Journal of Computer and Knowledge Engineering, Vol. 2, No.2, 2019.
DOI: 10.22067/cke.v2i2.81586

PLI-X: Temporal Association Rules Mining in Customer Relationship Management Systems

Research Article

Mohammad Reza Keyvanpour[1*] Soheila Mehrmolaei[2] Atekeh Etaati[3]

Abstract. The temporal association rules mining has recently become an important technology in the field of the Customer Relationship Management (CRM), which can be useful for improving the customer enterprise relationship. Also, the dynamic nature of the CRM systems is made necessity of using efficient and rapid algorithms in order to extract valid patterns in this field. Hence, this paper proposes an efficient algorithm of incremental mining for temporal association rules in CRM entitled PLI-X. The four significant features that are considered for this algorithm are:(1) generating valid temporal association rules after adding the new transactions to the database, (2) performing algorithm on the whole temporal database instead of a small section of it, (3) performing the temporal transactional databases of the non-numeric, and (4) quickly generating the temporal association rules and reducing the run time by partitioning the candidate itemsets based on the previous partitions and scanning database when scan is necessary. Experimental result is the valid proof for the correctnessof this assertion. It seems that the PLI-X algorithm can be used as a strong tool in order to extract valid patterns and discover useful temporal association rules in the field of CRM.

Keywords. Temporal Database, CRM, Incremental Mining, Pre-large Itemsets.

1. Introduction

In the last decade, the advent of various high-level technologies has brought forth difficulty in reducing the price of most products. In addition, the growing trend of the universal economy causes more challenging contests in the market. In such challenging situations, the concentration of enterprises is on the customers. In any enterprise, all the customers have not the same value. Thus, it is impossible to assign the same resources to each customer due to limitations in resources. These features indicate the necessity of customer value analysis in the CRM [1]. Enhancement of customer-enterprise relationship, offering strategies for the development of new customers, and the maintenance of loyal customers are some of the duties of CRM [2]. Customer value analysis and capability improvement of customer preservation of an enterprise are the main goals of CRM. Consequently, the CRM is effective in decision-making about which customers are more important, which group of customers should be intrinsically considered, and which services are required for each customer group [3].

The data mining is responsible for knowledge discovery, rules , and hidden patterns from the stored data in the computer [4]. The extracted information could be used to predict the precise and correct behavior of the customers. Generally, data mining analyzes an extensive amount of unstructured data in order to discover the relationships for a better understanding of fundamental processes. The temporal data mining (TDM) does a similar analysis for an ordered data stream with temporal dependencies [5]. TDM is one step of the knowledge discovery process that extracts available structures such as temporal patterns or models from the data. In other words, algorithms that generate temporal patterns or create appropriate models are termed as temporal data mining algorithm [6]. In fact, the ultimate purpose of TDM is to discover hidden relationships between sequences and sub-sequences of events [7, 8]. So far, various methods have been presented in the field of data analysis [9-12] and its mining [13-17]. But, the volume of the performed studies is few in terms of developing a temporal mining-based structure in CRM systems and this lack can pose a challenging problem in CRM systems. On the other side, most of the available methods are incapable of correcting the analysis of dynamic temporal data and there is no effective method that resolves all the required aspects of CRM systems. Consequently, the main purpose of this study is to suggest a method based on the incremental mining of temporal data. In fact, it seems that the proposed method can be capable for the correct analysis of dynamic temporal data and enhances CRM system performance.

The paper is structured as follows: this research starts with Section 2 that gives a brief review of the topic background. Then, it expresses incremental mining of temporal association rules by "*pre-large*" itemsets at a glance in Section 3. In Section 4, the proposed method is presented and thoroughly described. The experimental results are discussed in Section 5. Finally, conclusions of research work are remarked in Section 6 and future research is presented in this section.

2. Review of the Related Literature

Association Rules Mining (ARM) has two components of

Manuscript received June,27, 2019; accepted February, 11, 2020.

[1*] M. R. Keyvanpour, Associate Professor, Dep. Of Computer Engineering, Faculty of Engineering, Alzahra University, Tehran, Iran. Email:keyvanpour@alzahra.ac.ir

[2] S. Mehrmolaei, MSc, Data Mining Lab, Dep. Of Computer Engineering, Faculty of Engineering, Alzahra University, Tehran, Iran. Email: s.mehrmolaei@gmail.com.

[3] A. Etaati, MSc, Dep. Of Computer Engineering, Islamic Azad University, Qazvin Branch, Qazvin, Iran. Email: a.etaati@qiau.ac.ir

finding frequent itemsets and generating association rules. The major part of the algorithms is considered to discover frequent itemsets, which is highly time consuming while generating association rules is straightforward [13].

Besides, conventional algorithms of association rules mining shows that there are other enhanced algorithms that having incremental, multi-level rule, multi-dimensional rule, Temporal Association Rule Mining (TARM), and so on [18-23].

Temporal Data Mining (TDM) is a fairly modern branch which can be considered as the common interface of various fields, namely statistics, temporal pattern recognition, temporal databases, optimization, visualization, and high-level and parallel computations. TDM is different from traditional modeling techniques of a data stream in the size and nature of data set and the method of data-series gathering. The reason for this difference can pose two important points: the first point, the incapability of traditional modeling techniques in handling large data sets, while the size of the data set is large in TDM [5]; the second point, difference in type of knowledge discovered by TDM and techniques of data-series analysis. In the other word, the main problem is pattern discovery form the sequential data in TDM [24].

TARM is recently taken into account as a method for dynamic data processing such as transactional databases in CRM systems. An interesting extension to association rules is to include a temporal dimension. Actually, different association rules is discovered if different time intervals are considered. The lifetime of an item (such as egg, coffee, tea) is a time interval that originates from the occurrence of that item in the database and continues as long as its presence [22]. In other words, it is the time interval that an item is accessible for purchasing.

So far, different techniques have been presented for TARM in various applied fields and science the review of which indicates the dynamic growth of this research scope and various approach in this field [1, 3, 4, 14, 16]. On the other hand, literature indicates that most of TARM 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 [25].

Also, most of the previous algorithms cannot effectively be applied in the temporal databases because of two important parameters i.e. confidence and support coefficients which should be modified based on the new mining model [26].

In this section, more related recent articles are reviewed for the better comprehension which are coherently classified in Fig. 1.

2.1 Candidate generation category

In this category of algorithms such as the Apriori algorithm, the algorithm needs to scan the database repeatedly [16]. The general act of the search process performed is briefly stated level to level as it follows:

I. Let k=1
II. Generate frequent itemsets of length 1.
III. Repeat until no new frequent itemsets are identified

- Generate length ($k+1$) candidate itemsets from length k frequent itemsets.
- Prune candidate itemsets containing subsets of length k that are infrequent.
- Count the support of each candidate by scanning the database.
- Eliminate candidates that are infrequent leaving only those that are frequent.

Some of the algorithms of this category are covered in the following:

Sornalakshmi et al. in [12] reported that Apriori algorithm generates a large amount of rules and does not guarantee the efficiency and value of the knowledge created. Hence, they have proposed an Enhanced Apriori Algorithm (EAA) based on the knowledge of a context ontology methodology for sequential minimal optimization in order to overcome the weakness of the standard Apriori algorithm. Authores have said that the EAA to generate frequent k-itemsetsfinds finds the frequent itemsets directly and eliminates the infrequent subsets based on the standard Apriori algorithm.

Wang and Zheng is [16] proposed an improved Apriori algorithm of frequent itemset that gives the time constraints interval and uses the time interval algebra to filter and mine the data in the transaction data. The authors have said that our algorithm can be an effective method to reduce the transaction is given. For this purpose, their method reduces the number of candidate sets and improves the efficiency of the Apriori algorithm, but it also needs to scan the database repeatedly.

Kadir et al. [28] believed that most of the used systems to extract the existing temporal relation among temporal data suffers from sparseness of the available dataset such as market basket datasets. They have used Apriori algortihm to extract temporal relations in such data, which include two main steps: (1) extracting features from the dataset and (2) vectorizing the features so that Apriori algorithm can be applied on the data. In the end, the Apriori algorithm is used to generate frequent itemsets.

Maragatham and Lakshmi in [29] proposed an efficient algorithm, which mined temporal association rules based on Utility or value, namely UTARM. Authors have said that the UTARM algorithm combines both temporal (time periods) and utility for the mining of remarkable and helpful association rules. Actually, the different utility values are given for the items based on the time periods in the UTARM algorithm in a seprate table for each partition. This algorithm can be decomposed into seven steps: (1) generate all possible candidate 2-itemsets from partition $P1$, (2) mining of FTU 2-itemsets ($P1$), (3) generate candidate 2-itemsets from partition $P2$ and mining of FTU 2-itemsets ($P1+P2$), (4) mining of FTU 2-itemsets ($P1+P2+...+Pn$), (5) generate all FTU 1-itemsets from FTU 2-itemsets, (6) mining of all FTU k-itemsets, and (7) generate association rules using FTU itemsets.

Hong et al. in [30] presented the TPPF algorithm (three-phase algorithm with predicting strategy considering the first occurring transactions of items). They have introduced a new concept of temporal association rule mining with a hierarchy of time granules to find hierarchical temporal association rules in temporal databases, and they also presented an effective approach to find such rules. In particular, an effective strategy is designed to predict the upper-bound of support values for itemsets. The strategy can be used to remove unpromising itemsets at an early stage in the process, and the proposed TPPF can effectively reduce the computational cost of scanning a temporal database.

.2 Without candidate generation category

This category of algorithms such as the FP-growth tree algorithm applies a radically different approach to discover frequent itemsets. The algorithm does not subscribe to the generate-and-test paradigm of the Apriori. Literature shows that the general act of this category is considered to encode the data set using a compact data structure called an FP-tree and extracts frequent itemsets directly from this structure [31]. This category reduces infrequent items and is also much faster than Apriori algorithm. We have reviewed some of the algorithms of this category as follows:

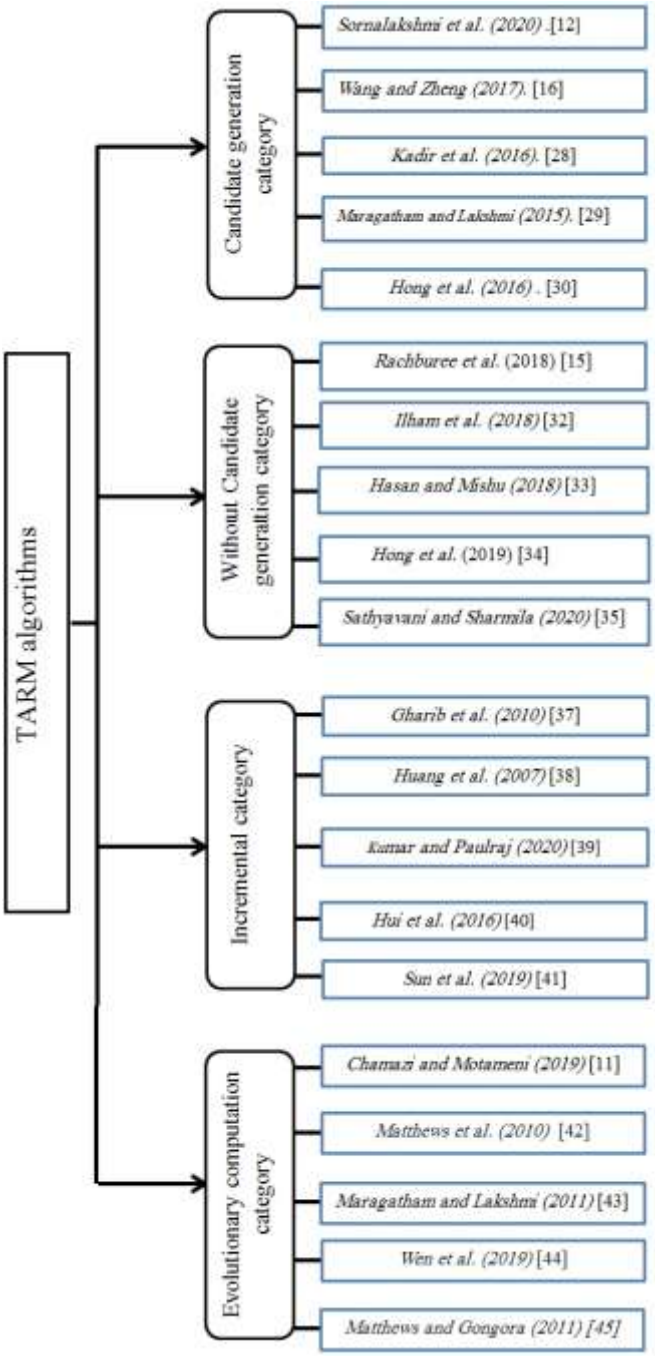

Fig 1. An overview of more related works for TARM

Rachburee et al. in [15] used apriori algorithm and FP-growth to discover association rules mining from maintenance transaction log of ATM maintenance. They have tried to focus on comparison of association rules between FP-growth and apriori algorithm. In the end, they have concluded that FP-growth has better execution time than apriori algorithm.

