2-itemsets is very close to the number of frequent 2-itemsets. As explained in Section 2.4, *SPF* progressively filters out infrequent candidate 2-itemsets from one partition to another. This feature enables us to apply the scan reduction technique to *SPF*. In addition, since *Twain* can generate frequent 2-itemsets directly, the effect of scan reduction technique becomes more remarkable. To explore more insights into the number of candidates generated by all algorithms, another experiment is conducted and examined in Figure 9. As shown in Figure 9, *Twain* generates exactly the same number of candidate 2-itemsets as that of frequent 2-itemsets while *SPF* leads to a 98% candidate reduction rate in candidate 2-itemsets over *Apriori*^{IP}. This feature not only enables the scan reduction technique to be used in *Twain*, but also efficiently reduces the CPU and memory overheads in the following procedures. Note that candidate 3-itemsets are generated by joining 2-itemsets. Since the number of candidate 2-itemsets obtained by *SPF* is larger than the number of frequent 2-itemsets discovered by *Twain*, the number of candidate 3-itemsets generated by *SPF* is larger than the number of candidate 3-itemsets generated by *Twain*. In addition, the number of candidate 3-itemsets of *Twain* is the same as that of *Apriori*^{IP}. Assume there are n partitions in a database. All items are supposed to be independent and in uniform distribution. Assume there are in average m frequent temporal 1-itemsets in each partition. In *Apriori*^{IP}, there will be $C_2^m = m(m-1)/2$ candidate 2-itemsets in each partition. In accordance with the results reported in Park et al. [1997], the first two passes of scanning database induces about 62% of the total execution time. Thus, in each partition, the execution time of *Apriori*^{IP} is in proportional to $n * (m + \frac{m(m-1)}{2}) * \frac{100}{62}$. In addition, there are $n(n-1)/2$ subdatabases. Therefore, the total execution time

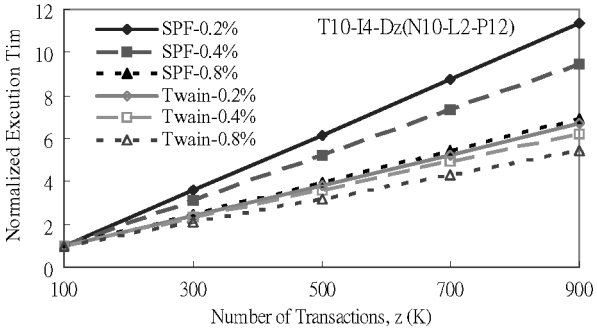


Fig. 10. Normalized execution time under various numbers of transactions.

of $Apriori^{IP}$ is in proportional to $\frac{n(n-1)}{2} * n * (m + \frac{m(m-1)}{2}) * \frac{100}{62}$. As for *SPF*, the execution time of ProcSG phase is in proportional to n. The number of candidate 2-itemset is about $2\% * m(m-1)/2$. By using scan reduction technique, the total candidate k-itemsets($k > 2$) are about 4 times of candidate 2-itemsets. Then, the total execution time of SPF is in proportion to $n + n * (m + \frac{m(m-1)}{2} * 0.02 * 5)$. Finally, *Twain* can generate exactly the same number of frequent 2-itemsets, which is about 85% of the candidate 2-itemsets generated by *SPF*. Therefore, the total execution time of *Twain* is in proportion to $n * (m + \frac{m(m-1)}{2} * 0.02 * 0.85 * 5)$. It is noted that although *SPF* generates more candidate k-itemsets, *SPF* still has the drawbacks of over-estimating and under-estimating the supports of frequent itemsets. *Twain* can overcome these two drawbacks successfully.

