Egyptian Computer Science Journal Vol.30 No3 September 2008

ITARM: Incremental Temporal Association Rules Mining

Mohamed Taha[1], Hamed Nassar[1], Tarek F. Gharib[2]

[1] *Faculty of Computers & Informatics, Suez Canal University, Ismaillia, Egypt*
[2] *Faculty of Computer & Information Science, Ain Shams University, Cairo, Egypt*
tgharib@asunet.shams.edu.eg

Abstract

A temporal association rule is an association rule that holds during specific time intervals. Temporal databases contain rich information that can be extracted by knowledge discovery and data mining techniques. Some algorithms were proposed for mining temporal association rules. In the real world, temporal databases are continually appended or updated so the discovered rules need to be updated. Re-running the temporal mining algorithm every time is inefficient since it ignores the previously discovered rules, and repeats the work done previously. Also, existing incremental mining techniques cannot deal with temporal association rules. In this paper, an incremental algorithm to maintain the temporal association rules in the transaction database is proposed. The algorithm exploits the results of earlier mining to derive the final mining output. The results show a significant improvement over the traditional approach of mining the whole updated database.

Keywords: TAR, Incremental Temporal Mining, Updating Temporal Association Rules, Temporal Mining

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 known as one of the core processes 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: the Apriori and its variations, partitioning, sampling, TreeProjection, and FP-growth algorithms [3, 4, 5]. Furthermore, other variants of mining algorithms are 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].

Recently, temporal data mining has become a core technical data processing technique to deal with changing data. Temporal data exist extensively in economical, financial, 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, this will lead to extract different forms of association rules such as discovering association rules that may hold during some time intervals but not during others [7].

Different methodologies have been proposed to explore the problem of discovering temporal association rules. In the early work, discovering association rules is performed from a given subset of a database specified by time. However, these works do not consider the individual exhibition period of each item. 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 [2]. That is, the exhibition period is the time duration when the item is 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 to 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 were presented for the general temporal association rules in transaction databases such as Progressive Partition Miner (PPM) [8], Segmented Progressive Filter (SPF) [12 - 15]. On the other hand, other algorithms have been proposed for mining temporal association rules with numerical attributes such as the TAR algorithm [9].

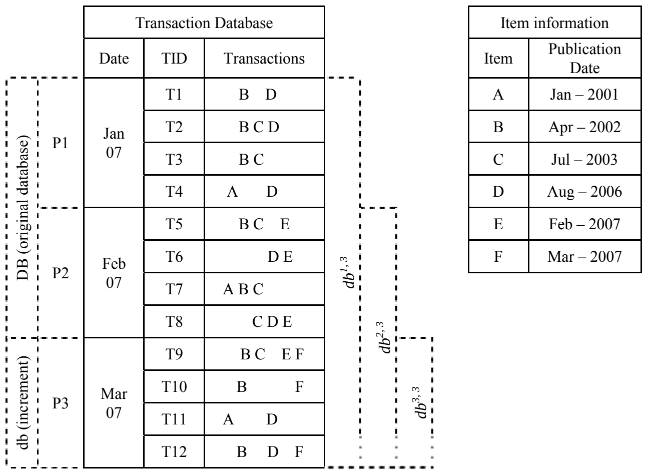
As a matter of fact, temporal databases are often appended by adding new transactions. Hence, the previously discovered rules have to be maintained by discarding the rules that become insignificant and including new valid ones. Currently, some algorithms are proposed for the incremental mining of temporal association rules with numerical attributes [10]. However, the incremental mining of temporal transaction databases is not as fortunate. Moreover, the incremental temporal mining algorithms with numerical attributes can not be easily adoptable to the transaction database because the problem terms are mainly different. For example, itemsets, support and confidence are used in transaction database while base cubes, density and strength are used with numerical attributes. 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].

For the rest of the paper, Section 2 gives a description of some preliminaries in temporal association rules mining. Section 3 provides a review of some related work. Section 4 presents the proposed algorithm ITARM in detail. The performance of the proposed algorithm is empirically evaluated in Section 5 whereas Section 6 concludes our work.

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 $|\mathrm{db}^{s,e}| = \sum_{h=s}^{e} |P_h|$ where $db^{s,e} \subseteq DB$ and $|P_h|$ is the number of transaction 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].

Example 1: Figure 1 shows a publication database DB containing the transaction data from January 2007 to February 2007. 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 Figure 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.