Ilham et al. in [32] performed the market basket analysis to determine the layout and planning of goods availability by the FP-growth algorithm. Also, they have proved the successful application of the FP-growth algorithm in generating informative association rules and found out the consumer spending pattern. They have reported that experimental results show that the FP-growth algorithm can analyze quickly and efficiently informing consumer shopping pattern and can increase revenue.

Hasan and Mishu in [33] believed that there exists a problem to define minimum support and to mine frequent itemsets on Apriori and FP-growth algorithm. If the threshold is set to low, too many frequent itemsets will be generated which may cause the Apriori and FP-growth *algorithm* to become inefficient or even loss of memory. Hence, authores have proposed an adaptive method based on Apriori and FP-growth algorithm to avoid this problem by using Binomial Distribution (BD) to find appropriate minimum support adaptively. They have shown that their method performs better than existing benchmark.

Hong et al. in [34] designed a tree structure from the frequent-pattern tree, then, presented a mining algorithm based on it to extract high temporal fuzzy utility patterns from temporal transactional datasets. They have said that the proposed method requires two-phase processing to find all high temporal fuzzy utility itemsets and to provide better results than the Apriori-based mining algorithm.

Sathyavani and Sharmila in [35] presented that the mining of UP-Tree (utility pattern) by FP-growth extracts high utility itemset and generates too many candidates. Hence, authors have proposed using the UP-Growth and UP-Growth$^+$ to shorten the candidate itemsets. In UP-Growth, two tactics such as Discarding Local Unpromising items (DLU) and Decreasing Local Node (DLN) were used in FP-growth to effectively reduce the memory usage. Authors have reported that these algorithms can overcome the spatial and temporal locality problem and effectively reduce memory usage.

2.3 Incremental category

Temporal transactional databases are continuously updated and increased. Thus, the rules that have previously been generated need to be updated, removing those rules that are no longer relevant, and adding valid new rules [36]. Hence, incremental mining concept is presented to mine temporal association rules, which can help to solve this problem. Some of the studied incremental algorithms are briefly stated as it follows:

Gharib et al. [37] proposed ITARM (incremental association rules mining) algorithm for updating temporal association rules in the transaction database. Authors have believed that the proposed ITARM algorithm reduces the time needed for generating new candidates by storing candidate 2-itemsets. In fact, they have presented a technique to update the previously generated candidates instead of re-generating them from scratch. Also, they have reported that the experiments results show a significant improvement over the traditional approach ofmining the whole updated database.

Huang et al. [38] presented Twain algorithm that progressively calculates the number of the occurrence of two-item candidates in each partition of the database. The Twain algorithm uses a progressive filtering method for the elimination of non-iterative two-item candidates. Authors have believed that the Twain algorithm generates the iterative two-item candidates after one scan of database and, then, directly creates the k-item candidates from the iterative two-item candidates. Also, the second scan of database gives the number of occurrence of the last item-sets and generates the iterative temporal item-sets.

Kumar and Paulraj in [39] presented a pattern mining algorithm based on incremental utility to identify the optimal patterns in a relational database. Authors have considered that frequent patterns are selected based on the minimum support and confidence where the next level pattern are generated based on the frequency of patterns in the selected set, which are measured iteratively. Also, they have reported that this algorithm improves to access the scalability and efficiency of transactional processing itemset to improve the knowledge enhances itemsets by identifying the process.

Hui et al. [40] presented an efficient algorithm, namely Inc_TPMiner (Incremental Temporal Pattern Miner) to mine incremental of temporal patterns from interval-based data. Authors have believed that the proposed algortihm can be useful to balance the efficiency and reusability based on a proper expression, dynamic representation.They have reported that the experimental results on the tested databases indicate that Inc_TPMiner significantly outperforms re-mining with static algorithms in execution time and possesses graceful scalability.

Sun et al. in [41] proposed a incremental mining algorithm for frequent itemsets using a Full Compression Frequent Pattern Tree (FCFP-Tree), which is named FCFPIM. Authors have said that FP-tree and the FCFP-Tree structures maintain complete information of all the frequent and infrequent items in the original dataset. But the act of FCFPIM algorithm is differing as it does not allows to waste any scan and computational overhead for the previously processed original dataset when the new datasets are added and the support changes. They have reported that the experimental results show that the space-consuming is worthwhile to win the gain of execution efficiency, especially in the situation that the support threshold is low.

2.4 Evolutionary Computation Category

In the past few years, ARM techniques based on Evolutionary Computation (EC) have emerged as one of the most popular

research areas for addressing the high computation time of traditional ARM [21]. In the last years, application of the EC algorithms have appeared in TARM problems to address the limitations of traditional approaches such as high computation time in different applied domains. The EC algorithms are a state-of-the-art and efficient strategy for finding nearoptimal solutions. A key characteristic of these algorithms is that strict termination conditions can be set to limit computation time while a nearly optimal solution can be obtained [46].

We tried to cover some works in which evolutionary computation is placed as it follows:

Chamazi and Motameni in [11] proposed combination of fuzzy temporal mining concepts and EC algorithms to identify temporal frequent itemsets. In fact, authors have designed an efficient fuzzy temporal-evolutionary mining based on the bees algorithm. This approach finds suitable membership functions for fuzzy temporal mining problems by the bees algorithm before searching for temporal frequent itemsets and fuzzy associations. The authors have reported the proposed approach provided for good performance with respect to the effectiveness of the obtained solution.

Matthews et al. (2010) [42] presented a new framework in which genetic algorithm is introduced as an impact factor for temporal association rules mining. Authors have asserted that their framework is an enhancement to existing temporal association rule mining methods as it employs a genetic algorithm to simultaneously search the rule space and temporal space.

Maragatham and Lakshmi in [43] presented an effective method based on the Utility for temporal association rule mining. Authors have proposed that the Particle Swam Optimization algorithm is used to optimize the generated rules by filtering out the redundant rules and, thereby, reducing the problem space. They have considered calculation of the support and confidence from the input data, the rule generation, initialization , updation of the velocity, position of the rules, and evaluation of fitness function as main processes.

Wen et al. in [44] proposed temporal association rules mining algorithm based on Genetic algorithm that is designed to extract temporal association rules in traffic environments. The rules are analysed by a classification mechanism so that a classifier can be built to predict the traffic congestion level. They have reported that experimental results demonstrate high and reasonable accuracy of output.

Matthews and Gongora in [45] presented a novel method for mining association rules that are both quantitative and temporal using a multi-objective evolutionary algorithm. Authors have reported that their method successfully identifies numerous temporal association rules that occur more frequently in areas of a dataset with specific quantitative values represented with fuzzy sets. Also, they have said that the novelty of this method lies in exploring the composition of quantitative and temporal fuzzy association rules.

As a result, it can be said that the common approach in many of the previous algorithms is to scan throughout the transactional database; whereas, this paper proposes a novel algorithm of incremental mining for temporal frequent itemset that prevents the complete scan of the database in each stage. For this purpose, the algorithm performs the scan act only when it is necessary. In many previous techniques, association rules mining has been performed on a part of the transactional database in a certain time. On the other hand, users of dynamic systems damand methods that discover customer's behavior patterns which provides the minimum time possible. In that case, current methods could not respond completely to the users' needs due to the various scans of the database for discovering association rules. Also, reviewing the literature, it is concluded that the research effort is few in the field of temporal association rule mining in dynamic transactional systems.

3. Incremental Mining of Temporal Association Rules by Pre-Large Items

This section is organized to define and introduce basic concepts of the proposed algorithm into two parts: preliminary concepts and association rules maintenance.

3.1 Preliminary Concepts

This paper employs the concepts of temporal granularity for database partitioning. Temporal granularity is a partition of the timeline. In the context of databases, a temporal granularity can be used to specify the temporal qualification of a set of data, similar to its use in the temporal qualification of statements in natural languages. For example, in a relational database, the time stamp associated with an attribute value or a tuple may be interpreted as associating that data with one or more granules of a given temporal granularity (e.g., one or more days) [47, 48]. In a temporal database, each tuple has two attributes, start and end, which can indicate the time period during which the information recorded in the tuple is valid. A tuple might also have many other attributes. A transaction originl database, DB = *{T_1, T_2, …, T_C}* is a set of transactions where each transaction T_d ($1 \le d \le c$) has a unique identifier, called T_{id}. Given a finite set of items I={i_1, i_2, ..,i_m}. Assume that n is the number of the database partitions based on a temporal partitioning parameter such as month, season, year, and etc. Also, $db^{s,e}$ denotes a partition of original database that is originated from partition p_s and ends in partition P_e , such that $\left|db^{s,e}\right| = \sum_{h=s,e} \left|p_h\right|$. Where $db^{s,e} \subset DB$ and $\left|p_h\right|$ denotes the number of transactions in partition p_h. An itemset X is a set of distinct k items {i_1, i_2, …, i_k}, where $i_j \in I, 1 \le j \le k$, k is the size of itemset X [43]. The interval (s,e) represents maximal lifetime or Maximal Common exhibition Period (MCP). A maximal temporal itemset $X^{s,e}$ *is defi*ned as follows [28]:

Definition 1: An itemset $X^{s,e}$ is called a maximal temporal itemset in a partial database $db^{s,e}$ if s is the latest starting partition number of all items belonging to X in database *DB*

and *e* is the partition number of the last partition in $db^{s,e}$ retrieved.

A temporal itemset $z^{s,e}$ is called a temporal Sub-Itemset (*SI*) of a maximal temporal itemset $X^{s,e}$ if $z \subset X$ [28]. As an example, consider the maximal temporal itemstes $BDE^{2,3}$ that contains the sub-itemsets {$DE^{2,3}$, $BE^{2,3}$, $BD^{2,3}$, $E^{2,3}$, $D^{2,3}$, $B^{2,3}$}. Let MCP (*X*) denote the MCP value of item *X*. The MCP value of an itemset *X* is the shortest MCP among the items in itemset *X*.

