



An efficient algorithm for incremental mining of temporal association rules

Tarek F. Gharib^a, Hamed Nassar^b, Mohamed Taha^{c,*}, Ajith Abraham^d

^a Faculty of Computer & Information Sciences, Ain Shams University, Cairo 11566, Egypt

^b Faculty of Computers & Informatics, Suez Canal University, Ismailia, Egypt

^c Faculty of Computers & Informatics, Benha University, Benha, Egypt

^d Machine Intelligence Research Labs (MIR Labs)¹, Scientific Network for Innovation and Research Excellence, Auburn, Washington, USA

ARTICLE INFO

Article history:

Received 14 September 2008

Received in revised form 5 March 2010

Accepted 8 March 2010

Available online 15 March 2010

Keywords:

Temporal Association Rules (TAR)

Incremental temporal mining

Updating temporal association rules

Temporal mining

ABSTRACT

This paper presents the concept of temporal association rules in order to solve the problem of handling time series by including time expressions into association rules. Actually, temporal databases are continually appended or updated so that the discovered rules need to be updated. Re-running the temporal mining algorithm every time is ineffective since it neglects the previously discovered rules, and repeats the work done previously. Furthermore, existing incremental mining techniques cannot deal with temporal association rules. In this paper, an incremental algorithm to maintain the temporal association rules in a transaction database is proposed. The algorithm benefits from the results of earlier mining to derive the final mining output. The experimental results on both the synthetic and the real dataset illustrate a significant improvement over the conventional approach of mining the entire updated database.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Data mining is the process of extracting interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from large information repositories and it is the core process of Knowledge Discovery in Database (KDD) [1]. Data mining techniques include association rules mining, classification, clustering, mining time series, and sequential pattern mining, to name a few, with association rules mining receiving a significant research attention [2].

Many algorithms for discovering association rules in transaction databases have been developed and widely studied: Apriori and its variations, partitioning, sampling, TreeProjection, and FP-growth algorithms [3–5,23]. Furthermore, other variants of mining algorithms were presented to provide more mining capabilities, such as incremental updating, mining of generalized and multi-level rules, mining of quantitative rules, mining of multi-dimensional rules, constraint-based rule mining, mining with multiple minimum supports, mining associations among correlated or infrequent items, and mining of temporal association rules [2,20,21].

Recently, temporal data mining has become a core technical data processing technique to deal with changing data. Temporal data exist extensively in economics, finance, communication, and other areas such as weather forecasting [6]. Temporal Association Rules (TAR) is an interesting extension to association rules by including a temporal dimension. When considering the time dimension, it leads to different forms of association rules such as discovering association rules that may hold during some time intervals but not during others [7].

Different methodologies were proposed to explore the problem of discovering temporal association rules. In the previous works, discovering association rules was performed on a given subset of a database specified by time. However, these works did not consider the individual exhibition period of each item. The exhibition period of an item is the time duration from the partition

* Corresponding author.

E-mail addresses: tgharib@eun.edu (T.F. Gharib), nassar66@hotmail.com (H. Nassar), moh_twins@hotmail.com (M. Taha), ajith.abraham@ieee.org (A. Abraham).

¹ <http://www.mirlabs.org>.

when this item appears in the transaction database to the partition when this item no longer exists [2]. That is, the exhibition period is the time duration when the item is available to be purchased. Hence, these works cannot be effectively applied to a temporal transaction database, such as a publication database, where the exhibition periods of the items are different from one another. As a result, the concept of general temporal association rules has been proposed where the items are allowed to have different exhibition periods, and their supports are made in accordance with their exhibition periods [8]. The accompanying mining measures, support and confidence, have been reformulated to reflect this new mining model. Also, new mining algorithms have been presented for the general temporal association rules in transaction databases such as Progressive Partition Miner (PPM) [8], and Segmented Progressive Filter (SPF) [12–15,22]. On the other hand, other algorithms have been proposed for mining temporal association rules with numerical attributes such as the TAR algorithm [9].

As a matter of fact, temporal databases are often appended by adding new transactions. Hence, the previously discovered rules have to be maintained by discarding the rules that become insignificant and including new valid ones. Currently, some algorithms are proposed for the incremental mining of temporal association rules with numerical attributes [10]. However, the incremental mining of temporal transaction databases is still at its infancy. Moreover, the incremental temporal mining algorithms with numerical attributes cannot be easily adapted to the transaction database. In the case of numerical attributes, we deal with objects. Each object has a unique ID and a set of numerical attributes. The database is viewed as sequence of snapshots of objects. The interestingness measures used are density and strength instead of confidence. A temporal association rule is discovered from what is called base cubes instead of itemsets as long as it achieves the minimum support and strength. Moreover, Many algorithms have been proposed for the incremental mining of association rules such as Fast UPdate algorithm (FUP), The Update Large Itemsets algorithm (ULI), Negative Border with Partitioning (NBP), Update With Early Pruning algorithm (UWEP), New Fast UPdate algorithm (NFUP), Fast Incremental Mining (FIM) algorithm and Pre-FUPP Algorithm [16–19]. Some of these algorithms address the problem of determining when to update, while the others simply treat arbitrary insertions and deletions of transactions. Unfortunately, none of these algorithms address the incremental mining of temporal association rules.

In this paper, the Incremental Temporal Association Rules Mining (ITARM) is proposed. It is used to maintain temporal frequent itemsets after the temporal transaction database has been updated. The proposed algorithm employs the skeleton of the incremental procedure of the Sliding-Window Filtering algorithm (SWF) [11].

Rest of the paper is organized as follows. Section 2 gives a description of some preliminaries in temporal association rules mining. Section 3 provides a review of some related works and Section 4 presents the proposed algorithm ITARM in detail and its correctness is proven in Section 5. The performance of the proposed algorithm is empirically evaluated in Section 6 and Section 7 concludes the paper.

2. Preliminaries

This Section presents some preliminaries to facilitate the presentation of the proposed algorithm. For a given temporal database DB , let n be the number of partitions with a time granularity, such as month, quarter, year. $db^{s,e}$ denotes the part of the transaction database formed by a continuous region from partition P_s to partition P_e and $|db^{s,e}| = \sum_{h=s}^e |P_h|$ where $db^{s,e} \subseteq DB$ and $|P_h|$ is the number of transactions in the partition P_h . An item $y^{s,e}$ is termed as a temporal item of a given item y , meaning that P_s is the starting partition of y and P_e is the ending partition of y [8,12].

Table 1

An illustrative transaction database where the items have individual exhibition periods. An itemset $x^{s,e}$ is called a maximal Temporal Itemset (TI) in a partial database $db^{s,e}$.