	Transaction Database		
	Date	TID	Transactions
P1	Jan 07	T1	B D
		T2	B C D
		T3	B C
		T4	A D
P2	Feb 07	T5	B C E
		T6	D E
		T7	A B C
		T8	C D E
P3	Mar 07	T9	B C E F
		T10	B F
		T11	A D
		T12	B D F

Item information	
Item	Publication Date
A	Jan – 2001
B	Apr – 2002
C	Jul – 2003
D	Aug – 2006
E	Feb – 2007
F	Mar – 2007

Figure 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 $db^{s,e}$ 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 it is denoted by MCP(*x*). For example, as shown in Figure 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-Itemsets (*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:

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

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:

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

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

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

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 association 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 (*SI*s) of frequent *TI*s.
3. Generate all temporal association rules that satisfy *min_conf* using the frequent *TIs* and/or *SIs.*

3. Related Work

Several algorithms are 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 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 early prune out those cumulatively infrequent 2-itemsets. 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 item in each sub-database will have either the common starting time or the 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 of adopting 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. Proposed Algorithm

In this section, the proposed algorithm will be described in details. 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 idea such as PPM, SPF and Twain. All of 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.

```
Inputs:
  DB, db, $C_2^{DB}$ , n and min_sup
Output:
  $L'$ , $C_2^{DB+db}$    // initially, $L' = \varnothing$, $C_2^{DB+db} = \varnothing$
Algorithm:
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 $(X.support_{DB+db} < min\_sup * \sum_{m=x.start}^{n} |P_m|)$
            Remove X from $C_2^{DB+db}$
4- Initialize $C^{DB+db} = C_2^{DB+db}$, $k$=2
   While ($C_k^{DB+db} \neq \varnothing$ ) //candidates 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 = $\varnothing$ and SI= $\varnothing$
   // 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<=p and e >=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 ($X^{s,e}.support \geq min\_supp \times |db^{s,e}|$ ) then
            $L' = L' \cup \{ X^{s,e} \}$
```

Figure 2. The ITARM algorithm.

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. One of the important features of the proposed algorithm is to reduce the number of database scans required for the updating process. The steps of the ITARM algorithm are shown in Figure 2 and Table 1 shows the meaning of various symbols used. First, the algorithm finds the candidate 2-itemsets of the increment database. Second, it updates the counts of the stored candidate 2-itemsets with the counts of the candidate 2-itemsets of the increment 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 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. Example 2 shows an example to illustrate how the ITARM algorithm works.

Example 2: Recall the transaction database shown in Figure 1, where the transaction database *DB* is assumed to be segmented into two partitions *P1* and *P2* according to the month time granularity from January 2007 to February 2007. Suppose that *min_sup*= 30% and *min_conf* = 75%.

Figure 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 increment database with their support counts. The results are also shown in Figure 3.b. 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 increment databases. Figure 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 [$min\_sup * \Sigma^3_{m=1} |P_m|$] = [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 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 Figure 4. 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 1: The list of symbols used in the proposed Algorithm

Symbol	Meaning
n	The number of partitions
DB	The original database
db	The increment 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

Figure 4 shows 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 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) so the MCP(CE)=(2,3). Hence, the TI will be $CE^{2,3}$ as shown in Figure 5 and the corresponding SI's are generated ($C^{2,3}$ and $E^{2,3}$) as shown in Figure 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 Figure 7. Figure 8 shows the new frequent temporal itemsets in the updated database.

candidate 2-itemsets in *DB*		
P1 + P2		
C_2	Start	count
BC	1	4
CE	2	2
DE	2	2

(a)

candidate 2-itemsets in *db*		
P3		
C_2	start	count
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

(b)

Figure 3. (a) Candidate 2-itemsets in DB
(b) Candidate 2-itemsets in db

Update candidate 2-itemsets in *DB+db*			
P1 + P2+P3			
C_2	Start	Count	Relative Support
AD	3	1	(4 × 30%) = 2
BC	1	5	(12 × 30%) = 4
BD	3	1	(4 × 30%) = 2
BE	3	1	(4 × 30%) = 2
BF	3	3	(4 × 30%) = 2
CE	2	3	(8 × 30%) = 3
CF	3	1	(4 × 30%) = 2
DE	2	2	(8 × 30%) = 3
DF	3	1	(4 × 30%) = 2
EF	3	1	(4 × 30%) = 2