The fraction of transaction *T* supporting an itemset *X* with respect to partial database $db^{s,e}$ is called the support of $X^{s,e}$ which is given by the following equation [25]:

$$supp\left(x^{MCP(x)}\right)=\frac{\left|\left\{T \in db^{MCP(x)} \mid x \subseteq T\right\}\right|}{\left|db^{MCP(x)}\right|} \quad (1)$$

The support of an itemset *X* is an indication of how frequently that *X* appears in database *DB*. The support value of *X* with respect to *T* is defined as the proportion of itemsets in a database containing *X*, denoted as supp (*X*) [43]. The support and confidence of a rule $(X \Rightarrow Y)^{MCP(XY)}$ are defined as follows [28, 43]:

$$supp\left(\left(X \Rightarrow Y\right)^{MCP(XY)}\right)=supp\left(\left(X \cup Y\right)^{MCP(XY)}\right) \quad (2)$$

$$conf\left(\left(X \Rightarrow Y\right)^{MCP(XY)}\right)=\frac{supp\left(\left(X \cup Y\right)^{MCP(XY)}\right)}{supp\left(\left(X\right)^{MCP(XY)}\right)} \quad (3)$$

Definition 2: An association rule $\left(X \Rightarrow Y\right)^{MCP(XY)}$ is called a general temporal association rule in the transaction set DB with [37]:

$$conf\left((X \Rightarrow Y)^{MCP(XY)}\right)=c \quad (4)$$

$$SUPP\left((X \Rightarrow Y)^{MCP(XY)}\right)=s \quad (5)$$

$$(X \Rightarrow Y)^{MCP(XY)} \in db^{MCP(XY)} \quad (6)$$

In which transaction itemsets *X* and *Y* have relative support and confidence greater than the corresponding thresholds. Thus, we have the following definition to identify the frequent general temporal association rules.

Definition 3: A general temporal association rule $\left(X \Rightarrow Y\right)^{MCP(XY)}$ is termed to be frequent if and only if:

$$SUPP\left((X \Rightarrow Y)^{MCP(XY)}\right) > \min\_SUPP \quad (7)$$

$$conf\left((X \Rightarrow Y)^{MCP(XY)}\right) > \min\_conf \quad (8)$$

Actually, temporal association rule with particular MCP is known as frequent if and only if its support coefficient is not less than the upper support threshold and its confidence coefficient is not less than the minimal user-defined confidence coefficient [49].

Generally, the temporal association rules discovery can be resolved into three steps [22, 49]:

1) Generate all frequent maximal temporal itemsets (*TIs*) with their support values.

2) Generate the support values of all corresponding temporal subitemsets (*SIs*) of frequent *TIs*.

3) Generate all temporal association rules that satisfy min confusing the frequent *TIs* and/or *SIs*.

3.2 Association Rules Maintenance

This section shows the method of association rules maintenance after updating temporal data in a CRM system. The proposed method uses pre-large item-sets concept that is not initially large but is expected to convert into a large set in the future. The pre-large item-sets work like a buffer aiming to reduce the number of direct item transfers from large to small item-sets and vice versa. Pre-large item-sets are defined using a lower support threshold and an upper support threshold to reduce the need for rescanning the original databases and to save the maintenance costs. Pre-large itemsets act like gaps that reduce the movements of itemsets directly from large to small and vice-versa [50]. The upper support threshold is the support coefficient that is applied in usual algorithms of association rules mining. An item locates in the large item-sets when its support coefficient is more than the upper support threshold. On the other side, the lower support threshold is a minimal support that isolates the small and pre-large item-sets. An item with smaller support than the lower support threshold is considered as a small item. The items with support between the lower and upper support thresholds locate in the pre-large item-sets. Adding a new transaction into the original database could result in nine different states [51] that are shown in Fig 2.

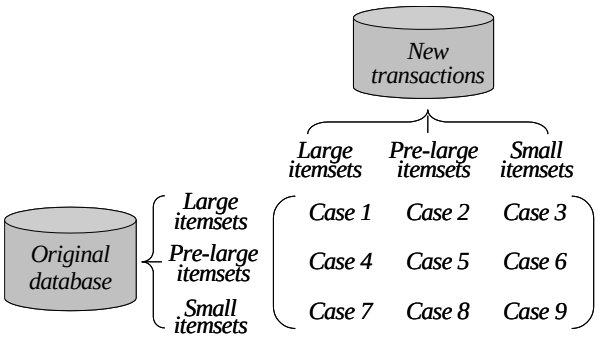

Fig 2. The probable states after addition of a new transaction into the original database [51]

Also, Table 1 summarizes the nine mentioned cases with their results. According to the average number of occurrences in cases 1, 5, 6, 8 and 9, these cases could not alter the final discovered rules. Cases 2, and 3 could change the existing association rules and remove the validity of some rules while cases 4 and 7 could add some new association rules to the old rules. If all locating items within the large and pre-large item-sets and the count of their occurrence are preserved after the execution of each stage, then the cases 2, 3 , and 4 are simply

managed. In the maintenance stage, the new transactions are usually very small in comparison with the old transactions. In this situation, the items in case 7 could not be located in the large item-set in the updated database.

For the proposed algorithm, all symbols used in the proposed algorithm are shown in Table 2.

4. The Proposed Method

In this section, we would like to present an efficient algorithm for incremental mining of temporal association rules on temporal customers database in a CRM system, which is called PLI-X. General schema of PLI-X algorithm is shown in Fig. 3. This algorithm is also applicable to situation that the new transactions are inserted in the database. The PLI-X algorithm uses the concept of pre-large itemsets and generates the dominant rules in a dynamic space on the basis of the Fast Update algorithm (FUP). This algorithm generates up-to-date and valid rules that are beneficial factors for enterprises in a short period. In fact, the main goal of the PLI-X algorithm is keeping the repeated temporal patterns after updating temporal transactions of database.

There is a significant difference among PLI-X algorithm and other previous works concerning the incremental mining of temporal frequent itemsets in dynamic systems, which are clarified as follows:

Reviewing the literature, it is concluded that most the previous resources in the field of incremental mining of temporal data in dynamic systems require a complete scanning of the whole updated database in the situation that new transactions are added to the original database [26, 28]. But, the scanning process performs in a the different way by PLI-X algorithm, which leads to a reduction in run-time because of scan number reduction. The proposed algorithm explores the whole database in particular conditions. For this propose, a safe threshold is presented to consider the newly added transactions. The appropriate definition of this threshold can lead to reducing scanning updated database. Thus, PLI-X algorithm performs the scan of the updated database due to the safe threshold after inserting several new transactions only when database scan is necessary. This schema not only preserves the previous frequent itemsets after updating the temporal database but also enhances the efficiency of incremental mining algorithm and saves the execution time of the proposed algorithm.

Table 1. Summary of the results of the mentioned cases

Cases: Original – New	**Results**
Case 1: Large – Large	Always large
Case 2: Large - Pre-large	Large or pre-large
Case 3: Large - Small	Large or pre-large or small
Case 4: Pre-large - Large	Pre-large or large
Case 5: Pre-large - Pre-large	Always pre-large
Case 6: Pre-large - Small	Pre-large or small
Case 7: Small - Large	Pre-large or small when the number of transactions is small
Case 8: Small - Pre-large	Small or Pre-large
Case 9: Small - Small	Always small

Table 2: sympols description of the proposed algorithm

#	**Symbol**	**Definition**	#	**Symbol**	**Definition**
1	DB	transaction originl database	12	*SIs*	temporal subitemsets
2	T_{id}	a transaction with *id* identifier	13	*Tdbi*	the added transactions to database without any changes
3	I	a finite set of items	14	*dbi*	the part of incremental database
4	p_s *and* p_e	start partition and end partition, respectivelly	15	*Merged db*	the sum of the added transactions and transactions in the last partition of the original database.
5	$db^{s,e}$	a partition of DB that is originated from partition p_s to partition P_e	16	*Ts (Tdbi*)	the time stamp of new transactions
6	$\lvert p_h \rvert$	the number of transactions in partition p_h	17	*Ts (DBj*)	the time stamp of transactions in the last partition of the original database
7	*X*	itemset	18	*DBj*	the part of the original database which inlcudes transactions in the last partition
8	$X^{s,e}$	a maximal temporal itemset in a partial database $db^{s,e}$	19	*TDBj*	transactions in the last partition of the original database
9	*k*	items number in the items sequence	20	*f*	safe threshold
10	*TIs*	frequent maximal temporal itemsets	21	*TDB*	transactions of the original database
11	s_u	upper support threshold	22	s_l	lower support threshold

We have briefly listed the paper innovations as follows:

- generating of the valid temporal association rules after adding the new transactions to the database.
- performing of algorithm on the whole temporal database instead of a small section of it
- performing the temporal transactional databases of the non-numeric
- generating the temporal association rules and reduction of the run time by partitioning the candidate itemsets based on the previous partitions and scanning database when the scan is necessary.
-

4.1 Pre-processing Step

The first step of the proposed method is pre-processing the flowchart of which is shown in Fig 4. In this step, customer's transactions are partitioned based on a time stamp like a month of the year that purchase has done on that month in the original database. As it is shown in Fig 4, the pre-processing step is based on a comparison between new transactions and the older transactions in the last partition of the original database. On this basis, two different situations could have occurred: *i. Ts (Tdbi*) is identical to time stamp of transactions in the last partition of the original database (*Ts (DBj*)). In this situation, the new transactions are merged to the transactions of the last partition of the original database and, then, are sent to PLI-X algorithm as an adjunct database. The number of the occurrence of common itemsets between the last partition of the original database and the newly added database should be subtracted from the extracted large and pre-large item sets. Since these items are converted into the items of the added database, their count should not be considered in the previously generated item sets. These items are investigated with the newly added transactions and are entered into the proposed algorithm. *ii.* The time stamp of new transactions is different from that of older transactions in the last partition of the original database. In this situation, the combination of transactions in the last section of the original database and new transactions is not required and the newly added transactions are entered into the algorithm as an adjunct database.

4.2 The PLI-X Algorithm

In the proposed algorithm, the temporal itemsets of maximized frequent generated after performing the nine steps is presented in flowchart (Fig 5). As is seen in Fig 5, research novelty is highlighted with a bold line that is in black color. Also, in the first step of the proposed method, *f* parameter is calculated due to the entered values. In the second step, 1 value is assigned to it *k* variable that indicates items number in the items sequence and is produced in the current execution algorithm. All k-items candidates and occurrences number of them in the newly added transaction are produced in the third step. The creation process of the new candidates is expressed briefly as it follows:

The large and pre-large items and their counts are stored from the last execution and are used for the preservation of association rules. The newly entered transactions are first scanned by the algorithm to generate the 1-itemset candidates. The outcome is then compared to the previously stored large and pre-large itemsets.

If a 1-item candidate is available in the new transactions of large or pre-large 1-itemsets of the original database, the final count of their occurrence could be calculated by adding count of their occurrence in the current and previous execution. This is the consequence of preservation of all previous large and pre-large items with count of their occurrence. Calculation of new support parameter helps to decide whether a large or pre-large item remains in the same category after inserting a new transactions or not. The support parameter is the ratio of total number of target items to the total number of transactions. If a new 1-item candidate does not pertain to large or pre-large item sets of the original database, it definitely could not be a large item for the updated database. This is true in the condition that count of newly added transactions is less than a safe threshold limit (f).

Consequently, in order to find the new pre-large itemsets during incremental insertion of new transactions, the database is scanned only when the count of new transactions exceeds the safe threshold limit. Scan of the original database is carried out similar to the implemented method in FUP algorithm. The next step is to create the 2-item candidates for the newly added transactions while the same procedure is repeated to find all large 2-itemsets. This process is continued until the generation of all the large itemsets. The *c* variable stores the number of new transactions since the last scan of the main database.