		Transaction database			Item information			
		Date	TID	Transactions	Item	Publication date		
DB (original database)	P1	Jan-09	T1	B D	db ^{1,3}	db ^{2,3}	A	Jan – 2002
			T2	B C D			B	Apr – 2003
			T3	B C			C	Jul – 2004
			T4	A D			D	Aug – 2008
P2	Feb-09	T5	B C E	E			Feb – 2009	
		T6	D E	F			Mar – 2009	
		T7	A B C					
		T8	C D E					
db (increment)	P3	Mar-09	T9	B C EF				
			T10	B F				
			T11	A D				

Example 1. Table 1 shows a publication database DB containing the transaction data from January 2009 to February 2009. The number of transactions recorded in March is the incremental database db . The original database DB is segmented into two partitions P_1 and P_2 in accordance with the “month” granularity and db contains one partition P_3 . The partial database $db^{2,3} \subseteq (DB \cup db)$ consists of partitions P_2 and P_3 . The publication date of each item is shown on the right side of Table 1. It is worth mentioning that, in the publication database, each item usually has the same cut-off date of the item exhibition period. A temporal item $E^{2,3}$ denotes that the exhibition period of $E^{2,3}$ is from the beginning time of partition P_2 to the end time of partition P_3 , if s is the latest starting partition number of all items belonging to x in the temporal database and e is the earliest ending partition number of the items belonging to x [8,12]. In this case, (s, e) is referred to as the Maximal Common exhibition Period (MCP) of the itemset x and is denoted by $MCP(x)$. For example, as shown in Table 1, itemset $DE^{2,3}$ is a maximal temporal itemset, whereas $DE^{3,3}$ is not a maximal temporal itemset because $MCP(D) = (1,3)$ and $MCP(E) = (2,3)$ hence $MCP(DE) = (2,3)$. A temporal itemset $z^{s,e}$ is called a temporal Sub-Itemset (SI) of a maximal temporal itemset $x^{s,e}$ if $z \subset x$ [13]. For example, the maximal temporal itemset $BDE^{2,3}$ has the sub-itemsets $\{B^{2,3}, D^{2,3}, E^{2,3}, BD^{2,3}, BE^{2,3}, DE^{2,3}\}$. The relative support of a temporal itemset x is given by the following equation:

$$supp(x^{MCP(x)}) = \frac{|\{T \in db^{MCP(x)} \mid x \subseteq T\}|}{|db^{MCP(x)}|}$$

Where the numerator indicates the number of transactions in the partial database $db^{s,e}$ that contain x . The general temporal association rule is defined as an implication in the form $(X \Rightarrow Y)^{MCP(XY)}$ with the following support and confidence:

$$supp((X \Rightarrow Y)^{MCP(XY)}) = supp((X \cup Y)^{MCP(XY)})$$

where

$$supp((x \cup y)^{MCP(xy)}) = \frac{|\{T \in db^{MCP(xy)} \mid x, y \subseteq T\}|}{|db^{MCP(xy)}|}$$

$$conf(X \Rightarrow Y)^{MCP(XY)} = \frac{supp((X \cup Y)^{MCP(XY)})}{supp(X)^{MCP(XY)}}$$

The general temporal association rule is termed to be frequent within its MCP if and only if its support is not smaller than the minimum support threshold (min_sup), and its confidence is not smaller than the minimum confidence needed (min_conf) [2]. Consequently, the problem of mining general temporal associations can be decomposed into the following three steps [8,13]:

1. Generate all frequent maximal temporal itemsets (TIs) with their support values.
2. Generate the support values of all corresponding temporal sub-itemsets (SIs) of frequent TIs.
3. Generate all temporal association rules that satisfy min_conf using the frequent TIs and/or SIs.

3. Related work

Several algorithms have been proposed for mining temporal association rules. Most of these algorithms are based on dividing the temporal transaction database into several partitions according to the time granularity imposed, and then mining temporal association rules by finding frequent temporal itemsets within these partitions.

Among these algorithms, Lee et al. proposed the PPM algorithm to discover general temporal association rules in a publication database [8]. The basic idea of PPM is to first partition the publication database in light of exhibition periods of items and then progressively accumulate the occurrence count of each candidate 2-itemset based on the intrinsic partitioning characteristics. The PPM algorithm is designed to employ a filtering threshold in each partition to prune out those cumulatively infrequent 2-itemsets early on. Also, it employs the scan reduction technique to reduce the number of database scans depending on the feature that the number of candidate 2-itemsets generated by PPM is very close to the number of frequent 2-itemsets.

Also, the Segmented Progressive Filter algorithm (SPF) is proposed for mining temporal association rules where the exhibition periods of the items are allowed to be different from one to another [12,13]. The algorithm consists of two procedures: the first procedure segments the database into sub-databases in such a way that items in each sub-database will have either a common starting time or a common ending time. Then, for each sub-database, the second procedure progressively filters candidate 2-itemsets with cumulative filtering thresholds either forward or backward in time. This feature allows SPF to adopt the scan reduction technique by generating all candidate k -itemsets from candidate 2-itemsets directly. Then, these candidates are transformed to TI's and the corresponding SI's are generated. Finally, the database is scanned once to determine all frequent TI's and SI's.

Moreover, Huang et al. devised the TWo end Association miNER algorithm (Twain) to give more precise frequent exhibition periods of frequent temporal itemsets [2]. Twain employs the start time and the end time of each item to provide precise frequent

exhibition period. It generates candidate 2-itemsets with their Maximal Frequent Common exhibition Periods (MFCPs) 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, it adopts the scan reduction technique to generate all frequent k -itemsets from the generated frequent 2-itemsets.

4. The proposed algorithm

The main objective of the proposed algorithm is to maintain temporal frequent itemsets after the temporal transaction database has been updated. The algorithm employs the skeleton of the incremental procedure of the Sliding-Window Filtering algorithm (SWF) [11]. The idea of SWF is similar to temporal mining algorithms such as PPM, SPF and Twain. All these algorithms are similar in partitioning the temporal database according to a time granularity and generating the candidate 2-itemsets. They require two database scans: the first for generating candidate 2-itemsets and the second for checking candidate k -itemsets generated directly from candidate 2-itemsets.

The proposed algorithm depends on storing only candidate 2-itemsets generated from the previous mining process with their support counts instead of storing all the previously found frequent itemsets. Its main idea is based on updating these candidates and utilizing the scan reduction technique to find new frequent itemsets with only one database scan. In the traditional approach, re-running a temporal mining algorithm costs at least two database scans. Hence, one of the important features of the proposed algorithm is to reduce the number of database scans required for the updating process.

In practice, the incremental algorithm is not invoked every time a transaction is added to the database. However, it is invoked after a non-trivial number of transactions are added. In our case, the proposed algorithm is invoked when no more transactions can be recorded in the imposed time granularity (e.g. current month). However, it is also designed to handle the problem of extending a given partition several times. Sometimes, the incremental database transactions are recorded in the same time granularity of the last partition of the original database (e.g. the same month). When the proposed algorithm is invoked, it first decides whether the incremental database will be added to the last partition or it will be treated as a new partition. The flowchart of the ITARM algorithm is depicted in Fig. 1. The algorithm consists of two main procedures: the Incremental procedure, which actually performs the incremental mining and the Update_C2 procedure, which performs preprocessing for the Incremental procedure. The proposed algorithm first checks if the time stamp of the incremental database is equal to the time stamp of the transactions of the last partition of the original database. If yes, the algorithm makes some preprocessing by calling Update_C2 procedure then merging the transactions of last partition of original database with the incremental database before calling the incremental procedure. If no, the algorithm calls the incremental procedure of the ITARM algorithm directly.

Fig. 2 illustrates the incremental procedure of the ITARM algorithm and Table 2 shows the meaning of various symbols used. First, the algorithm finds the candidate 2-itemsets of the incremental database. Second, it updates the counts of the stored candidate 2-itemsets with the counts of the candidate 2-itemsets of the incremental database. That is, the support counts of the common itemsets are summed while the counts of remaining itemsets are kept as it is. In the third step, a relative minimum

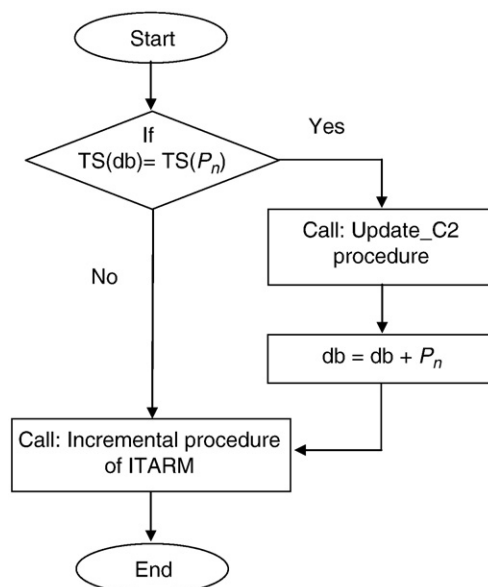


Fig. 1. The general flowchart of the ITARM algorithm.