Figure 4. Update candidate 2-itemsets in DB+db

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(BF)	*TI's*
	start	end	Start	end		
CE	1	3	2	3	(2,3)	$CE^{2,3}$

Figure 5. Generate temporal itemsets in DB+db

The Sub-temporal Itemsets in *DB+db*	
TI's	*SI's*
$BC^{1,3}$	$B^{1,3}$
	$C^{1,3}$
$BF^{3,3}$	$B^{3,3}$
	$F^{3,3}$
$CE^{2,3}$	$C^{2,3}$
	$E^{2,3}$

Figure 6. Generate sub-temporal itemsets in DB+db

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

Figure 7. Update the support counts of temporal & sub-temporal itemsets

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

Figure 8. The frequent temporal itemsets in DB+db

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 can be easily adapted to handle the problem of extending a given partition several times. Sometimes, the increment database transactions are recorded in the same time granularity of the last partition of the original database (e.g. the same month). Our algorithm can deal with this situation by storing the candidate 2-itemsets of the last partition of the original database separated from the candidate 2-itemsets of the other partitions. By this way, the algorithm does not cost more candidates to be stored. However, it is just a separation process of the candidates. When the algorithm is invoked, it first decides whether the increment database will be added to the last partition or it will be treated as a new partition.

5. 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 and Twain 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 increment 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#.

5.1. Dataset

The experiments were performed on a publication database using the synthetic data 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 of which each item contains an individual exhibition period. For the simplicity of presentation, the notation [Tx - Iy - 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 increment 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.

5.2. Performance Evaluation

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 Figures 9, 10 and 11 with different sizes of the original database 50k, 100k and 150k respectively. Also, Figure 12 and Figure 13 show the performance of the proposed algorithm for various values of m and n on the dataset T10-I4-D100-d20. Note that no matter what 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 needs two database scans to perform the update.