4.2.1 Optimization of Lower Support Threshold

In contrast to the previous studies, the present research considers the lower support to constraint itemsets and minimizes the execution time. The lower threshold coefficient is a key parameter in determining the association rules. This parameter separates the small and pre-large itemsets based on the number of their occurrence in the database. In all the previous resources in this field, this coefficient was manually set by the user. But, in this research, the optimal and desirable value of the lower support threshold is obtained by using the optimization process to enhance the efficiency of association rules discovery algorithms.

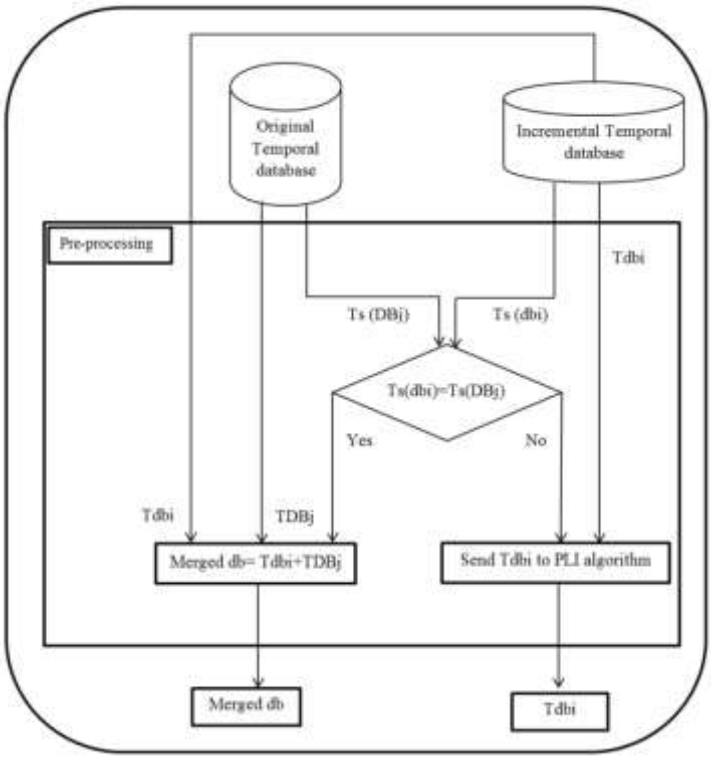

Fig 3. General schema for the proposed PLI-X algorithm

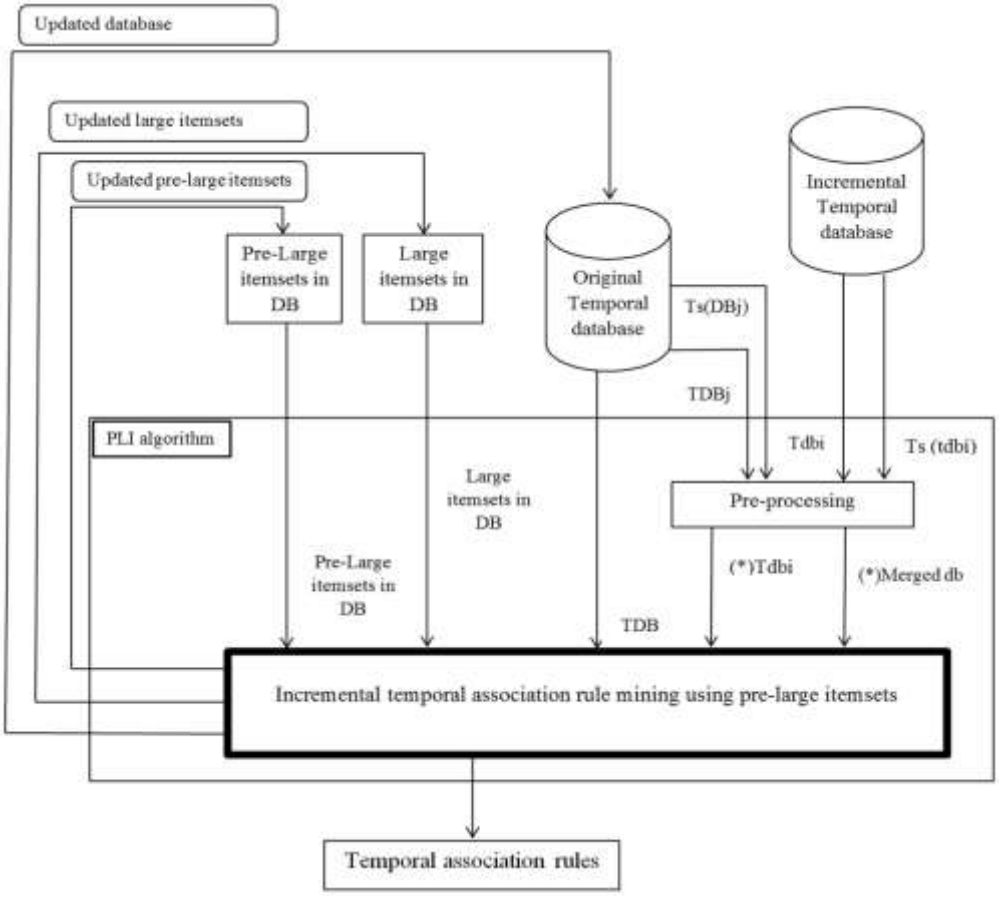

Fig 4. The flowchart of pre-processing step in PLI-X

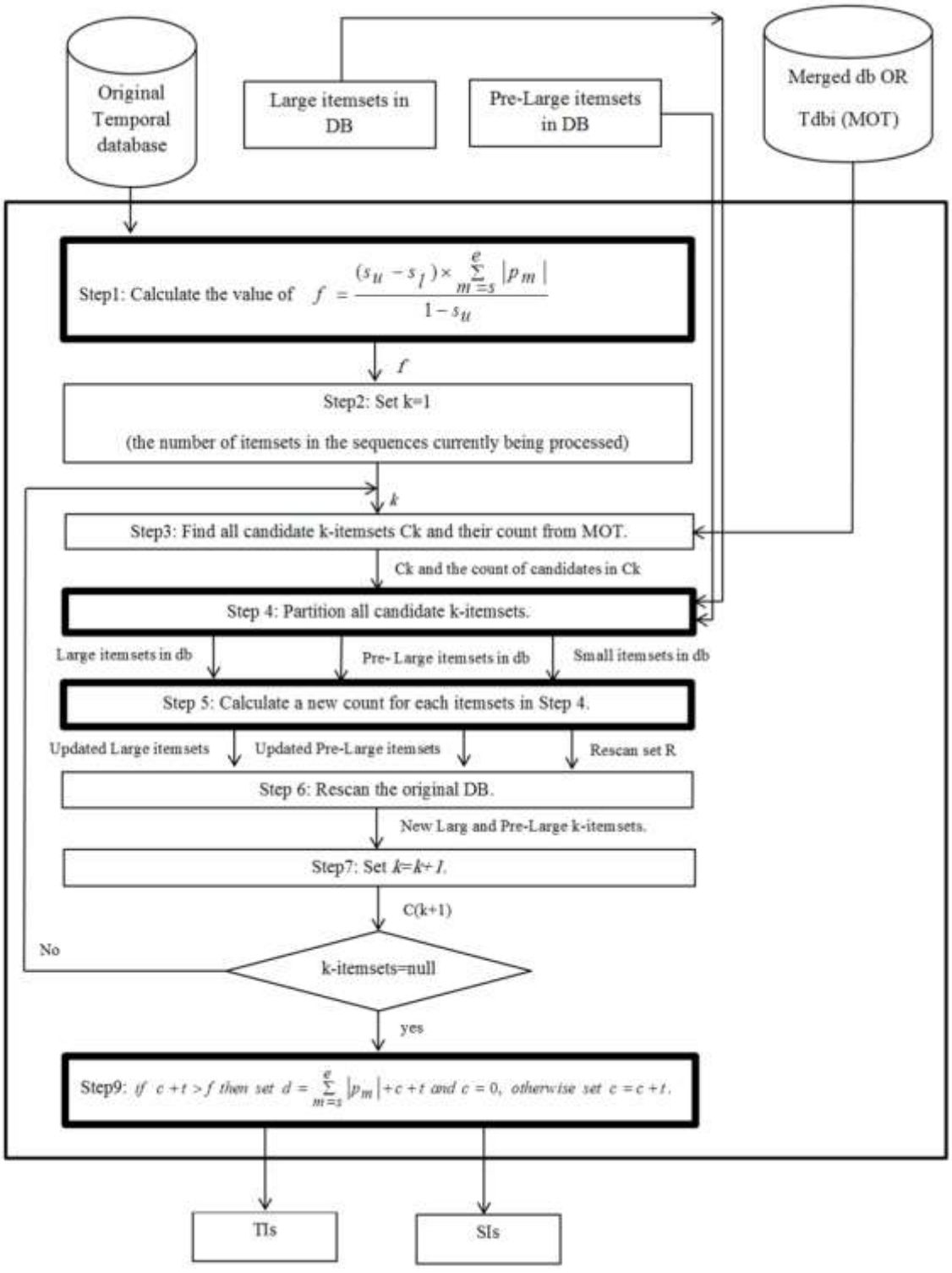

Fig 5. The generating temporal itemsets of the maximized frequent

Table 3. Optimal values of the lower support threshold for various values of the upper support threshold

Values	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
Upper-limit support threshold	10	20	30	40	50	60	70	80	90
Lower-limit support threshold	9	10	20	20	20	30	30	40	50
Run time (ms)	76.55	76.81	70	70	75	72	72	77	71

1) The test-based optimization

In order to determine the optimal value of lower support threshold, ten tests are performed and the execution time of the proposed algorithm is extracted. These tests cover various values of upper support threshold. The experiments results are shown in Table 3. In each test, the lower support threshold varies between zero and the upper support threshold value. The moderate safety factor of 50% is considered in our tests. Accordingly, the proposed test-based formula for the optimal lower support threshold is stated as follows:

$$S_l = 0.5\, S_u \tag{9}$$

This table shows that the optimal execution time is obtained when the lower support threshold is approximately one-half of the upper support thresholds. Performance of the proposed algorithm is the best in such a situation for association rules discovery.

2) Curve fitting-based optimization

The curve fitting technique could be applied to the results of our tests to obtain an approximate relation between the lower-limit and upper support threshold parameters. The curve fitting toolbox in MATLAB software is used for this purpose and the first, second, and third order polynomial models are employed as the fitting bases. In order to determine the unknown coefficients in these models, the error between actual tests (simulations) and models is minimized. The extracted first, second and third-order polynomial models are obtained as follows:

$$S_l = 0.4782 S_u + 1.6 \tag{10}$$

$$S_l = 0.002424 \left(S_u\right)^2 + 0.2115 S_u + 6.933 \tag{11}$$

$$S_l = 6.216 \times 10^{-6} \left(S_u\right)^3 + 0.001399 \left(S_u\right)^2 + 0.2588 S_u + 6.4 \tag{12}$$

***3) Mathematical proof-based optimiz*ation**

The optimal value of lower support threshold is one-half of the upper support thresholds. To prove this theorem, a general formula for expressing the lower support threshold as a function of the upper support threshold could be defined as $S_l = (m/n)\, S_u$ .Three conditions should be investigated for the ratio of m/n :

i. $m/n > 1$: Since the lower support threshold is always less than the upper support threshold, this situation is not meaningful and does not occur.

ii. $m/n = 1$: This means an equal value for both the lower and upper support thresholds which is not a practical situation as well.

iii. $m/n < 1$: This is the acceptable condition that considers the lower support threshold in the range of $(0, S_u)$.