4.5 Scalability

Then, we conduct the experiments on different number of transactions in the synthetic dataset ($|D|$) to investigate the scalability of *Twain*. Among these, we consider three different minimum supports, that is, 0.2%, 0.4%, and 0.8%, in the experiments. Note that the execution time under various numbers of transactions is normalized with respect to the time for $T10-I4-D100$ of each experiment. As shown in Figure 10, the execution time of *SPF* and *Twain* increases linearly while the number of transactions in the synthetic database increases. It shows that both *SPF* and *Twain* perform well and do not suffer from the exponential increment of execution time as other Apriori-like algorithms do. It is because the bottleneck of Apriori-like algorithms is on the huge number of candidate 2-itemsets and candidate 3-itemsets. To generate frequent 2-itemsets and frequent 3-itemsets is time consuming. However, both *SPF* and *Twain* generate few candidate 2-itemsets efficiently as shown in Figure 9. *Twain* can even finds frequent 2-itemsets directly. In addition, *SPF* and *Twain* apply scan reduction technique to reduce scanning time of the database. Therefore, both algorithms can conquer this bottleneck. More specifically, *Twain* is of better scalability than *SPF* because the slopes of the lines for *Twain* are all smaller than the slopes of the lines for *SPF*. This feature shows *Twain* is more practicable than *SPF*.

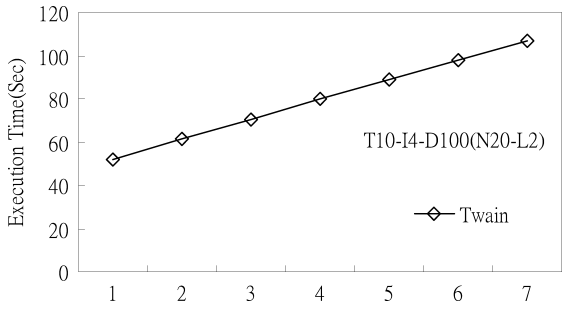


Fig. 11. The cumulative execution time of the incremental database.

4.6 Incremental Ability

Finally, we investigate the incremental ability of *Twain*. We divide $T10 - I4 - D20(N20 - L2 - P24)$ dataset into 7 subdatasets. The first subdataset contains the 1st to the 12th partitions in the original dataset. Each of the other subdatasets contains the following 2 partitions in the original dataset. These subdatasets are fed into *Twain* one by one. Note that the candidate 2-itemsets generated in each partition are examined and those whose occurrence frequencies are larger than the relative supports are passed to the next partition. Therefore, *ProcPM* can pass the latest candidate 2-itemsets of the previous sub-dataset to the new partition in the following sub-dataset. Then, *ProcPM* handles the transactions in the new partition and generates new candidate 2-itemsets in this partition. As shown in Figure 11, X-axis represents each run of the mining process. For example, the first point includes the 1st to the 12th partitions in the original dataset, the second point includes the 1st to the 14th partitions in the original dataset, the third point includes the 1st to the 16th partitions in the original dataset, and so on. Y-axis represents the execution time of each run. It is noted that the execution time increases linearly, which means *Twain* can utilize the information carried from the previous partitions well and can incrementally generate frequent itemsets efficiently.

5. CONCLUSION

We have presented a general model of mining association rules in a temporal database where the exhibition periods of the items are allowed to be different from one to another as in Chang et al. [2002]. However, some interesting rules may be under-estimated. In addition, the frequent exhibition periods of temporal itemsets may be over-estimated. To address this issue, we introduced the notions of *FCP* and *MFCP* to give more precise frequent exhibition periods of frequent temporal itemsets. In addition, we developed an efficient algorithms, referred to as *Twain* to discover precise general temporal association rules. Specifically, *Twain* can generate frequent 2-itemsets directly, which allows us to apply scan reduction technique to find candidate k-itemsets ($k > 2$) effectively. Moreover, *Twain* not only overcomes the drawbacks of *SPF* in Chang et al. [2002] for mining precise general temporal association rules, but also possesses the incremental mining ability. Some related theoretical properties were

derived in this article as well. The experimental results showed that *Twain* outperforms other algorithms in the quality of frequent patterns, execution time, I/O cost, CPU overhead and scalability.

REFERENCES

AGARWAL, R., AGGARWAL, C., AND PRASAD, V. 2000. A tree projection algorithm for generation of frequent itemsets. *J. Para. Distrib. Comput. (Special Issue on High Performance Data Mining)*.

AGRAWAL, R., IMIELINSKI, T., AND SWAMI, A. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, ACM, New York, 207–216.

AGRAWAL, R. AND SRIKANT, R. 1994. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, 478–499.