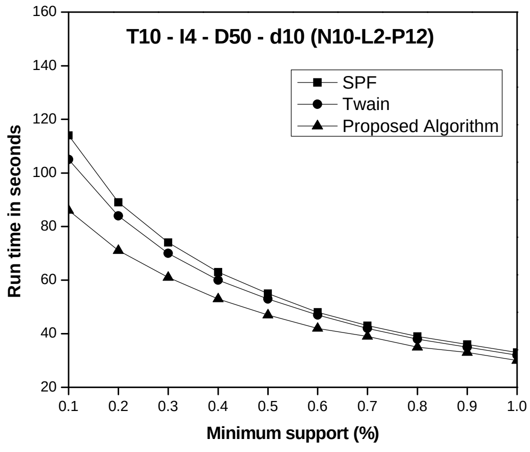


Figure 9. The run time under various minimum support

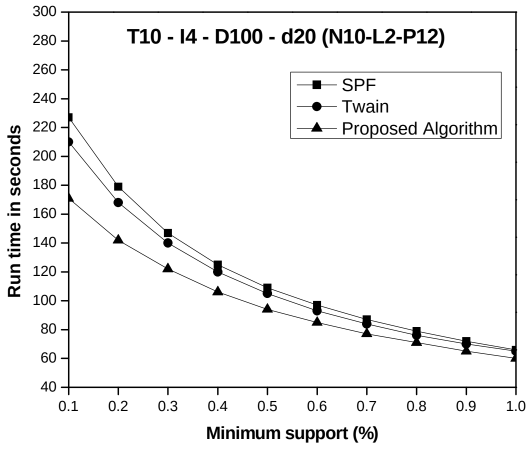


Figure 10. The run time under various minimum support

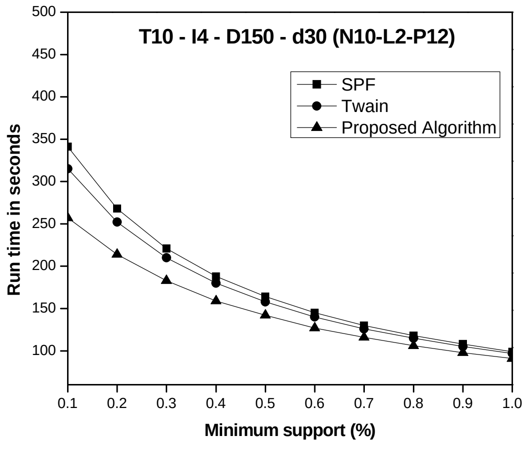


Figure 11. The run time under various minimum support

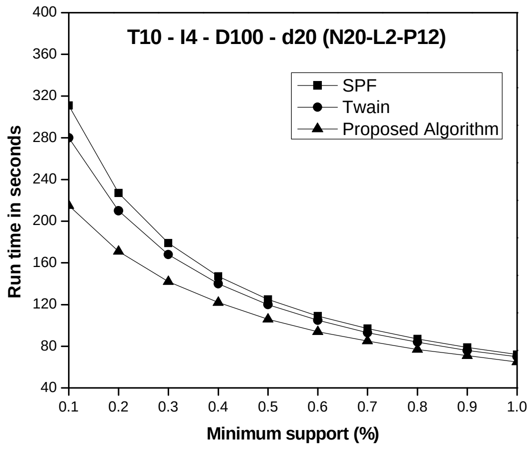


Figure 12. The run time under various minimum support

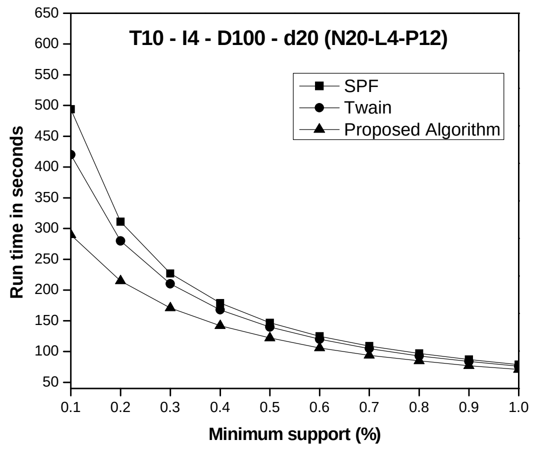


Figure 13. The run time under various minimum support

From these Figures, it can be noted that the margin grows as the minimum support decreases. This is due to the large number of frequent itemsets produced with the low support thresholds while this number decreases with high support thresholds. Figure 14 shows the time needed by each algorithm to generate all the candidate itemsets. It can be seen from the Figure that the proposed algorithm reduces significantly the time needed to generate the candidate itemsets with respect to the other two algorithms. In addition, Figure 15 shows 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 increment databases. Three different minimum support thresholds are considered in these experiments: 0.3%, 0.5% and 1%. Figure 16 shows the scale-up performance of the proposed algorithm as the size of the original database increases while the size of the increment database is fixed. Also, Figure 17 shows the scalability of the algorithm by varying the size of the increment 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.