Accordingly, the proof is continued based on the third situation. According to the theorem when the criterion $t < (S_u - S_l)d / (1-S)$ is satisfied, an item that pertains to small items in the incremental database could not be a large or pre-large item in the updated database. A larger value of $(S_u - S_l)d / (1-S)$ increases the probability for satisfaction of this formula and consequently, the resulted small itemsets do not require an examination which enhances the algorithm performance as these items do not require any examination. Therefore, the following relations could be written

$$S_l = \left(0, 1/2\right] S_u \Rightarrow S_l = \max\left(0, 1/2\right] S_u \Rightarrow$$
$$S_l = \frac{1}{2} S_u \Rightarrow m = 1, n = 2 \tag{13}$$

Again, the value of the lower support threshold is suggested to be one-half of the value of the upper support threshold. The proposed algorithm considers a safe threshold limit for the newly added transactions and reduces the number of scans to the minimum possible count.

5. Results and Discussions

This section is adjusted into four parts: the evaluation criteria, the test method, the implemented data set, and experimental results. Each part is expressed with more details in the following.

5. 1 Evaluation Criteria

The evaluation criteria of PLI-X are expressed as follows:

a) Execution time: the execution time in the generation of temporal association rules is measured by this measure in [29, 30] and compared with that of the previously presented algorithms. The execution time should be evaluated in various conditions after setting the contributing parameters in the proposed algorithm.

b) Upper support threshold: this measure is one of the quantities that is considered as the basis of execution time evaluation. For the temporal item x, this parameter is defined in [30, 50] and different values of upper support threshold are obtained as follows:

$$SUPP((X \cup Y)^{MCP(X)}) = \frac{\left|\{T \in db^{MCP(X)} \mid X \subseteq T\}\right|}{\left|db^{MCP(X)}\right|} \tag{14}$$

c) Number of transactions in the original database: According to the significant role of transactions number in the execution time, it is important to evaluate the execution period for different amounts of transactions [30-32].

d) Number of inserted transactions in the incremental database: This is also an important contributor to the execution time and should be considered to assess the algorithm performance at different sizes [29, 50].

5.2 Test Method

The test goal of the proposed method, PLI-X, is evaluating the execution speed of the method based on the mentioned measures in the previous subsection. The test approach used in the present research is similar to the employed test methods in the literature [30, 37, 50]. In addition to the execution speed evaluation, the performed tests help to adjust the prerequisite and initial parameters of the proposed algorithm and its comparison with the previous ones. The size of the original database is considered the same in the all performed tests. Also, in each performed test, there is a difference between the sizes of added database in comparison to the size of the added database in the other test. All the tests of the present study are performed by a computer with a dual-core processor with 2.53 GHz of clock frequency and 4 GB of RAM.

5.3 Implemented Data sets

Two data sets including artificial data and real data were used to evaluate the efficiency of the proposed algorithm in this research. In terms of the real data, the tests of the present study are performed on the real BMS-POS dataset which is available on the KDDCUP website. It is a well-known dataset for association rules discovery in the field of CRM and is used in the tests of some previous researches [37, 38]. The BMS-POS dataset pertains to sale information of a large electrical equipment market during a few years. This market supplies different products and ; therefore, each group of products is considered an item. Each transaction in the database contains the purchases of a customer at a specific time. The purpose is to find the rules between different products dataset that are purchased by the customer. The characteristics of the BMS-POS data set are presented in Table 4. In terms of the artificial data, we chose the classic data set, T10I4N4KD100K, which can conduct to validate the efficiency of the proposed algorithm, PLI-X algorithm [29, 30]. Synthetic data was generated by the public IBM data generator. The temporal database was generated by the model used in [13].The detail of the dataset is shown in Table 5.

Table 4. Characteristics of BMS-POS data set

Characteristics	Value
Number of transaction	515597
Number of the entire items	1657
Number of items in the largest transaction	164
Average length of transactions	6.5

Table 5. Characteristics of artificial data set

Characteristics	Value
Number of transaction	100000
The average length of maximal potentially frequent itemsets	4
The total number of items	4000
The average length of items per transaction	10

5.4 Experimental Results

In this section, the proposed method, PLI-X, is compared with some of the other methods in the field of association rules mining for the discovery of temporal association rules in the dynamic transactional databases. The efficiency comparison is conducted with the UTARM [29] and TPPF [30] methods as these methods are recent algorithms in the field of temporal association rules mining, which are performed on the transactional databases. Also, there is an act likeness between the proposed method and the act of those. Hence, given the purpose of the research, the mentioned resources can provide the possibility to compare PLI-X method and other the recently published resources from different aspects including execution time, minimum support, original database size, and incremental database size. The same adjustments and attributes are considered for the above-mentioned methods and our proposed method in order to obtain meaningful results. This section is orgenized into three subsections, experiment on real data set, experiment on artificial data set, and analysis of computational complexity.

5.4.1 Experiment on Real Data set

In this subsection, the real data set for efficiency evaluation of the proposed algorithm in terms of execution time, minimum support, original database size, and incremental database size is used.

A) Evaluating Execution Time Under Various Upper Support Thresholds

In experiments, four tests have been performed for evaluating the execution time of the proposed method with that of the two recent algorithms, i.e. UTARM and TPPF methods under different values of upper support thresholds. There are differences between the four performed tests including the size of the original database, the size of the increased database, and the number of partitions in the tested database. Each test is executed ten times under different values of the upper support threshold in the range of 10-100 and the 10 spacing. Also, the average value of safety factor is 50% which is considered as the value of safety factor in all the tests. Evaluation results of execution time are drawn for three methods in Fig 6.

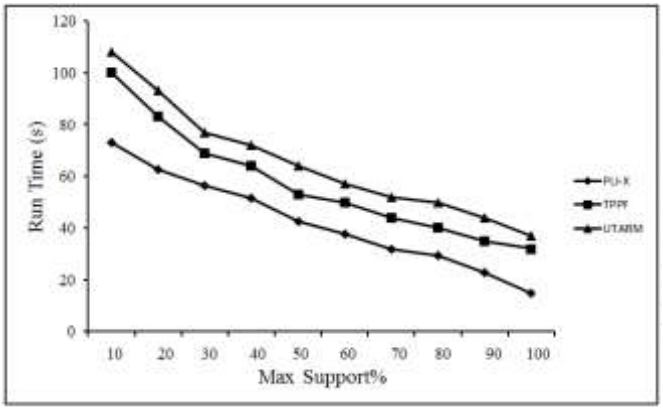

Fig 6. Evaluating Execution Time of Methods Under Various Upper Support Thresholds in the original database

As it is seen from Fig 6, the proposed method has a less execution time compared to UTARM and TPPF algorithms. It can be said that the main reason for this reduction is the number of scan processes in each of the implemented algorithms. In the other word, the proposed algorithm scans the database only when it is required, while the scan process of database and the number of those is different in the UTARM and TPPF algorithms for discovering the association rules. Indeed, lack of using the incremental problem and pre-large itemset in discovery of temporal association rules by UTARM and TPPF algorithms [29, 30] can be discussed as one major reason, which makes a difference between the scan counts of the database in the tested algorithms. The proposed method uses pre-large itemsets concept that is not initially large but is expected to convert into a large set in future. The pre-large itemsets work like a buffer with the purpose of reduceing the number of direct item transfers from large to small itemsets and vice versa. Also, the evaluation results of Fig 6 shows that the execution time reduces by increasing the upper support threshold in all methods. From Fig 6 it is inferred that if a large value and near to maximum value is selected for the upper support threshold, items count reduces for sending to the next step. Thus, the execution time decreases with the reduction in items count. Given that items count has a direct impact on the algorithms' efficiency of temporal association rule mining, the efficiency also increases.

B) Evaluating Scalability

In this section, scalability of PLI-X, UTARM and TPPF methods are investigated from two aspects: number of transactions in the original database and number of increased transaction in the incremental database. In the performed investigation, the execution time is obtained according to the two aspects: number of transactions in the original database and number of the increased transactions in the incremental database for PLI-X, UTARM, and TPPF methods. Then evaluation results are compared.

i) Evaluating Scalability of Methods With the Number of Transactions in the Original Database

In this part, an experiment is designed to investigate the scalability of the proposed method against different sizes of the original database in comparison to the UTARM and TPPF methods. For this purpose, PLI-X, UTARM, and TPPF methods are performed on the original database in order to discover temporal association rules, and the execution time is computed for different sizes of original database by changing the transactions' number. Thus, the role of transactions' number in the execution time of methods are computed and the evaluation results of the scalability in the mentioned situations are drawn in Fig 7.

In the presented investigation, three tests have been performed in order to analyze the effect of original database size in the execution time of methods in which values of upper support threshold, lower support threshold, the size of added database, and the safety factor are considered the same. As it is inferred from Fig 7, the execution time of methods is increased linearly with the addition of transactions number in the original database. The addition of the execution time of methods after each of database resizing is clear and reasonable because the addition of transactions number in the original database leads to the addition of processes volume and increasing the execution time of methods.

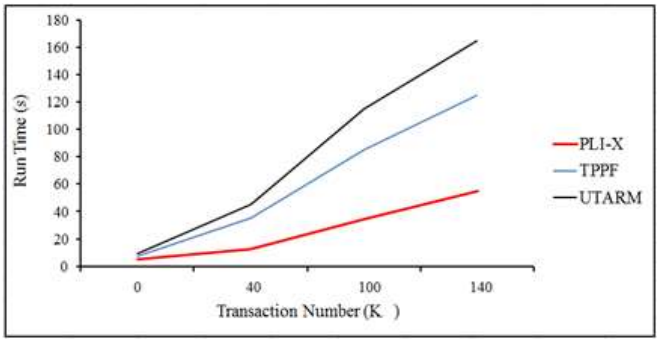

Fig 7. Evaluating Scalability of Methods With Changing the Number of Transactions in the Original Database

Also, it is observed in Fig 7 that the lowest execution time is presented by the proposed method, PLI-X in the different sizes of the original database that is obvious due to reducing the scan number of original database by PLI-X. Then, the lowest execution time is obtained by TPPF method in the performed tests in comparison to the UTARM method. The TPPF method adopted a predicting strategy which can reduce the number of data scan by the upper-bound support in comparison to UTARM. Thus, the volume of the scan process is not the same in the TPPF and UTARM methods. In fact, the computational cost is reduced for scanning a temporal database. Because, TPPF method is applied the prediction strategy and removed unpromising item-sets in the scan process of database.

ii) Evaluating Scalability of Methods With Number of Increased Transaction in the Incremental Database

In this part, other experiment is designed to check the scalability of the proposed algorithm against different sizes of incremental database. For this purpose, the role of increased transactions number is evaluated in the time execution variations of PLI-X, TPPF, and UTARM methods. The evaluation results are depicted in Fig 8.

In this case, three tests have been performed in order to analyze the effect of increased transactions number in the execution time of methods with the values of upper support threshold, lower supportthreshold, the size of original database, and safety factor considered the same. As it is observed in Fig 8, the execution time of PLI-X, TPPF, and UTARM methods increased linearly along with the addition of new transactions in the incremental database.

Besides, based on the execution time obtained by the proposed method, PLI-X is the lowest run time over TPPF and UTARM methods in the performed tests, the results of which can demonstrate again the claim of the superior performance of the proposed method in comparison to the

other two investigated methods. Actually, evaluation results obviously indicate that the execution time of the PLI-X for the obtained temporal association rules discovery is better than TPPF and UTARM methods.