Algorithm 1: ITARM Algorithm

Inputs: DB, db, C_2^{DB} , n and min_sup
Output: L' , C_2^{DB+db} // initially, $L' = \emptyset$, $C_2^{DB+db} = \emptyset$

1. Find the candidate 2-itemsets (C_2^{db}) of db.
2. For each itemset $X \in C_2^{DB}$
 - If $X \in C_2^{db}$ then
 - $X.support_{DB+db} = X.support_{DB} + X.support_{db}$
 - $C_2^{DB+db} = C_2^{DB+db} \cup \{X\}$
 - Remove X from C_2^{DB} and C_2^{db}
 - // adding the remaining itemsets to C_2^{DB+db}
 - $C_2^{DB+db} = C_2^{DB+db} \cup C_2^{DB} \cup C_2^{db}$
3. // Filtering candidate 2-itemsets
 - For each itemset $X \in C_2^{DB+db}$
 - If $(X.support_{DB+db} < min_sup * \sum_{m=Z.start}^n |P_m|)$
 - Remove X from C_2^{DB+db}
4. Initialize $C^{DB+db} = C_2^{DB+db}$, $k=2$
 - While $(C_k^{DB+db} \neq \emptyset)$ //candidate generations
 - $C_{k+1}^{DB+db} = C_k^{DB+db} \times C_k^{DB+db}$
 - // where \times is the Apriori join operator
 - $C^{DB+db} = C^{DB+db} \cup C_{k+1}^{DB+db}$
 - $k = k+1$
5. Initialize TI = \emptyset and SI = \emptyset
 - // Generation of Candidate TI's
 - For each itemset $X \in C^{DB+db}$
 - TI = TI $\cup \{X^{MCP(X)}\}$
 - // Generation of candidate SI's
 - For each itemset $X^{MCP(X)} \in TI$
 - SX = $\{Z^{MCP(X)} \mid Z \subset X\}$
 - SI = SI $\cup SX$
6. For p=1 to n
 - For each itemset $X^{s,e} \in (TI \cup SI)$
 - If $s \leq p$ and $e \geq n$ then
 - $X^{s,e}.support = X^{s,e}.support + X^{s,e}.support_p$
 - For each itemset $X^{s,e} \in (TI \cup SI)$
 - if $(X^{s,e}.support \geq min_sup * |db^{s,e}|)$ then
 - $L' = L' \cup \{X^{s,e}\}$

Fig. 2. The incremental procedure of the ITARM algorithm.

support count is used to filter the candidates in order to employ the scan reduction technique. In the fourth step, the algorithm applies the scan reduction technique by generating all candidate k -itemsets from candidate $(k - 1)$ -itemsets directly. Then, these candidates are transformed to temporal itemsets and the corresponding sub-itemsets are generated based on these temporal itemsets in the fifth step. Finally, the frequent temporal itemsets and sub-itemsets can be determined by scanning the updated database only once.

On the other hand, if the transactions of the incremental database are recorded with the same time stamp of the transactions of the last partition of the original database, some preprocessing is needed before applying the incremental procedure. Both the transactions of the incremental database and the transactions of the last partition of the original database are joined. Hence, the counts of the stored candidate 2-itemsets of the original database should be updated. Fig. 3 shows the Update_C2 procedure of the proposed algorithm. This procedure scans the transactions of the last partition of the original database to subtract the counts of the candidate 2-itemsets occurring in the last partition from the counts of the stored candidate 2-itemsets. Example 2 illustrates how the ITARM algorithm works in the two cases.

Table 2

The list of symbols used in the proposed algorithm.

Symbol	Meaning
N	The number of partitions
DB	The original database
Db	The incremental database
$DB + db$	The updated database
min_sup	The minimum support
C_2^{DB}	The candidate 2-itemsets in DB
C_2^{db}	The candidate 2-itemsets in db
$C_2^{DB + db}$	The new candidate 2-itemsets in $DB + db$
$C^{DB + db}$	The candidate itemsets of all sizes in $DB + db$
L'	The set of frequent itemsets in $DB + db$
X	An itemset
$X.start$	The starting partition number of x
$X.support$	The number of transactions containing X in the database
$X.support_{DB}$	The number of transactions containing X in DB
$X.support_{db}$	The number of transactions containing X in db
$X.support_{DB + db}$	The number of transactions containing X in $DB + db$
$X.support_p$	The number of transactions containing X in partition p
$ P_m $	The number of transactions in partition m
SX	The set of all subsets of a given temporal itemset

Example 2. Recall the transaction database shown in Table 1, where the transaction database DB is assumed to be segmented into two partitions P_1 and P_2 according to the month time granularity from January 2009 to February 2009. Suppose that $min_sup = 30\%$ and $min_conf = 75\%$.

As evident from Table 1, the transactions of the incremental database are recorded in March, while the transactions of last partition of the original database P_2 are recorded in February. Hence, the incremental procedure will be applied directly without any preprocessing. Table 3.a shows the candidate 2-itemsets that are obtained from a previous temporal mining process using a temporal association rules mining algorithm. The proposed algorithm first finds the candidate 2-itemsets of the incremental database with their support counts. The results are also illustrated in Table 3b. Then, the algorithm begins to determine the candidate 2-itemsets of the updated database by updating the support counts of the candidate 2-itemsets of both the original and the incremental databases. Table 4 shows the new candidate 2-itemsets after the proposed algorithm updates the support counts of BC to be 5 and CE to be 3. Each candidate itemset has two attributes: the start and the count attributes. The start attribute indicates the start partition that itemset occurs and the count attribute indicates the support count from the start partition to the end partition. Then, a filtering process is performed to determine new candidate 2-itemsets, which are those itemsets that have support count equal to or greater than the relative minimum support count. For example, the itemset BC occurs from P_1 to P_3 . Its relative minimum support count is equal to $[min_sup * \sum_{m=1}^3 |P_m|] = [12 * 30\%] = 4$ where 12 is the number of transactions from P_1 to P_3 . This item has a support count greater than relative support count ($5 > 4$) hence it will be in the set of new candidate 2-itemsets. Note that although the itemset BD appears in two transactions of P_1 and in one transaction of P_3 , its final count is one not three as shown in Table 3. This is because these counts are not the total support counts over the whole database. It is just relative counts employed to be able to utilize the scan reduction technique. However, the real support counts are computed for all itemsets of all sizes in the database scan.

Table 4 depicts the new candidate 2-itemsets BC, BF and CE. The scan reduction technique is then applied to the new candidate 2-itemsets to generate high-level candidates. However in our case, no more candidates are produced. Then, these candidates are transformed to temporal itemsets (TI's) and sub-temporal itemsets (SI's) by calculating the maximal common exhibition period of the

Algorithm 2: Update C2 procedure of the ITARM algorithm

Inputs: C_2^{DB} , P_n and min_sup

Output: C_2^{DB}

1. For each itemset $X \in C_2^{DB}$
 - For each Transaction $T \in P_n$
 - If $X \subseteq T$ then
 - $X.support_{DB} = X.support_{DB} + 1$
-

Fig. 3. The Update_C2 procedure of the ITARM algorithm.