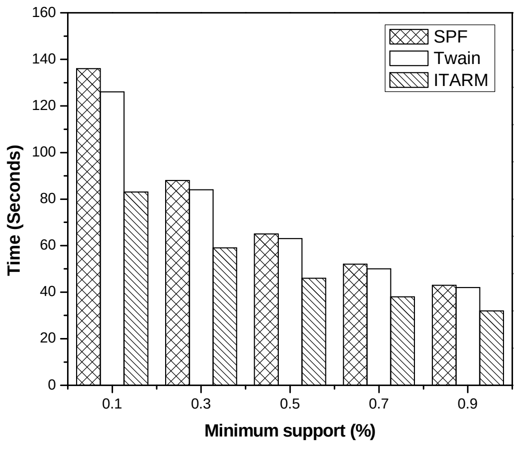


Figure 14. The time needed to generate candidate itemsets

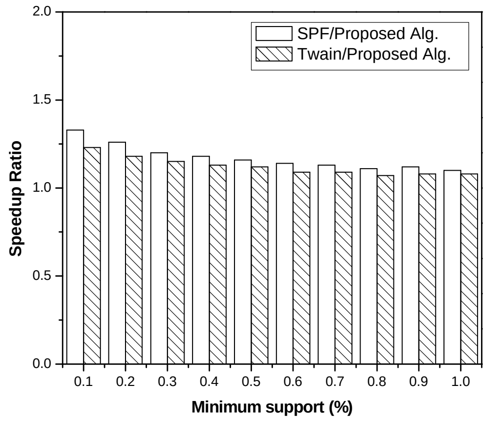


Figure 15. Speedup ratio

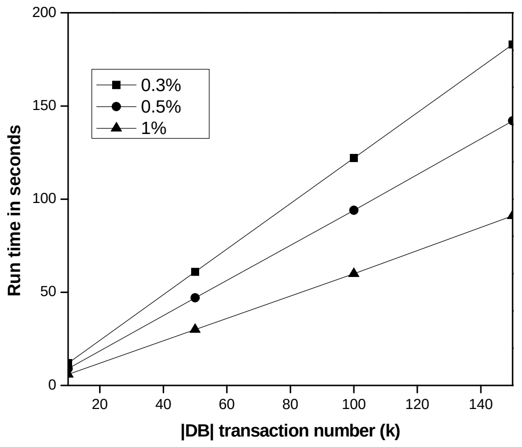


Figure 16. Scalability with the number of transactions in DB

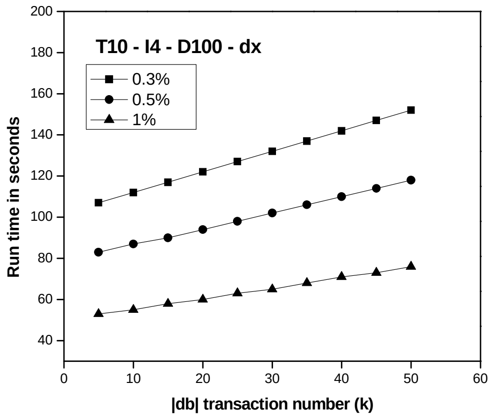


Figure 17. Scalability with the number of transactions in db

6. 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.

References

[1] Qiankun Zhao, and 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, and Ming-Syan Chen, “Twain: Two-End Association Miner with Precise Frequent Exhibition Periods,” *In the ACM Transactions on Knowledge Discovery from Data*, Volume 1, Number 2, Article 8, August 2007.

[3] R.C. Agarwal, C.C. Aggarwal, and V.V.V. Prasad, “A Tree Projection Algorithm for Generation of Frequent Item Sets,” *In the Journal of Parallel and Distributed Computing*, Volume 61, Number 3, pp. 350-371, March 2001.

[4] Jiawei Han, Jian Pei, and Yiwen Yin, “Mining Frequent Patterns Without Candidate Generation,” *In Proceedings of the ACM-SIGMOD International Conference on Management of Data*, Dallas, Texas, USA, pp. 1-12, May 2000.

[5] Gosta Grahne, and Jianfei Zhu, "Fast Algorithms for Frequent Itemset Mining Using FP-Trees," *In the IEEE Transactions on Knowledge and Data Engineering*, Volume 17, Number 10, pp. 1347-1362, October 2005.

[6] Hui Ning, Haifeng Yuan, and 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, Volume 2, pp. 739-742, June 2006.

[7] Yingjiu Li, Peng Ning, Xiaoyang Sean Wang, and Sushil Jajodia, "Discovering Calendar-Based Temporal Association Rules," *In Proceedings of the 8th International Symposium on Temporal Representation and Reasoning*, Cividale del Friuli, Italy, pp. 111-118, June 2001.

[8] Chang-Hung Lee, Ming-Syan Chen, and Cheng-Ru Lin, “Progressive Partition Miner: An Efficient Algorithm for Mining General Temporal Association Rules,” *In the IEEE Transaction on Knowledge and Data Engineering*, Volume 15, Number 4, pp. 1004-1017, July/August 2003.

[9] Wei Wang, Jiong Yang, and Richard Muntz, “TAR: Temporal Association Rules on Evolving Numerical Attributes,” *In Proceedings of the 17th International Conference on Data Engineering (ICDE01)*, Heidelberg, Germany, pp. 283-292, April 2001.

[10] Vincent Ng, Stephen Chan, Derek Lau, and 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, pp. 123 - 132 , January 2007.

[11] Chang-Hung Lee, Cheng-Ru Lin, and 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, pp. 263-270, November 2001.

[12] Junheng-Huang, and Wang-Wei, “Efficient Algorithm for Mining Temporal Association Rule,” *In the International Journal of Computer Science and Network Security (IJCSNS)*, Volume 7, Number 4, pp. 268-271, April 2007.

[13] Cheng-Yue Chang, Ming-Syan Chen, and 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, pp. 59-66, December 2002.

[14] Abdullah Uz Tansel, and 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, pp. 371-376, April 2007.

[15] Hui Ning, Haifeng Yuan, and 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, Volume 2, pp. 739-742, June 2006.