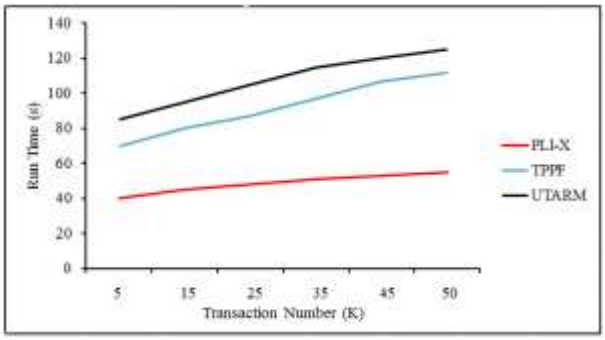

Fig 8. Evaluating Scalability of Methods With Changing the Number of Increased Transactions in the Incremental Database

It is obvious with regard to the number reduction of scanning processes and different classification of candidate items from the added database by the proposed method. In other words, PLI-X method does not scan the original database for each iteration of algorithm. In fact, a safe threshold is considered by PLI-X method for new increased transactions, which leads to the scan reduction of updated database. On this basis, the algorithm does not scan the updated database after inserting each new transaction to original database. Therefore, the proposed method prevents a complete scan of the database in each stage by doing the scan only when it is necessary due to the considered safe threshold. whereas, the common approach in many previous published resources like TPPF and UTARM methods is to scan throughout the transactional database in the situation that new transactions are added to the original database. For example, as it is seen in Figure 8, the maximum execution time is obtained by UTARM method. In this method, the original database is partitioned, then, each partition is defined as a utility table for items with different values. In situation that new transactiones are added to the original database, cost of execution time increases due to the addition of scan processes volume, which is reasonable. Actually, computations volume is high in UTARM method because of the additionality processes which performed in order to scan the utility tables.

Also, Fig 8 are shown that the size of the incremental database is always smaller in comparison to the original database size. In the other hand, the inserted transactions' number or the variation of transactions number has less effect on the execution time of methods; the slope of the resulted curves is less when the number of the inserted transactions changes in incremental database compared to the variation of transactions number in the original database. There are key and important points in the above evaluation results which is scalability of the proposed method. From the analysis of Figs 7 and 8, it can be concluded that the proposed method is scalable in comparison to other used methods in the performed tests. Hence, given to the scalability of the proposed method, the execution time of PLI-X is the lowest execution time for temporal association rules discovery when there are resizing in the original and incremental databases. Since in the real-world problems, there are dynamic databases, the scalability of the proposed method can play the key role in temporal association rules in a CRM system for presenting reasonable results in the lowest time.

C) Evaluating SpeedUp

In this section, the speedup ratio is investigated in the methods of PLI-X, TPPF, and UTARM for various values of upper support thresholds; the evaluation results are given in Fig 9.

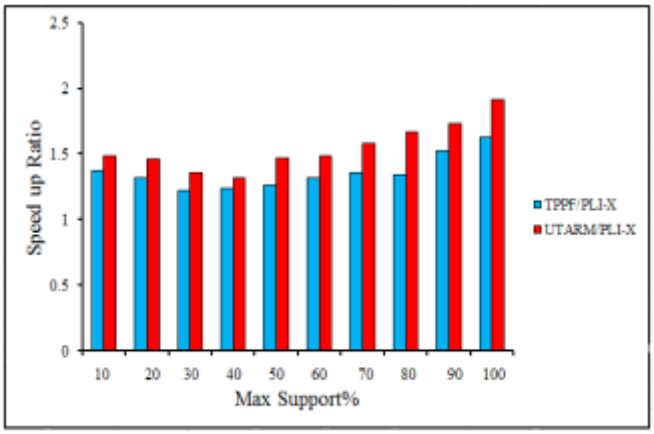

Fig 9. The comparing PLI-X, TPPF and UTARM Methods in Terms of Speedup ratio

The results of Fig 9 explain the speedup ratio by the proposed method with regards to TPPF and UTARM methods. As it is drawn in Fig 9, the speedup ratio of the PLI-X method has reduced with the gradual increase of the upper support threshold from 10 to 50. Thus, it can be concluded that the speedup ratio has direct relevance to the execution time of the proposed method. Also, the results of Fig 9 observed a reversed trend in the interval 60-100 of the upper support threshold; because there is more reduction of execution time by PLI-X method in comparison to other tested methods. Actually, the proposed method, PLI-X, starts with 1-item candidates for generating the candidate sets, while the other methods start with the generation of 2-item candidates. For example, all possible 2-item candidates have generated from partition *P1* in the UTARM method. Then, for each candidate, two itemsets, the frequency, and the utility values for the partition P_1 are calculated by scanning the database and the utility table through which the mentioned processes have performed for each partition of the database.

Also, the liltering operation is started in an earlier time according to the largeness of the upper support threshold by the proposed method. Many of candidate items are eliminated and, consequently, the method performance is enhanced. From results achieved in the Fig 9 by TPPF method it can be concluded that there is more reduction of execution time by TPPF method in comparison to UTARM method. It is obvious since there is differenc between computational complexity of TPPF and UTARM methods due to using an effective strategy for predicting upper-bound of support values for itemsets by TPPF method. The mentioned strategy leads to the reduction of run time in comparison to UTARM method. Therefore, it can be said that the proposed method

has higher speed up ratio than other compared methods. In the other word, the proposed method obtains to a speed up ratio up to 1.3 faster than the TPPF method and 1.6 faster than the UTARM method, respectively.

5.4.2 Experiment on Artificial Data set

In this subsection, we use an artificial data set for the effectiveness and usefulness evaluation of the proposed algorithm in terms of the execution time, minimum support, original database size, and incremental database size.

A) Evaluating Execution Time Under Various Upper Support Thresholds

For experimentation, PLI-X, UTARM, and TPPF methods are applied on the specified artificial dataset, T10I4N4KD100K to compute the execution time of methods under various upper support thresholds. Then, three mentioned methods are compared in terms of execution time by changing the various upper support thresholds. In the performed testes, the average value of safe threshold is considered 50%. Also, each test is executed ten times under different values of the upper support threshold in the range of 10-100 and the 10 spacing. The relationship between the execution time of the methods and upper support at different densities are shown in Fig 10.

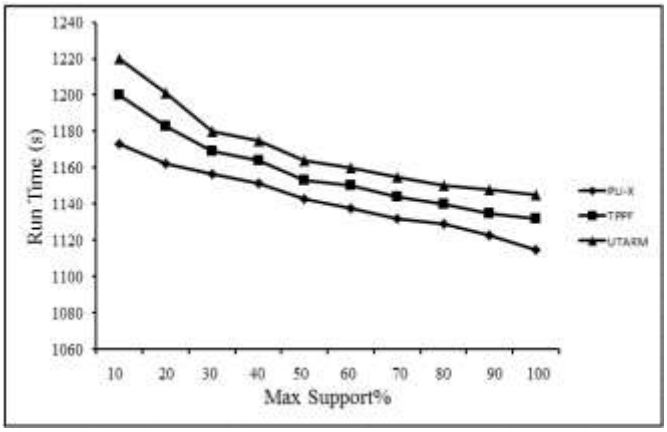

Fig 10. Evaluating Execution Time of Methods Under Various Upper Support Threshold in the original database (artificial data set)

As is inferred from comparing the evaluation results in Fig 10, the lower execution time is provided by all the tested methods in the higher the maximum support. Also, the proposed method has a less execution time in comparison to UTARM and TPPF methods, which is reasonable. In fact, the main reason for this reduction is the number of scan processes in each of the implemented algorithms. It can be concluded that act of the proposed method on the artificial data is similar to act of that on the real data. In the other word, execution time is saved when PLI-X performed on the both data sets in comparison to other tested methods.

B) Evaluating Scalability

In this section, two different experiments are designed to evaluate the scalability of PLI-X, UTARM, and TPPF methods when they performed on the artificial data against the changing number of transactions in the original database and the changing number of increased transaction in the the incremental database.

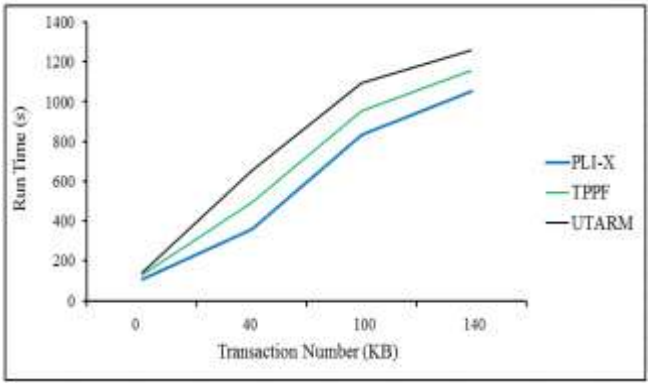

Fig 11. Evaluating Scalability of Methods With Changing the Number of Transactions in the Original Database (artificial data set)

i) Evaluating Scalability of Methods With Changing the Number of Transactions in the Original Database

In this part, for the purpose of scalability evaluation, PLI-X, UTARM, and TPPF methods are executed on the artificial data; temporal association rules are discovered under various sizes of the original database by changing transactions number, the execution time is computed for different sizes in original database. Then, the comparative analysis of execution time of methods is depicted in Fig 11.

It notes that values of upper support threshold, lower support threshold, the size of the added database, and the safe threshold are considered same in the performed experiment. Also, three tests are performed to analyze methods of scalability under the original database resizing.

Two main obtained results are concluded from Fig 11. The first, the execution time of methods is increased after each of original database resizing, which is logical as the growth of transactions number in the original database leads to the increase of the process volume. Second, the execution time of the proposed method decreases when the amount of the original database grows larger in comparison to other methods in this experiment. In fact, the efficiency of PLI-X method is comparatively better than UTARM and TPPF in terms of scalability. As the number of transactions in the original database keeps increasing, the execution time of PLI-X method showed significant improvement in comparison to other tested methods.

ii) Evaluating Scalability of Methods With Changing the Number of Increased Transactions in the Incremental Database

This part has carried out PLI-X, UTARM, and TPPF methods on the artificial data to evaluate their scalability with regard to the execution time of methods under various sizes of incremental database. Then, effectively analyzing the scalability of methods. Values of upper support threshold, lower support threshold, the size of original database, and safe threshold are considered same in the designed experiment. Also, three tests are performed to analyze

methods of scalability under incremental database resizing. Evaluation results are drown in Fig 12.

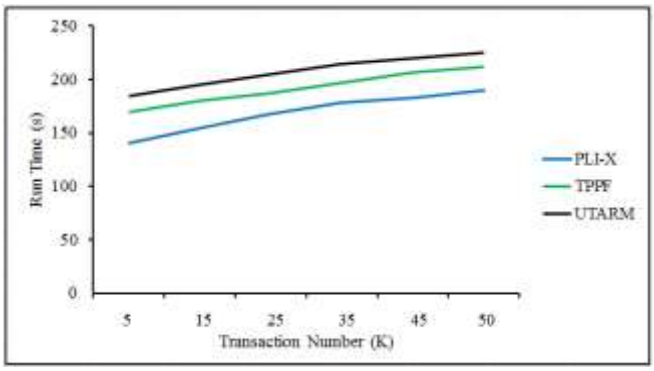

Fig 12. Evaluating Scalability of Methods With Changing the Number of Increased Transactions in the Incremental Database (artificial data set)

In this experiment, as it is seen in Fig 12, the execution time of PLI-X, TPPF, and UTARM methods are also increased linearly along with adding new transactions in the incremental database. It can be concluded from Fig 12 that the scalability of the PLI-X method is better than TPPF and UTARM methods for the test case of varying number of new transactions in the incremental database. In the other words, the computational time required for PLI-X method is comparatively lesser than for other tested methods when the methods applied on the artificial data. It is logical due to the number reduction of scanning processes and different classification of candidate items from the added database by the proposed method.

Actually, these results demonstrate that the proposed method can incrementally generate frequent itemsets efficiently in the situations where there are not real data.

Analyzing the results provided in Figs 11 and 12, it can be seen how the proposed algorithm performs well in almost all of the experiments on the artificial data.