Table 3
(a) Candidate 2-itemsets in DB, (b) Candidate 2-itemsets in *db*.

C_2	Start	Count
<i>(a) P1 + P2</i>		
BC	1	4
CE	2	2
DE	2	2
<i>(b) P3</i>		
AD	3	1
BC	3	1
BD	3	1
BE	3	1
BF	3	3
CE	3	1
CF	3	1
DF	3	1
EF	3	1

Table 4
Update candidate 2-itemsets in *DB + db*.

$P1 + P2 + P3$			
C_2	Start	Count	Relative support
AD	3	1	$(4 \times 30\%) = 2$
BC	1	5	$(12 \times 30\%) = 4$
BD	3	1	$(4 \times 30\%) = 2$
BE	3	1	$(4 \times 30\%) = 2$
BF	3	3	$(4 \times 30\%) = 2$
CE	2	3	$(8 \times 30\%) = 3$
CF	3	1	$(4 \times 30\%) = 2$
DE	2	2	$(8 \times 30\%) = 3$
DF	3	1	$(4 \times 30\%) = 2$
EF	3	1	$(4 \times 30\%) = 2$

items that appear in each itemset. For example, the itemset CE consists of two items C and E. Since $MCP(C) = (1,3)$ and $MCP(E) = (2,3)$, we have $MCP(CE) = (2,3)$. Hence, the TI will be $CE^{2,3}$ as shown in Table 5 and the corresponding SI's are generated ($C^{2,3}$ and $E^{2,3}$) as shown in Table 6. After that, the proposed algorithm scans the updated database to calculate the support counts of all TI's and SI's as shown in Table 7. Table 8 illustrates the new frequent temporal itemsets in the updated database.

On the other hand, if the transactions of the incremental database are recorded in February instead of March, then this means they are to be considered as an extension to the P_2 transactions, instead of being a new partition P_3 . Therefore, the proposed algorithm joins both transactions in one partition P_2 . However, before this joining, the proposed algorithm updates the stored candidate 2-itemsets of the original database by scanning P_2 . That is, by removing any occurrence count of any itemset in P_2 from the counts of the stored candidate 2-itemsets of the original database. Table 9a shows the updated candidate 2-itemsets of the original database after removing P_2 from the original database and merging it with P_3 to be the new P_2 . Also, Table 9b depicts the candidate 2-itemsets of the incremental database with their support counts. Then, the algorithm begins to determine the candidate 2-itemsets of the updated database by updating the support counts of the candidate 2-itemsets of both the original and the incremental databases. The results of the updating process are illustrated in Table 10. After that, the algorithm proceeds in the same way as explained above.

Table 5
Generate temporal itemsets in *DB + db*.

Itemset	B		C		MCP(BC)	TI's
	Start	End	Start	End		
BC	1	3	1	3	(1,3)	$BC^{1,3}$
Itemset	B		F		MCP(BF)	TI's
	Start	End	Start	End		
BF	1	3	3	3	(3,3)	$BF^{3,3}$
Itemset	C		E		MCP(CE)	TI's
	Start	End	Start	End		
CE	1	3	2	3	(2,3)	$CE^{2,3}$

Table 6

Generate sub-temporal itemsets in $DB + db$.

TI 's	SI 's
$BC^{1,3}$	$B^{1,3}$ $C^{1,3}$
$BF^{3,3}$	$B^{3,3}$ $F^{3,3}$
$CE^{2,3}$	$C^{2,3}$ $E^{2,3}$

Table 7

Update the support counts of temporal and sub-temporal itemsets.

Scanning the updated database for TI 's and SI 's			
Candidate itemsets		Counts	Relative support
SI 's	$B^{1,3}$	8	$(12 \times 30\%) = 4$
	$C^{1,3}$	6	$(12 \times 30\%) = 4$
	$B^{3,3}$	3	$(4 \times 30\%) = 2$
	$F^{3,3}$	3	$(4 \times 30\%) = 2$
	$C^{2,3}$	4	$(8 \times 30\%) = 3$
TI 's	$E^{2,3}$	4	$(8 \times 30\%) = 3$
	$BC^{1,3}$	5	$(12 \times 30\%) = 4$
	$BF^{3,3}$	3	$(4 \times 30\%) = 2$
	$CE^{2,3}$	3	$(8 \times 30\%) = 3$

The computational complexity of the proposed algorithm can be studied in terms of both the number of database scans and the number of comparisons made. As mentioned earlier, the proposed algorithm needs only one database scan in order to make the update, while SPF and Twain algorithms need two database scans. Assuming that the time needed to scan all the transactions of the database is n , the proposed algorithm needs time (n) while both SPF and Twain need time $(2n)$. Hence, each of these algorithms, including the proposed algorithm, are of order $O(n)$ in terms of number of database scans. For the number of comparisons needed, the number of comparisons performed in the only database scan of the proposed algorithm is equivalent to the number of comparisons performed in the second database scan of both SPF and Twain algorithms. To update the candidate 2-itemsets, another scan of the whole database is required for both SPF and Twain, while the proposed algorithm just scans the incremental database. The proposed algorithm handles the incremental database in the same way that both SPF and Twain handle the last partition of the database in the first scan. Consequently, the number of comparisons needed by the proposed algorithm is just equal to the number of comparisons needed by the other two algorithms in handling one partition.

5. Correctness of proposed algorithm

In this section, the correctness of the proposed algorithm is proved using the same definitions provided in [8]. Generally, TAR algorithms deal with two types of candidate itemsets, 1) Type α : the candidate itemsets that were carried over from the previous progressive candidate set in the previous phase and remain as candidate itemsets after the current partition is included into consideration, and 2) type β : the candidate itemsets that were not in the progressive candidate set in the previous phase, but are newly selected after only taking the current data partition into account. Let $N_{ph}(X)$ be the number of transactions in partition P_h that contain itemset X . The region ratio of an itemset is defined as follows:

Table 8

The frequent temporal itemsets in $DB + db$.

Frequent itemsets	Counts	
$L1$	$B^{1,3}$	8
	$C^{1,3}$	6
	$B^{3,3}$	3
	$F^{3,3}$	3
	$C^{2,3}$	4
	$E^{2,3}$	4
$L2$	$BC^{1,3}$	5
	$BF^{3,3}$	3
	$CE^{2,3}$	3

Table 9(a) Candidate 2-itemsets in *DB*, (b) candidate 2-itemsets in *db*.

C_2	Start	Count
<i>(a) P1</i>		
BC	1	2
<i>(b) P2</i>		
AB	2	1
AC	2	1
AD	2	1
BC	2	3
BD	2	1
BE	2	2
BF	2	3
CD	2	1
CE	2	3
CF	2	1
DE	2	2
DF	2	1
EF	2	1

Definition 1. A region ratio of an itemset X for the transaction database db^{ij} , denoted by $r_{i,j}(X)$, is

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

From the definition, it is clear that the region ratio of an itemset is the support of that itemset if only the part of transaction database db^{ij} is considered.