ALE, J. AND ROSSI, G. 2000. An approach to discovering temporal association rules. In *Proceedings of the ACM Symposium on Applied Computing 1*, ACM, New York, 294–300.

AYAD, A. M., EL-MAKKY, N. M., AND TAHA, Y. 2001. Incremental mining of constrained association rules. In *Proceedings of the 1st ACM-SIAM Conference on Data Mining*. ACM, New York.

BESEMANN, C. AND DENTON, A. 2005. Integration of profile hidden Markov model output into association rule mining. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, ACM, New York, 538–543.

BETTINI, C., WANG, X., AND JAJODIA, S. 1998. Mining temporal relationships with multiple granularities in time sequences. *Bulle. IEEE Comput. Soc. Tech. Comm. Data Eng.*.

BLANCHARD, J., GUILLET, F., GRAS, R., AND BRIAND, H. 2005. Using information-theoretic measures to assess association rule interestingness. In *Proceedings of the 5th IEEE International Conference on Data Mining*. IEEE Computer Society Press, Los Alamitos, CA.

CHANG, C.-Y., CHEN, M.-S., AND LEE, C.-H. 2002. Mining general temporal association rules for items with different exhibition periods. In *Proceedings of the 2nd IEEE International Conference on Data Mining*. IEEE Computer Society Press, Los Alamitos, CA.

CHEN, J., HE, H., WILLIAMS, G., AND JIN, H. 2004. Temporal sequence associations for rare events. In *Proceedings of the 8th Pacific Asia Conference on Knowledge Discovery and Data Mining*.

CHEN, X. AND PETR, I. 2000. Discovering temporal association rules: algorithms, language and system. In *Proceedings of the 16th IEEE International Conference on Data Engineering*. IEEE Computer Society Press, Los Alamitos, CA.

CHEN, X., PETROUNIAS, I., AND HEATHFIELD, H. 1998. Discovery of association rules in temporal databases. In *Proceedings of the Issues and Applications of Database Technology*.

COHEN, E., DATARY, M., FUJIWARAZ, S., GIONISX, A., INDYK, P., MOTWANIK, R., ULLMAN, J. D., AND YANGYY, C. 2001. Finding interesting associations without support pruning. *IEEE Trans. Knowl. Data Eng.*, 64–78.

HAN, J. AND FU, Y. 1995. Discovery of multiple-level association rules from large databases. In *Proceedings of the 21th International Conference on Very Large Data Bases*, 420–431.

HAN, J. AND KAMBER, M. 2000. *Data Mining: Concepts and Techniques*. Morgan-Kaufmann. San Francisco, CA.

HAN, J. AND PEI, J. 2000. Mining frequent patterns by pattern-growth: Methodology and implications. *ACM SIGKDD Explorations (Special Issue on Scaleble Data Mining Algorithms)*. ACM, New York.

HAN, J., PEI, J., MORTAZAVI-ASL, B., CHEN, Q., DAYAL, U., AND HSU, M.-C. 2000a. FreeSpan: Frequent pattern-projected sequential pattern mining. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, New York. 355–359.

HAN, J., PEI, J., AND YIN, Y. 2000b. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ACM, New York. 486–493.

HARMS, S. K. AND DEOGUN, J. S. 2004. Sequential association rule mining with time lags. *Journal of Intelligent Informatics Systems*.

JIANG, N. AND GRUENWALD, L. 2006. An efficient algorithm to mine online data streams. In *Proceedings of the 2006 KDD TDM Workshop*.

KE, Y., CHENG, J., AND NG, W. 2006. MIC framework: An information-theoretic approach to quantitative association rule mining. In *Proceedings of the 22nd IEEE International Conference on Data Engineering*. IEEE Computer Society Press, Los Alamitos, CA.

KIFER, D., BUCILA, C., GEHRKE, J., AND WHITE, W. 2002. DualMiner: A dual-pruning algorithm for itemsets with constraints. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York.

LAKSHMANAN, L., NG, R., HAN, J., AND PANG, A. 1998. Exploratory mining and pruning optimization of constrained associations rules. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*. ACM, New York.

LAKSHMANAN, L. V. S., NG, R., HAN, J., AND PANG, A. 1999. Optimization of constrained frequent set queries with 2-variable constraints. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, ACM, New York. 157–168.