C) Evaluating SpeedUp

In this experiment, PLI-X, TPPF, and UTARM methods are compared in terms of speedup ratio. The comparisons are dipected in Fig 13.

As it is seen in Fig 13, the execution time is reduced by PLI-X method in comparison to TPPF, and UTARM methods, which is reasonable as there is a difference in the performed strategy by PLI-X method for the temporal association rules discovery. Hence, a key achievement of this experiment can be a better speedup ratio of PLI-X method than that of TPPF and UTARM methods. Similarly, the results obtained on the synthetic dataset show that there is a reversed trend in the interval 60-100 of the upper support threshold. Also, speedup ratio of the proposed method has reduced with the gradual increase of the upper support threshold from 10 to 50 . In fact, it can be inferred that the speedup ratio has a direct relevance to the execution time of the proposed method.

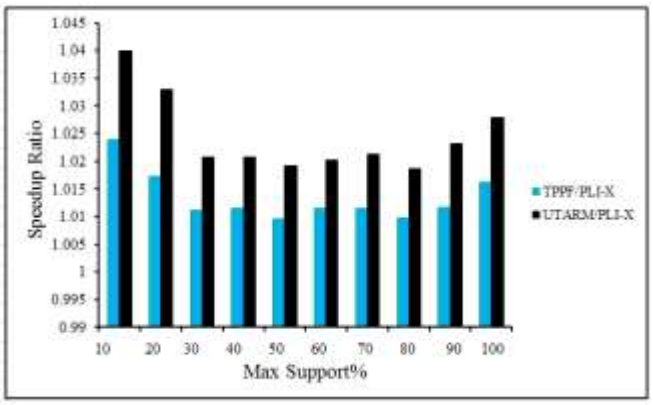

Fig 13. The Comparing of PLI-X, TPPF and UTARM Methods in Terms of Speedup ratio (artificial data set)

In the end, as a result, it can be said that the proposed method of PLI-X provides acceptable results to discover temporal association rules when PLI-X method applied to the real and artificial datasets.

5.4.3 Analysis of Computational Complexity

There is a common strategy in the temporal rules discovery algorithms that decompose the problem into two main subtasks: frequent itemset generation and rules generation. The computational requirements for frequent itemset generation are generally more expensive than those of rules generation [13]. Also, the space complexity of algorithms (memory consuming) is generally more inexpensive than the time complexity of algorithms due to the development and growth of data storage devices. In fact, the space complexity is the number of tape cells used by the computation the analysis of which is out of scope of this study. The time complexity of execution is the number of steps until the machine halts. Typically, it is tried to bound the time complexity as a function of the size n of the input, defined as the number of cells occupied by the input, excluding the infinite number of blanks that surround it [49]. The computational complexity of the association rules mining algorithms can be affected by some of the following factors:

- *Support Threshold*: Lowering the support threshold often results in more itemsets being declared as frequent. This has an adverse effect on the computational complexity of the algorithms because more candidate itemsets must be generated and counted.
- *Number of items* (*Dimensionality*): As the number of items increases, more space will be needed to store the support counts of items. If the number of frequent items also grows with the dimensionality of the data, the computation and I/O costs will increase because of the larger number of candidate itemsets generated by the algorithm.
- *Number of Transactions*: Run time algorithm increases with a larger number of transactions because of repeated passes over the data set [50]. The effect of the mentioned factors is empirically investigated on the time complexity (consumed

time) of the proposed algorithm by experiments performed in the two previous subsections.

As it is reported in the two previous subsections, empirical results demonstrated that the proposed algorithm reduces run time in comparison to the other temporal association rule mining algorithms for both data sets. In this subsection, a theoretical analysis is presented for the time complexity of the proposed algorithm. The *big-O* notation represents a theoretical analysis upon which we can compare two or more algorithms.

FournierViger et al. [40] mentioned that there are two general ways to reduce the computational complexity of frequent itemset generation such as reducing the number of candidate itemsets and reducing the number of comparisons. First, some of the candidate itemsets are eliminated without counting their support values. Second, instead of matching each candidate itemset against every transaction, the number of comparisons can be reduced by using more advanced data structures, either to store the candidate itemsets or to compress the data set. In this paper, rapid generation of temporal association rules is one of the main features of the proposed algorithm. In fact, using the proposed algorithm, PLI-X algorithm causes partitioning of candidate itemsets on the basis of previous partitions and scanning of the database only when it is necessary; it prevents a complete scan of database in each stage by doing the scan only when it is necessary. Actually, the main goal of the proposed method is keeping repeated temporal patterns after updating temporal transactions of the database. Thus, the used methodology improves efficiency, reduces the number of database scans, and also saves time complexity. The computational complexity analysis of the proposed method is described in detail as follows:

a) Calculate the value of $f = \dfrac{(S_u - S_l)\sum_{m=s}^{e}|p_h|}{1-S_u}$: In In the first step of the proposed algorithm, *f* parameter is obtained as a safe threshold. Using this safe threshold can cause count reduction of database scans. Thus, the updated database will be scanned due to the *f* value and after inserting multiple new transactions. The computing *f* parameter is performed in a constant time. Hence, the total time complexity for calculating the *f* parameter can be given as, *O(c)*.

b) *Find all condidate k-itemsets ck and their count*: *k* variable shows item counts in the items' sequence,where there are two states for *k* variable. If *k*=1, then, condidate 1-itemsets was found. In this situation, for each transaction, the support count for every item present in the transaction needs to be updated. Assuming that there are *n* number of transactions and average w items per transaction in the database, condidate 1-itemsets can be found with their count require *O(nw)* time. Besides, we find condidate *k*-itemsets using self-joining due to the condidate (*k-1*)-itemsets which is stored in the previously execution. Each merging operation requires at most *k-2* equality comparisons. In the best-case scenario, every merging step produces a viable candidate k-itemset. In the worst-case scenario, the algorithm must merge every pair of condidate (*k - 1*)-itemsets found in the previous iteration. Because the maximum depth of the tree is *k*, the cost for populating the hash tree with candidate itemsets is $O(\sum_{k=2}^{w} k\,|c_k|)$.

c) Partition all condidate k-itemsets and calculate a new count for their:
This step contain two stages, whichi is performed for each condidate *k*-itemsets *ck* found in the previously step. In the first stage, condidate *k*-itemsets are divided into three sets. To this end, a comparison was performed between each of *k*-items sequence of condidate itemsets and pre-large, large itemsets in the original database. Then, condidate itemsets are placed in the small, large, and pre-large stes. In the next stage, the conditions below were investigated for each *k*-items sequence (*I*) being placed in the small itemsets.

$$S^{U}(I) \geq S_u \cdot \sum_{m=s}^{e}|p_m| \qquad (15)$$

If condition=True then *I* moves in to large itemsets, else *I* move in to pre-large item-sets. After partitioning k-itemsets condidate, the count of there is updated. Since we carry out a comparison between each of *k*-items sequence of condidate itemsets and pre-large, large itemsets in the original database, thus, it can be said that the time required is based on the number of condidate *k*-itemsets, which is *O(m).*

d) Rescan original DB: The common opinion in many of the previous algorithms in terms of temporal association rule discovery is to scan throughout the transactional database after increasing each of new transactions in each execution algorithm, while the proposed algorithm PLI-X prevents complete scan of the database in each stage by doing the scan only when it is necessary according to the safe threshold. There are two conditions that if one of each comes true, an algorithm does not scan the whole original database after increasing new transactions. Two conditions are defined as follows:

(1) Is R set= $\varnothing$?
R set contains items that are not placed in the pre-large and large itemsets in the previous run but may increase the occurrence count of those after increasing new transactions. Thus, these items place in a large or pre-large itemsets after re-counting.
(2) $c + t \leq f$
Where *f* parameter is a safe threshold, *c* is transaction count which is newly inserted in the current run, and *t* is transaction count in the last scan of the original database. In fact, the proposed algorithm do scan the whole original database only when it is necessary due to the safe thresholde after inserting multiple new transaction, for example the increase of *x* new transactions.

In the worst-case scenario, no of the mentioned conditions are correct. In this condition, the algorithm does scan the whole original database after inserting each of the new transactions. If we assume that, there are *m* number of transactions and average *n* items per transaction in the database, *i* number of increased transactions, the time required to database scan is bounded by complexity *O*(*mni*).

In the best-case scenario, the mentioned conditions are true. In such condition, algorithm does scan the original database due to the f parameter for each x new transactions which increased into the database. Hence, the time required is $O(mn\frac{i}{x}),\frac{i}{x} \prec i$.

6. Conclusion and Future Research

There are three important challenges in the field of Customer Relationship Management systems (CRM) including 1) an extremely high rate of customer data generation; 2) the requirement of extraction of useful/frequent rules and patterns for enhancement of the enterprise profitability; 3) updating commercial applications of a dynamic information system according to the customer needs. Also, improving association rules mining especially temporal association rule mining is an important yet often difficult task facing systems of CRM in many areas. So far, various methods have been proposed by researchers in the field of the temporal association rules mining in the CRM systems that are executable on the partitions of the database an identical timestamps. The available methods need numerous scanning of the database for the discovery of patterns and, consequently, they are not successful in the satisfaction of this requirement. Most of the previous developed methods in this field are applicable in the databases with numerical values and attributes. There are a few methods that could encounter with the challenges in temporal mining of association rules. In the present research, a novel incremental mining algorithm, PLI-X is proposed for the discovery of temporal association rules that is more efficient in comparison to the previous methods in terms of the execution time. In order to extract temporal association rules in dynamic systems, the present study implements incremental mining of the database. This is carried out using more than one support threshold for item grouping and different methods for partitioning of itemsets. Incremental mining of temporal databases has capability of generation of the valid rules in incremental databases. On the other side, implementation of more than one support threshold in the partitioning process on itemsets reduces the execution time of algorithm because many of items are eliminated in the initial stages of algorithm execution and are not examined anymore. The proposed PLI-X method examines the new items after the generation of candidate itemsets to identify whether it is related to large, pre-large or small itemsets and not requiring the scanning of the database for each algorithm execution. In order to obtain an optimal relation between the lower and upper support threshold parameters, the curve fitting technique is applied to the results of PLI-X algorithm and the unknown coefficients are determined after the minimization of the error between actual tests and models. The implementation of the proposed algorithm with the optimal support thresholds has generated all maximized frequent items in a more efficient procedure.

It is obvious that the consequence of the increase in run speed can be an accurate reduction in generating an outcome as there is a drawback in the proposed method. Also, in the previous methods and the proposed method, the size of increased databases is considered the same the lack of which can also be discussed as a weakness of the proposed method. Hence, some of the future research are listed as it follows:

(i) Improving the accuracy of the proposed method by keeping the current run time of the method.
(ii) Extending the proposed method for performing in situations that the size of an increase database is variable.
(iii) Presenting the practical software with the graphical interface of user-friendly by using the proposed method.