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

Proof. To prove the “if” condition, consider the following two cases: first, suppose the 2-itemset X is not in the candidate 2-itemsets before the processing of partition P_i . Since $r_{i,i}(X) \geq \min_sup$, Itemset X will be selected as a type β candidate itemset by our proposed algorithm after the processing of partition P_i . On the other hand, if the itemset X is already in the candidate 2-itemsets in *DB* before the processing of partition P_i , itemset X will remain as a type α candidate itemset. Obviously, for the above two cases, itemset X will remain in candidate 2-itemsets throughout the processing from P_i to P_j since for any integer t in the interval $[i, j]$, $r_{i,t}(X) \geq \min_sup$. To prove the “only if” condition, i.e., if X remains in candidate 2-itemsets after the processing of partition P_j , then there exists an i such that for any t in the interval $[i, j]$, $r_{i,t}(X) \geq \min_sup$. Note that itemset X can be either type α or type β candidate itemset in the candidate 2-itemsets after the processing of partition P_j . Suppose X is a type β candidate itemset there, then this implication follows by setting $j = i$ since $r_{i,i}(X) \geq \min_sup$.

On the other hand, suppose that X is a type α candidate itemset after the processing of P_j , which means itemset X has become a type β candidate itemset in a previous phase. Then, we will trace backward the type of itemset X from partition P_j (i.e., looking over

Table 10Update candidate 2-itemsets in *DB + db* $P1 + P2$.

Update candidate 2-itemsets in <i>DB + db</i> $P1 + P2$				
C_2	Start	Count	Relative support	
AB	2	1	$(8 \times 30\%) = 3$	
AC	2	1	$(8 \times 30\%) = 3$	
AD	2	1	$(8 \times 30\%) = 3$	
BC	1	5	$(12 \times 30\%) = 4$	
BD	2	1	$(8 \times 30\%) = 3$	
BE	2	2	$(8 \times 30\%) = 3$	
BF	2	3	$(8 \times 30\%) = 3$	
CD	2	1	$(8 \times 30\%) = 3$	
CE	2	3	$(8 \times 30\%) = 3$	
CF	2	1	$(8 \times 30\%) = 3$	
DE	2	2	$(8 \times 30\%) = 3$	
DF	2	1	$(8 \times 30\%) = 3$	
EF	2	1	$(8 \times 30\%) = 3$	

$P_j, P_{j-1}, P_{j-2}, \dots$) until the partition that records itemset X as a type β candidate itemset is first encountered. Referring to the partition identified above as partition P_i , we have, for any t in the interval $[i, j]$, $r_{i,t}(X) \geq \min_sup$, completing the lemma proof. \square

Lemma 2. An itemset X remains in candidate itemsets after the processing of partition P_j if and only if there exists an i such that $r_{i,j}(X) \geq \min_sup$, where \min_sup is the minimal support required.

Proof. Clearly, the proof of “only if” condition follows directly from Lemma 1. To prove the “if” condition of this lemma. If there exists an i such that $r_{i,j}(X) \geq \min_sup$, then let u be the largest v such that $r_{i,v}(X) < \min_sup$. If such a u does not exist, it follows from Lemma 1 that itemset X will remain in the candidate itemsets after the processing of partition P_j . If such a u exists, we have $r_{u+1,j}(X) \geq \min_sup$ since $r_{i,u}(X_2) < \min_sup$ and $r_{i,j}(X_2) \geq \min_sup$. It again follows from Lemma 1 that itemset X will remain in the candidate itemsets after the processing of partition P_j , completing the lemma proof. \square

Lemma 2 leads to the following theorem that states the completeness of candidates 2-itemsets generated by the proposed algorithm.

Theorem 1. If there exists a frequent itemset $X^{t,n}$ in the transaction database $db^{t,n}$ such that $r_{t,n}(X) \geq \min_sup$, then X will be in the candidate itemsets produced by the proposed algorithm.

Proof. Let n be the number of partitions of the transaction database. Since the itemset $X^{t,n}$ is a frequent itemset, we have $r_{t,n}(X) \geq \min_sup$, which essentially is a special case of Lemma 2 for $i=t$ and $j=n$, proving this theorem. \square

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

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

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

Theorem 1 and Lemma 3 lead to the following theorem which states the correctness of the proposed algorithm.

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

Proof. Since itemset $X_k^{t,n}$ is frequent, we have $r_{t,n}(X_k) \geq \min_sup$. As mentioned above, all of its sub-itemsets $X_h^{t,n}$ s ($h < k$) will be frequent with $r_{t,n}(X_h) \geq \min_sup$. Precisely, $X_2^{t,n}$ s are in essence special cases of $X_h^{t,n}$ s with $h=2$. Thus, according to the implication of Theorem 1, $X_2^{t,n}$ will be in the candidate set of itemsets produced by the proposed algorithm. In this way, according to the anti-monotone Apriori-like heuristic, i.e., if any length k itemset $X_k^{t,n}$ is not frequent in the database, its length $(k+1)$ super-itemset $X_{k+1}^{t,n}$ will never be frequent, the super-itemset $X_k^{t,n}$ of $X_2^{t,n}$ will be produced by the proposed algorithm, proving this theorem. \square

In addition, if there exists a frequent TI 3-itemset $ABC^{t,n}$, for example, in the transaction database $db^{t,n}$, meaning that $r_{t,n}(ABC) \geq \min_sup$, then we have $r_{t,n}(AB) \geq \min_sup$, $r_{t,n}(AC) \geq \min_sup$, and $r_{t,n}(BC) \geq \min_sup$. According to Theorem 1, all SIs of $ABC^{t,n}$ ($AB^{t,n}$, $AC^{t,n}$, $BC^{t,n}$, $A^{t,n}$, $B^{t,n}$, and $C^{t,n}$) will be in the progressive candidate set of itemsets produced by the proposed algorithm. Hence, Theorem 2 states the correctness of the proposed algorithm.

6. Experimental results

In this section, a performance comparison of the proposed algorithm with some other temporal algorithms is presented. The comparison is conducted with the SPF [12,13] and Twain [2] algorithms as they are recent algorithms for mining general temporal association rules. The comparisons are evaluated from different aspects including: run time, minimum support, original database size and incremental database size. All the experiments are performed on a 1.8 GHz Intel Core 2 Duo PC machine with 1 Gigabytes main memory, running on Microsoft Windows XP Professional and all the programs are coded in C#.

The experiments were performed on synthetic publication databases, used in the experimental results of the previously mining algorithms introduced in [2,8,12,13]. In essence, a publication database is a set of transactions where each transaction T is a set of items, where each item contains an individual exhibition period. For the simplicity of presentation, the notation $[Tx - ly - Dz - dr(Nm - Ln - Po)]$ is used to represent a dataset in which x is the average size of the transactions, y is the average size of maximal potentially frequent itemsets, z is the number of transactions in the original database (in thousands), r is the number of transactions in the incremental database (in thousands), m is the number of distinct items (in thousands), n is the number of maximal potentially frequent itemsets (in thousands), and o is the number of partitions.

Also, a real dataset (BMS-POS) from KDDCUP web site is used to evaluate the performance of the proposed algorithm. It is the same dataset that is used to evaluate the performance of the Twain algorithm [2]. The BMS-POS dataset contains several years' worth of point-of sale data from a large electronics retailer. Since this retailer has so many different products, product categories are used as items. The transaction in this dataset is a customer's purchase transaction consisting of all the product categories

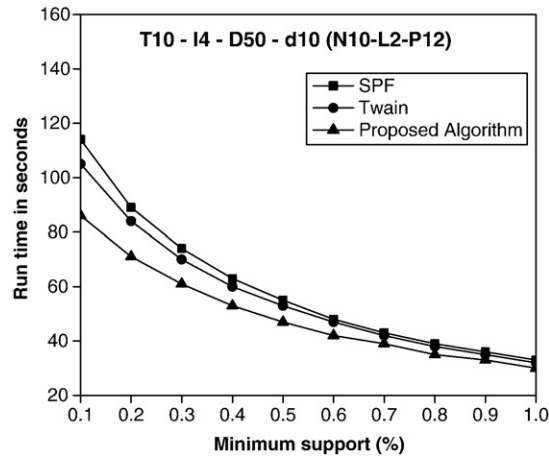


Fig. 4. The run time under various minimum supports.