LEE, C.-H., CHEN, M.-S., AND LIN, C.-R. 2003. Progressive partition miner: An efficient algorithm for mining general temporal association rules. *IEEE Trans. Knowl. Data Eng. 15*, 4 (Aug.), 1004–1017.

LEE, C.-H., LIN, C.-R., AND CHEN, M.-S. 2001a. On mining general temporal association rules in a publication database. In *Proceedings of the 1st IEEE International Conference on Data Mining*. (Nov.). IEEE Computer Society Press, Los Alamitos, CA.

LEE, C.-H., LIN, C.-R., AND CHEN, M.-S. 2001b. Sliding-window filtering: An efficient algorithm for incremental mining. In *Proceedings of the 10th ACM International Conference on Information and Knowledge Management*. ACM, New York.

LIN, J.-L. AND DUNHAM, M. 1998. Mining association rules: Anti-skew algorithms. In *Proceedings of the 14th IEEE International Conference on Data Engineering*, IEEE Computer Society Press, Los Alamitos, CA. 486–493.

LIU, B., HSU, W., AND MA, Y. 1999. Mining association rules with multiple minimum supports. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York.

MUELLER, A. 1995. Fast sequential and parallel algorithms for association rule mining: A comparison. Tech. Rep. CS-TR-3515, Dept. of Computer Science, Univ. of Maryland, College Park, MD.

MUHONEN, B. G. J. AND TOIVONEN, H. 2005. Mining non-derivable association rules. In *Proceedings of the 5th ACM SIAM Conference on Data Mining*. ACM, New York.

NG, R. AND HAN, J. 1994. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, 144–155.

PARK, J.-S., CHEN, M.-S., AND YU, P. S. 1997. Using a hash-based method with transaction trimming for mining association rules. *IEEE Trans. Knowl. Data Eng. 9*, 5 (Oct.), 813–825.

PEI, J. AND HAN, J. 2000. Can we push more constraints into frequent pattern mining? In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York.

PEI, J., HAN, J., MORTAZAVI-ASL, B., PINTO, H., CHEN, Q., DAYAL, U., AND HSU, M.-C. 2001. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th IEEE International Conference on Data Engineering*. IEEE Computer Society Press, Los Alamitos, CA.

SAVASERE, A., OMIECINSKI, E., AND NAVATHE, S. 1995. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21th International Conference on Very Large Data Bases*, 432–444.

SRIKANT, R. AND AGRAWAL, R. 1995. Mining generalized association rules. In *Proceedings of the 21th International Conference on Very Large Data Bases*. 407–419.

SRIKANT, R. AND AGRAWAL, R. 1996. Mining quantitative association rules in large relational tables. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*. ACM, New York.

STEINBACH, M., TAN, P.-N., AND KUMAR, V. 2005. Support envelopes: A technique for exploring the structure of association patterns. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM, New York.

SZYMON AND JAROSZEWICZ. 2006. Polynomial association rules with applications to logistic regression. In *Proceedings of the 11st ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM, New York.

TANSEL, A. AND AYAN, N. 1998. Discovery of association rules in temporal databases. In *Proceedings of the AAAI on Knowledge Discovery in Databases*.

TOIVONEN, H. 1996. Sampling large databases for association rules. In *Proceedings of the 22nd International Conference on Very Large Data Bases*, 134–145.

WANG, K., HE, Y., AND HAN, J. 2000. Mining frequent itemsets using support constraints. In *Proceedings of the 26th International Conference on Very Large Data Bases*.

XUAN-HIEU, P., LE-MINH, N., TU-BAO, H., AND SUSUMU, H. 2005. Improving discriminative sequential learning with rare-but-important associations. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM, New York.

YANG, C., FAYYAD, U., AND BRADLEY, P. 2001. Efficient discovery of error-tolerant frequent itemsets in high dimensions. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. ACM, New York.

ZHENG, Z., KOHAVI, R., AND MASON, L. 2001. Real world performance of association rule algorithms. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, F. Provost and R. Srikant, Eds. ACM, New York. 401–406.

Received October 2006; revised January 2007 and April 2007; accepted May 2007