References

[1] Fares, A., Gama, J. and Campos, P.: Process mining for analyzing customer relationship management systems: A case study. In Learning from Data Streams in Evolving Environments, Part of the Studies in Big Data book series, Springer, Cham, pp. 209-221 (2019)

[2] Srivastava, S.K., Chandra, B. and Srivastava, P.: The impact of knowledge management and data mining on CRM in the service industry. In Nanoelectronics, Circuits and Communication Systems, Springer, Singapore, pp. 37-52 (2019)

[3] Rahman, N.: A taxonomy of data mining problems. In Cognitive Analytics: Concepts, Methodologies, Tools, and Applications, IGI Global, pp. 512-528 (2020)

[4] Tembhurne, D.S., Adhikari, J. and Babu, R.: A Review study on Application of Data Mining Techniques in CRM of Pharmaceutical Industry. *International Journal of Scientific Research in Science and Technology*, Vol. 6, No. 2, pp. 1-7 (2019)

[5] Keyvanpour, M. R., Etaati, A.: Analytical Classification and Evaluation of Various Approaches in Temporal Data Mining. Advanced Information Technology in Education. In: Springer, pp. 303-311 (2012)

[6] Grossmann, W., Rinderle-Ma, S.: Data Mining for Temporal Data. In: Fundamentals of Business Intelligence. Data-Centric Systems and Applications. Springer, Berlin, Heidelberg, pp. 207-244 (2015)

[7] Radhakrishna, V., Kumar, P. V., Janaki, V.: A survey on temporal databases and data mining. *In Proceeding of the The International, Conference on Engineering and MIS. ACM*, New York, NY, USA, pp. 52-58 (2015)

[8] Mehrmolaei, S. and Keyvanpour, M.R.: An enhanced hybrid model for event prediction in healthcare time series. International *Journal of Knowledge-based and Intelligent Engineering Systems*, Vol. 23, No. 3, pp.131-147 (2019)

[9] Emtiyaz, S., Keyvanpour, M. R.: Customers behavior modeling by semi-supervised learning in customer relationship management. arXiv preprint arXiv:1201.1670 (2012)

[10] Tripathi, T. and Yadav, D.: Performance Evaluation of Methods for Mining Frequent Itemsets on Temporal Data. In International Conference on Computer Networks and Inventive Communication Technologies, . Springer, Cham, pp. 910-917 (2019)

[11] Chamazi, M.A. and Motameni, H.: Finding suitable membership functions for fuzzy temporal mining problems using fuzzy temporal bees method. *Soft Computing*, Vol. 23, No. 10, pp.3501-3518 (2019)

[12] Sornalakshmi, M., Balamurali, S., Venkatesulu, M., Navaneetha Krishnan, M., Ramasamy, L.K., Kadry, S.,

Manogaran, G., Hsu, C.H. and Muthu, B.A.: Hybrid method for mining rules based on enhanced Apriori algorithm with sequential minimal optimization in healthcare industry. Neural Computing and Applications, pp. 1-14 (2020)

[13] Ghorbani, M., Abessi, M.: A New Methodology for Mining Frequent Itemsets on Temporal Data. *IEEE Transactions on Engineering Management*, Vol. 64, No. 4, pp. 566-573 (2017)

[14] Kiran, R. U., Reddy, P.P.C., Zettsu, K., Toyoda, M., Kitsuregawa, M. and Reddy, P.K.: Efficient Discovery of Weighted Frequent Neighborhood Itemsets in Very Large Spatiotemporal Databases. *IEEE Access*, Vol. 8, pp. 27584-27596 (2020)

[15] Rachburee, N., et al.: Failure Part Mining Using an Association Rules Mining by FP-Growth and Apriori. Algorithms: Case of ATM Maintenance in Thailand. IT Convergence and Security. In: Springer, pp. 19-26 (2018)

[16] Wang, C. and Zheng, X.: Application of improved time series Apriori algorithm by frequent itemsets in association rule data mining based on temporal constraint. *Evolutionary Intelligence*, Vol. 13, No. 1, pp. 39-49 (2020)

[17] Hareendran, S. A., Chandra, S. V.: Association Rule Mining in Healthcare Analytics. In International Conference on Data Mining and Big Data, In: Springer, pp. 31-39 (2017)

[18] Wang, L., et al.: Mining temporal association rules with frequent itemsets tree. *Applied Soft Computing*, Vol. 62, pp. 817-829 (2018)

[19] Ait-Mlouk, A., Gharnati, F. and Agouti, T.: An improved approach for association rule mining using a multi-criteria decision support system: a case study in road safety. *European Transport Research Review*, Vol. 9, No. 40, pp. 1–13, (2017)

[20] Logeswaran, K., Suresh, P., Savitha, S. and KR, P.K.: Optimization of Evolutionary Algorithm Using Machine Learning Techniques for Pattern Mining in Transactional Database. In Handbook of Research on Applications and Implementations of Machine Learning Techniques, IGI Global, pp. 173-200 (2020)

[21] Telikani, A., Gandomi, A.H. and Shahbahrami, A.: A survey of evolutionary computation for association rule mining. *Information Sciences*, Vol. 524, pp. 318-352 (2020)

[22] Song, Y-G., et al.: Parallel Incremental Frequent Item set Mining for Large Data. *Journal of Computer Science and Technology*, Vol. 32, No. 2, pp. 368-385 (2017)

[23] Li, H., et al.: A Heuristic Rule Based Approximate Frequent Item set Mining Algorithm. *Procedia Computer Science*, Vol. 91, pp. 324-333 (2016)

[24]Agarwal, R.: Decision-Making with Temporal Association Rule Mining and Clustering in Supply Chains. In Optimization and Inventory Management, Springer, Singapore, pp. 459-470 (2020)

[25] Mai, T., Vo, B., Nguyen, L. T.: A lattice-based approach for mining high utility association rules. *Information Sciences*, Vol. 399, pp. 81-97 (2017)

[26] Lee, C-H., et al.: Progressive partition miner: an efficient algorithm for mining general temporal association rules. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 15, No. 4,pp. 1004-1017 (2003)

[27] Xiong, L., Liu, X., Guo, D. and Hu, Z.: Access patterns mining from massive spatio-temporal data in a smart city. *Cluster Computing*, Vol. 22, No. 3, pp. 6031-6041 (2019)

[28] Kadir, M., Sobhan, S. and Islam, M.Z.: Temporal relation extraction using Apriori algorithm. In proceeding of the 5th International Conference on Informatics, Electronics and Vision, IEEE Xplore, DOI: 10.1109/ICIEV.2016.7760133, pp. 915-920, (2016)

[29] Maragatham, G., Lakshmi, M.: UTARM: an efficient algorithm for mining of utility-oriented temporal association rules. *International Journal of Knowledge Engineering and Data Mining*, Vol. 3, No. 2, pp. 208-237 (2015)

[30] Hong, T. P., et al. Discovery of temporal association rules with hierarchical granular framework. *Applied Computing and Informatics*, Vol. 12, No. 2, pp. 134-141 (2016)

[31] Tan, T. F., et al.: Temporal Association Rule Mining. In: He X. et al. (eds) Intelligence Science and Big Data Engineering. Big Data and Machine Learning Techniques. IScIDE Lecture Notes in Computer Science, vol 9243. Springer, Cham, DOI: 10.1007/978-3-319-23862-3_24 , pp. 247-257 (2015)

[32] Ilham, A., et al.: Market Basket Analysis Using Apriori and FP-Growth for Analysis Consumer Expenditure Patterns at Berkah Mart in Pekanbaru Riau. In Journal of Physics: Conference Series, IOP Publishing, Vol. 1114, No. 1, DOI:10.1088/1742-6596/1114/1/012131, pp. 1-10, (2018)

[33] Hasan, M.M. and Mishu, S.Z.: An Adaptive Method for Mining Frequent Itemsets Based on Apriori And FP Growth Algorithm. *In IEEE xplore International Conference on Computer, Communication, Chemical*, Material and Electronic Engineering (IC4ME2), pp. 1-4 (2018)

[34] Hong, T.-P. , Yu Lin, Ch., Huang , W. M., Li , S. , M., Wang, S. L. , Lin, J. C. W.: Mining Temporal Fuzzy Utility Itemsets by Tree Structure, In proceeding of IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, USA, DOI: 10.1109/BigData47090.2019.9006317, pp. 1-5 (2019)

[35] Sathyavani, D. and Sharmila, D.: An improved memory adaptive up-growth to mine high utility itemsets from large transaction databases. *Journal of Ambient Intelligence and Humanized Computing*, pp. 1-10 (2020)

[36] Wang, W., et al.:TAR: Temporal association rules on evolving numerical attributes. *In Proceedings of IEEE 17th International Conference on Data Engineering*, pp. 283-292 (2001)

[37] Gharib, T. F., et al.: An efficient algorithm for incremental mining of temporal association rules. *Data and Knowledge Engineering*, Vol. 69, No. 8, pp. 800-815 (2010)

[38] Huang, J-W., et al.: Twain: Two-end association miner with precise frequent exhibition periods. ACM Transactions on Knowledge Discovery from Data (TKDD), Vol. 1, No. 2, DOI: 10.1145/1267066.1267069 pp. 8-25 (2007)

[39] Kumar, B.P. and Paulraj, D.: Frequent mining analysis using pattern mining utility incremental algorithm based on relational query process. *Journal of Ambient Intelligence and Humanized Computing*, pp. 1-11 (2020)

[40] Hui, L., Chen, Y.C., Weng, J.T.Y. and Lee, S.Y.: Incremental mining of temporal patterns in interval-based database. *Knowledge and Information Systems*, Vol. 46, No.2, pp. 423-448 (2016)

[41] Sun, J., Xun, Y., Zhang, J. and Li, J.: Incremental Frequent Itemsets Mining With FCFP Tree. *IEEE Access*, Vol. 7, pp. 136511-136524 (2019)

[42] Matthews, S.G., Gongora, M.A., Hopgood, A.A.: Evolving temporal association rules with genetic algorithms, in: International Conference on Innovative Techniques and Applications of Artificial Intelligence, pp. 107-120 (2010)

[43] Maragatham, G., Lakshmi, M.: A weighted particle swarm optimization technique for optimizing association rules, in: International Conference on Computing and Communication Systems, pp. 655-664 (2011)

[44] Wen, F., Zhang, G., Sun, L., Wang, X., Xu, X.: A hybrid temporal association rules mining method for traffic congestion prediction, Computers & Industrial Engineering, Vol. 130, DOI:10.1016/j.cie.2019.03.020, pp. 779-787 (2019)

[45] Matthews, S.G. , Gongora, M.A., Hopgood, A.A.: Evolving temporal fuzzy association rules from quantitative data with a multi-objective evolutionary algorithm, in: International Conference on Hybrid Artificial Intelligence Systems, pp. 198-205 (2011)

[46] Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S., Coello, C.A.C.: A survey of multiobjective evolutionary algorithms for data mining: Part I, *IEEE Transactions on Evolutionary Computation*, Vol. 18, No. 1, pp. 4-19 (2014)

[47] Ahmed, C. F., et al.: Interactive mining of high utility patterns over data streams. Expert Systems with Applications, Vol. 39, No. 15, pp. 11979-11991 (2012)

[48] Bettini, C., Wang, X. S., and Jajodia, S.: Temporal Granularity. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. DOI: 10.1007/978-0-387-39940-9_397 (2009)

[49] Kusumakumari, V., et al.: Frequent pattern mining on stream data using Hadoop Can Tree-GTree. *Procedia Computer Science*, Vol. 115, pp. 266-273 (2017)

[50] Lin CW, et al.: Mining High Utility Itemsets Based on the Pre-large Concept. In: Chang RS., Jain L., Peng SL. (eds) Advances in Intelligent Systems and Applications, Smart Innovation, *Systems and Technologies*, Vol. 20. Springer, Berlin, Heidelberg, pp. 243-250 (2013)

[51] Junheng-Huang,W-W.: Efficient algorithm for mining temporal association rule. *The International Journal of Computer Science and Network Security (IJCSNS)*, Vol. 7, No. 4, pp. 268-271 (2007)

[52] Nguyen, L. T. T., Nguyen, N. T.: Incremental Mining Class Association Rules Using Diffsets. In: Le Thi H., Nguyen N., Do T. (eds) Advanced Computational Methods for Knowledge Engineering. *Advances in Intelligent Systems and Computing, Springer*, Cham, Vol. 358, pp. 197-208 (2015)