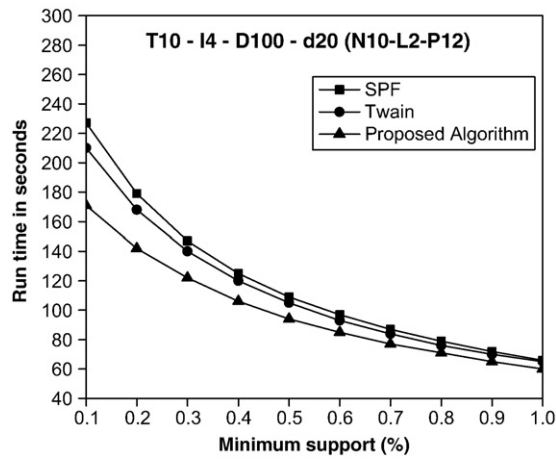


Fig. 5. The run time under various minimum supports.

purchased at one time. The goal for this dataset is to find associations between product categories purchased by customers in a single visit to the retailer. The dataset contains 515,597 transactions and 1657 distinct items. The maximum transaction size is equal to 164 while the average transaction size is equal to 6.5.

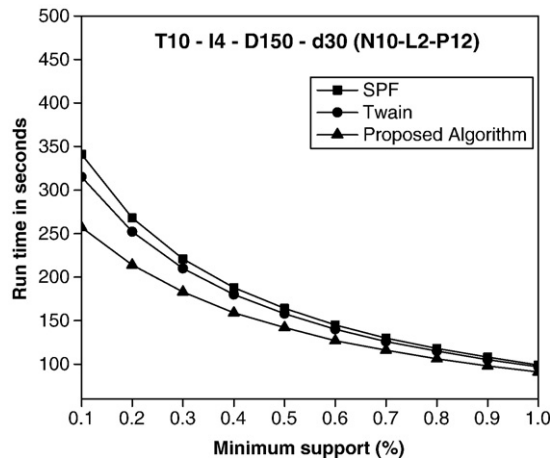


Fig. 6. The run time under various minimum supports.

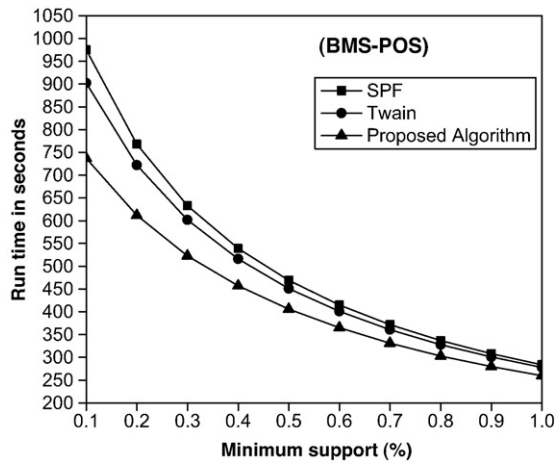


Fig. 7. The run time under various minimum supports.

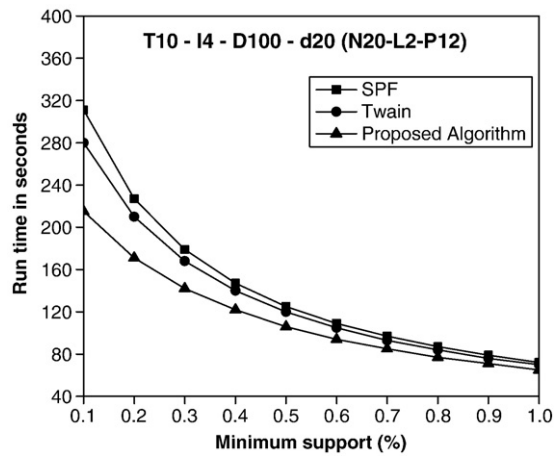


Fig. 8. The run time under various minimum supports.

In the first experiment, several datasets are used to investigate the run time of the proposed algorithm comparing with the other two algorithms by varying the minimum support from 0.1% to 1%. The experimental results on various datasets are shown in Figs. 4, 5 and 6 with different sizes of the original database 50 k, 100 k and 150 k respectively. In addition, Fig. 7 illustrates the relative performance of the proposed algorithm on the real dataset (BMS-POS). Figs. 8 and 9 depict the performance of the

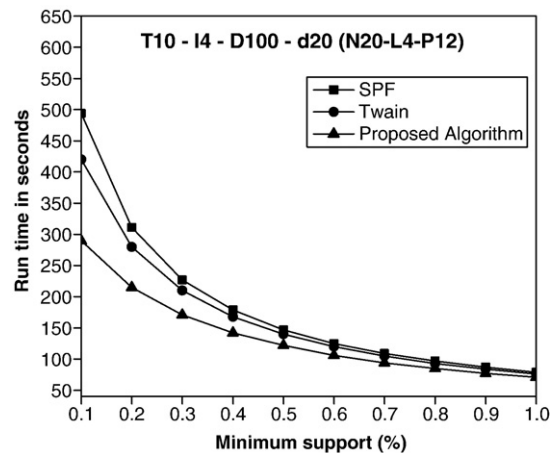


Fig. 9. The run time under various minimum supports.

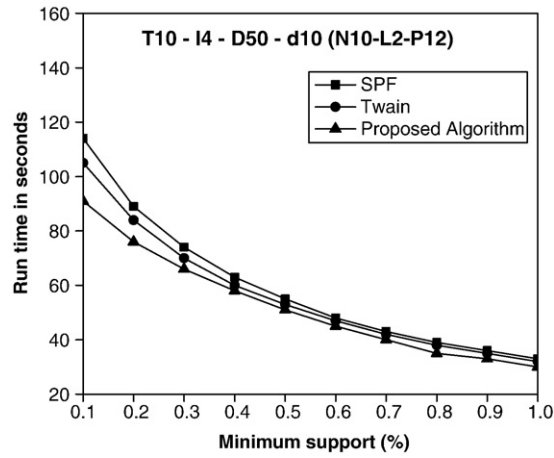


Fig. 10. The run time under various minimum supports.

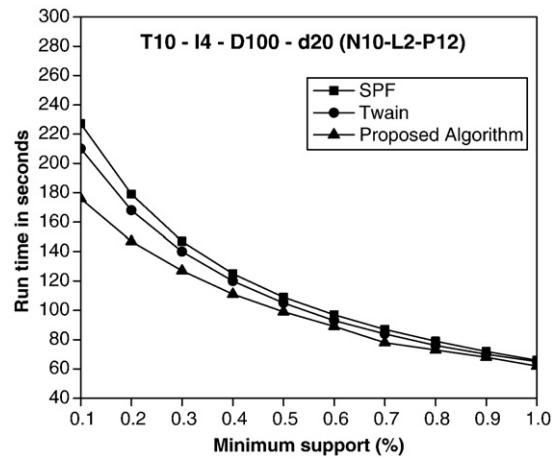


Fig. 11. The run time under various minimum supports.

proposed algorithm for various values of m and n on the dataset T10-I4-D100-d20. In all these datasets, the incremental database has a new time stamp. Figs. 10 and 11 illustrate the performance of the proposed algorithm when the incremental database has the same time stamp as that of the last partition of the original database. Experimental results reveal that no matter what the

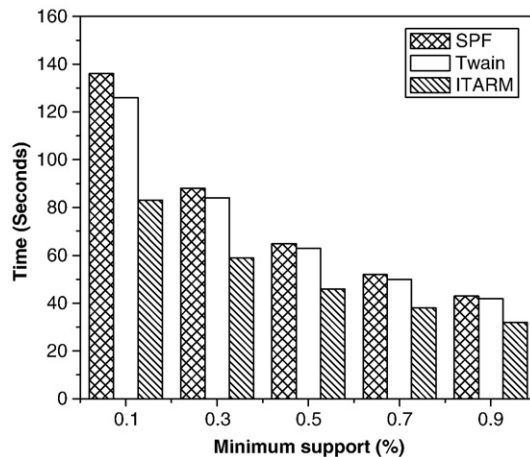


Fig. 12. The time needed to generate candidate itemsets.

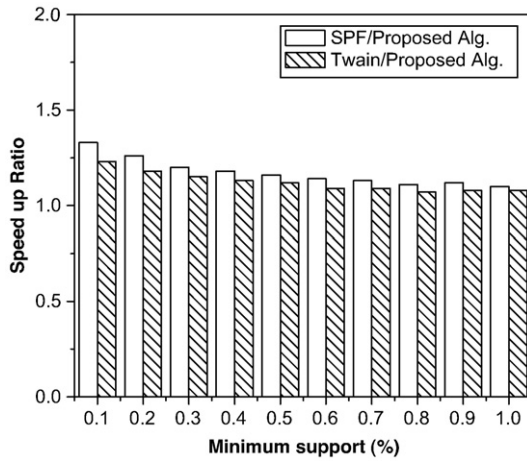


Fig. 13. Speedup ratio.

combination of different parameters is, the proposed algorithm performs significantly better than the other two algorithms in terms of the run time. This is due to the scan reduction achieved by the proposed algorithm where it needs only one database scan to perform the update, while both the SPF and Twain algorithms need two database scans to perform the update.

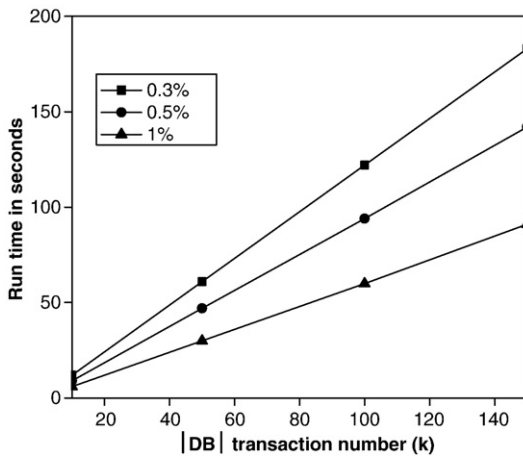


Fig. 14. Scalability with the number of transactions in DB.

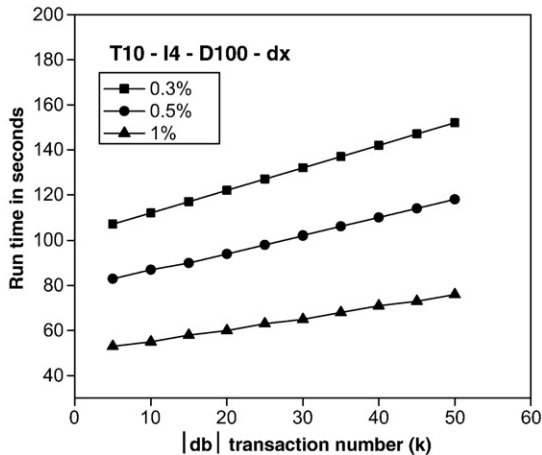


Fig. 15. Scalability with the number of transactions in db.

From Figs. 4–11, it can be noted that the margin grows as the minimum support decreases, due to the large number of frequent itemsets produced with the low support thresholds, while this number decreases with high support thresholds. Fig. 12 shows the time needed by each algorithm to generate all the candidate itemsets. It can be seen that the proposed algorithm reduces significantly the time needed to generate the candidate itemsets with respect to the other two algorithms. In addition, Fig. 13 illustrates the speedup ratio achieved by the proposed algorithm with respect to SPF and Twain algorithms. The proposed algorithm reaches to a speedup ratio up to 1.33 faster than the SPF algorithm and 1.23 faster than the Twain algorithm.

Moreover, two different experiments are designed to investigate the scalability of the proposed algorithm against different sizes of both the original and the incremental databases. Three different minimum support thresholds are considered in these experiments: 0.3%, 0.5% and 1%. Fig. 14 depicts the scale-up performance of the proposed algorithm as the size of the original database increases while the size of the incremental database is fixed. Also, Fig. 15 shows the scalability of the algorithm by varying the size of the incremental database. It can be noticed that the run time of the proposed algorithm increases linearly as the number of transactions in the database increases. This shows that the proposed algorithm can utilize the information carried from the previous mining well and can incrementally generate frequent itemsets efficiently.

7. Conclusion

The concept of temporal association rule (TAR) has been introduced in order to solve the problem of handling time series by including time expressions into association rules. We have presented an algorithm called ITARM for updating temporal association rules in the transaction database. The proposed algorithm reduces the time needed for generating new candidates by storing candidate 2-itemsets. It presents a technique to update the previously generated candidates instead of re-generating them from scratch. The experiments show a significant improvement over the traditional approach of mining the whole updated database. In all experiments, the proposed algorithm consistently outperforms SPF and TWAIN in terms of run time. Moreover, the experiments also show that the proposed algorithm is scalable and can work with large databases.

Acknowledgements

My thanks to Dr. Mohammed J. Zaki for his valuable comments and his revision.

References

- [1] Qiankun Zhao, Sourav S. Bhowmick, Association Rule Mining: A Survey, Technical Report, Center for Advanced Information Systems (CAIS), Nanyang Technological University, Singapore, 2003.
- [2] Jen-Wei Huang, Bi.-Ru. Dai, Ming-Syan Chen, Twain: Two-End Association Miner with Precise Frequent Exhibition Periods, In the ACM Transactions on Knowledge Discovery from Data, vol. 1, August 2007, Number 2, Article 8.
- [3] R.C. Agarwal, C.C. Aggarwal, V.V.V. Prasad, A tree projection algorithm for generation of frequent item sets, In the Journal of Parallel and Distributed Computing 61 (3) (March 2001) 350–371.
- [4] Jiawei Han, Jian Pei, Yiwen Yin, Mining Frequent Patterns Without Candidate Generation, In Proceedings of the ACM-SIGMOD International Conference on Management of Data, Dallas, Texas, USA, May 2000, pp. 1–12.
- [5] Gosta Grahne, Jianfei Zhu, Fast algorithms for frequent itemset mining using FP-trees, In the IEEE Transactions on Knowledge and Data Engineering 17 (10) (October 2005) 1347–1362.
- [6] Hui Ning, Haifeng Yuan, Shugang Chen, Temporal association rules in mining method, In Proceedings of the 1st International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06), Zhejiang, China, vol. 2, June 2006, pp. 739–742.
- [7] Yingjiu Li, Peng Ning, Xiaoyang Sean Wang, Sushil Jajodia, Discovering Calendar-Based Temporal Association Rules, In Proceedings of the 8th International Symposium on Temporal Representation and Reasoning, Cividale del Friuli, Italy, June 2001, pp. 111–118.
- [8] Chang-Hung Lee, Ming-Syan Chen, Cheng-Ru Lin, Progressive partition miner: an efficient algorithm for mining general temporal association rules, In the IEEE Transaction on Knowledge and Data Engineering 15 (4) (July/August 2003) 1004–1017.
- [9] Wei Wang, Jiong Yang, Richard Muntz, TAR: Temporal Association Rules on Evolving Numerical Attributes, In Proceedings of the 17th International Conference on Data Engineering (ICDE01), Heidelberg, Germany, April 2001, pp. 283–292.
- [10] Vincent Ng, Stephen Chan, Derek Lau, Cheung Man Ying, Incremental Mining for Temporal Association Rules for Crime Pattern Discoveries, In Proceedings of the 18th Australasian Database Conference (ADC2007), Ballarat, Victoria, Australia, January 2007, pp. 123–132.
- [11] Chang-Hung Lee, Cheng-Ru Lin, Ming-Syan Chen, Sliding-Window Filtering: An Efficient Algorithm for Incremental Mining, In Proceedings of the 10th International Conference on Information and Knowledge Management, Atlanta, Georgia, USA, November 2001, pp. 263–270.
- [12] Junheng-Huang, Wang-Wei, Efficient algorithm for mining temporal association rule, In the International Journal of Computer Science and Network Security (IJCSNS) 7 (4) (April 2007) 268–271.
- [13] Cheng-Yue Chang, Ming-Syan Chen, Chang-Hung Lee, Mining General Temporal Association Rules for Items with Different Exhibition Periods, In Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02), Maebashi City, Japan, December 2002, pp. 59–66.
- [14] Abdullah Uz. Tansel, Susan P. Imberman, Discovery of Association Rules in Temporal Databases, In Proceedings of the 4th International Conference on Information Technology (ITNG'07), Las Vegas, Nevada, USA, April 2007, pp. 371–376.
- [15] Chang-Hung Lee, Cheng-Ru Lin, Ming-Syan Chen, On Mining General Temporal Association Rules in a Publication Database, In Proceedings of the First IEEE International Conference on Data Mining (ICDM'01), San Jose, California, USA, November 2001, pp. 337–344.
- [16] Tarek F. Gharib, Mohamed Taha, Hamed Nassar, An efficient technique for incremental updating of association rules, In the International Journal of Hybrid Intelligent Systems (IJHIS) (ISSN: 1448-5869) 5 (1) (January 2008) 45–53 IOS Press.
- [17] Ahmed Emam, Future Direction of Incremental Association Rules Mining, In Proceedings of the 47th ACM Annual Southeast Regional Conference, Clemson, South Carolina, March 2009, Article No. 74.
- [18] Wang Yuping, Dong Nanping, Zhou Guanling, Incremental Updating Algorithm Based on Partial Support Tree for Mining Association Rules, In Proceedings of IITA International Conference on Control, Automation and Systems Engineering (CASE 2009), Zhangjiajie, China, July 2009, pp. 17–20.
- [19] Chun-Wei Lin, Tzung-Pei Hong, Wen-Hsiang Lu, The Pre-FUFP algorithm for incremental mining, In the International Journal of Expert Systems with Applications 36 (5) (July 2009) 9498–9505 Elsevier Science.
- [20] Shin-Yi Wu, Yen-Liang Chen, Discovering hybrid temporal patterns from sequences consisting of point- and interval-based events, In the International Journal of Data & Knowledge Engineering 68 (11) (November 2009) 1309–1330.

- [21] Haifeng Lia, Hong Chen, Mining non-derivable frequent itemsets over data stream, In the International Journal of Data & Knowledge Engineering 68 (5) (May 2009) 481–498.
- [22] Anjana Pandey, K.R. Pardasani, PPCI algorithm for mining temporal association rules in large databases, In International Journal of Information & Knowledge Management (JIKM) 08 (04) (April 2009) 345–352.
- [23] Ying-Hsiang Wen, Jen-Wei Huang, Ming-Syan Chen, Hardware-enhanced association rule mining with hashing and pipelining, In Proceedings of IEEE Transactions on Knowledge and Data Engineering 20 (6) (June 2008) 784–795.



Tarek Fouad Gharib is an Associate Professor in the department of Information Systems at Ain Shams University, Cairo, Egypt. He received his Ph.D. degree in Theoretical Physics from the University of Ain Shams in 1994. His research interests include data mining techniques, bioinformatics, graph and sequential data mining and Information retrieval. He has published over 20 papers on data mining. He received the National Science Foundation Award in 2001.



Hamed Nassar is Full Professor at the Computer Science Department, Suez Canal University, Egypt, since 2004. He obtained his B.Sc. in Electrical Engineering from Ain Shams University, Egypt, in 1979, and obtained his M.Sc. in Electrical Engineering and Ph.D. in Computer Engineering, both from the New Jersey Institute of Technology, USA, in 1985 and 1989, respectively. He has published over 30 research papers in refereed international journals and conferences, and over 50 scientific articles in local and international trade magazines. He has supervised around 30 M.Sc. and Ph.D. students, and has taught computer science and engineering courses and served as external examiner in numerous universities in Egypt, USA, Saudi Arabia, and Kenya.



Mohamed Taha received his B.Sc. degree in computer science, at Suez Canal University, Egypt, in June 2001, and his M.Sc. degree in computer science, at Ain Shams University, Egypt, in February 2009. He is currently a Ph.D. student at Ain Shams University and works as a lecturer assistant at computer science department, Benha University, Egypt. His research interests concern data mining, knowledge discovery, image forgery, distance learning.



Ajith's research and development experience includes over 20 years in the Industry and Academia. He works in a multi-disciplinary environment involving machine intelligence, terrorism informatics, network security, sensor networks, e-commerce, Web intelligence, Web services, computational grids, data mining and applied to various real world problems. He has given more than 35 plenary lectures and conference tutorials in these areas. He has published over 500+ publications and some of the works have also won best paper awards at International conferences and also received several citations.

Currently he is also coordinating the activities of the Machine Intelligence Research Labs (MIR Labs), International Scientific Network of Excellence, which has members from over 60 countries. He has a world wide academic experience with formal appointments in Monash University, Australia; Oklahoma State University, USA; Chung-Ang University, Seoul; Jinan University, China; Rovira i Virgili University, Spain; Dalian Maritime University, China; Yonsei University, Seoul and Open University of Catalonia, Spain, National Institute of Applied Sciences (INSA), France and Norwegian University of Science and Technology (NTNU), Norway. For about 2.5 years, he was working under the Institute of Information Technology Advancement (IITA) Professorship Program funded by the South Korean Government. He received Ph.D. degree in Computer Science from Monash University, Australia and a Master of Science degree from Nanyang Technological University, Singapore.

He co-chairs the IEEE Systems Man and Cybernetics Society, Technical committee on soft computing. He serves the editorial board of several reputed International journals and has also guest edited 35 special issues on various topics. He is actively involved in the Hybrid Intelligent Systems (HIS); Intelligent Systems Design and Applications (ISDA); Information Assurance and Security (IAS); and Next Generation Web Services Practices (NWeSP) series of International conferences, besides other conferences. He is a Senior Member of IEEE, IEEE Systems Man and Cybernetics Society, IEEE Computer Society, IET (UK), IEAust (Australia) etc. More information at: <http://www.softcomputing.net>.