A mining method for tracking changes in temporal association rules from an encoded database

Chelliah Balasubramanian*, Karuppaswamy Duraiswamy**

K.S.Rangasamy College of Technology, Tiruchengode, Tamil Nadu-637215, India.

Abstract :Mining of association rules has become vital in organizations for decision making. The principle of data mining is better to use complicative primitive patterns and simple logical combination than simple primitive patterns and complex logical form. This paper overviews the concept of temporal database encoding, association rules mining. It proposes an innovative approach of data mining to reduce the size of the main database by an encoding method which in turn reduces the memory required. The use of the anti-Apriori algorithm reduces the number of scans over the database. The Apriori family of algorithms is applied on the encoded temporal database and their performances are compared. Also an important method on how to track the association rules that change with time is focused. This method involves initial decomposition of the problem. Later the changing association rules are tracked by dividing the time into smaller intervals and observing the changes in the itemsets obtained in each such interval. Thus the results obtained are lower complexities of computations involved, time and space with effective identification of changing association rules resulting in good decisions making. This helps in formalizing the database metrics in a better way.

Keywords: Temporal database encoding; Association rules mining; Data mining; Anti-Apriori algorithm; Apriori family of algorithms; Database metrics

I. INTRODUCTION

Knowledge discovery in databases (KDD) is often called as data mining. It discovers useful information from large collections of data [1]. The discovered knowledge can be rules describing properties of the data, frequently occurring patterns, clusterings of the objects in the database, etc. The amount of data stored in database is growing rapidly.

Intuitively, these large amounts of stored data contain valuable hidden knowledge [2]. Data mining especially association rule discovery tries to find interesting patterns from databases that represent the meaningful relationships between products and customers or other relationships in some other applications.

Because the amount of these transaction data can be very large, an efficient algorithm needs to be designed for discovering useful information.

An association rule describes the associations among items in which the presence of some items implies the presence of some other items in a transaction. In order to find association rules, there is a need to discover all large itemsets from a large database of transactions [3]. A large itemset is a set of items which appear often enough within the same transactions. The frequent itemset and association rules mining problem has received a great deal of attention and many algorithms have been proposed to solve this problem. Discovering association rules in these algorithms are usually done in two phases [4][5]. In the first phase, the frequent itemset are generated and in the second phase, the interesting rules are extracted from these frequent itemset. If the support and confidence of a rule is above the minimum threshold, the rule will be interest. The task of discovering all frequent itemset is quite challenging especially in large databases because the databases may be massive, containing millions of transactions. A famous algorithm, called Apriori, was proposed in [1], which generates (k+1)-candidates by joining frequent k-itemset. So all subsets of every itemset must be generated for finding superior frequent itemset, although many of them may be not useful for finding association rules because some of them have no interesting antecedent or consequent in the rules. This process takes a long time. And it also requires thousands of times of database scan. The complexity of the calculation increases exponentially. Additionally, the size of database is the main problem of this algorithm. Some modified algorithms of Apriori (AprioriTid and AprioriHybrid) are proposed to solve this problem but these algorithms also have the database size problem [2]. A method to encoding the database and an algorithm, which is called

anti-Apriori algorithm, is brought into focus. By using this algorithm, only the frequent itemset that are of interest and can be converted into association rules are generated, so it has a lower complexity of time and space. At the meantime, the times of the database scan are also reduced [2].

In real life, media information has time attributes either implicitly or explicitly. This kind of media data is called temporal data. Temporal data exist extensively in economical, financial, communication, and other areas such as weather forecast. Temporal databases store past, present and possibly future data. Temporal databases are append-only and current data values become historical data as new data values are added to the database [6]. Therefore they naturally are fertile data repositories for data mining and knowledge discovery. In this paper, the problem of discovery of association rules in temporal databases is examined. Instead of extracting rules throughout the entire time line, we will extract rules for consecutive time intervals with different time granularities. Thus, the user will be able to see the changes and fluctuations in the association rules as well as the time periods over which these rules are valid. When a database is expanded with an incremental increase in the number of transactions, we can either rerun an algorithm such as Apriori, or use the knowledge obtained from the "original" database to reduce the processing time for finding small itemsets that have become large or any large itemsets that have become small. UWEP (Update With Early Pruning) is used for this purpose and has been shown to be an efficient algorithm that addresses the incremental association rule problem [7]. UWEP scans the original database at most once and the increment in the database exactly once. It has been shown to generate and count the minimum number of candidates in order to determine the new set of association rules.

The rest of this paper is organized as follows. Section 2 introduces the encoding of temporal databases, along with the application of the Apriori family of algorithms and the anti-Apriori algorithm that are made compatible with the encoded temporal database. Section 3 describes the proposed method of discovery of association rules from an encoded temporal database. Finally, in Section 4 the conclusion and future enhancements are given which includes the best features of the described method and the ways in which it can be further improved.

II. ENCODING AND APRIORI FAMILY

In this section, a discussion on the encoding method and the application of the family of Apriori algorithms for association rule mining on a static database is presented.

2.1 The Encoding Method

The presentation of database is an important consideration in almost all algorithms. The most commonly used layout is the horizontal database layout and vertical one [2]. In both layouts, the size of the database is very large. A large database to be transformed into a smaller one with all properties of its original layout is expected. Database encoding is a new presentation, which can reduce the size of database and improve the efficiency of algorithms. Instead of maintaining a large table in the transaction database, one table is created with only two columns. The first one is the transaction identifier and another is for the entire items that occur in the transaction. All items in one transaction are converted into only one number that has all properties of these items. By this way, the new database is much smaller than the previous one and can be loaded into memory easily. So the cost of memory is reduced. According to the assumption that only one number represents an itemset, when converting an itemset into a number, a measure attribute is defined, which is a numerical attribute associated with every item in each transaction in the database layout. A binary number expresses a numerical attribute, that is, those items that are occurring in one transaction are depicted with 1 and all the other items are represented with 0. The transaction measure value, denoted as tmv (I_p , T_q), is a value of a measure attribute related to an item I_p in a transaction T_q . tmv (I_p , T_q)= 0 means item I_p does not occur in the transaction T_q, while tmv (I_p , T_q)= 1 means item I_p occurs in the transaction T_q . In table 1, for example, tmv (I_4, T_1) is equal to 1. Any item I_p in the set of items is encoded as one prime number, denoted as E (I_p). Prime numbers are used because any number except 1 and themselves cannot divide them. For any item I_p in the transaction T_q, a new measure denoted as M (I_p, T_q) is equal to the product of tmv (I_p, T_q) and its encoding number E (I_p) is assigned. This value is gotten by Eq. (1). After this step, for all I_p and T_q, if M (I_p, T_q) equal to 0, then convert M (I_p, T_q) into 1. This operation is described in Eq. (2). For any transactions, the value M_{Tq} is equal to the multiplication of all M (I_p, T_q). The value of M_{Tq} is represented in Eq. (3).

$$\mathbf{M\ (I_p, T_q) = tmv(I_p, T_q) \times E(I_p)} \quad (1)$$

$$\mathbf{For\ all\ (I_p, T_q)\ If\ M\ (I_p, T_q\ ) = 0 => M(I_p, T_q\ ) = 1} \quad (2)$$

$$\mathbf{M_{Tq} = \Pi\ (I_p, T_q)} \quad (3)$$

For any itemset I=(I_{p1}, I_{p2}, …, I_{pn}), there is one value denoted as M_I is equal to the multiplication of all $E(I_p)$ if its I_p occur in I, as described in Eq. (4). The value M_I shows the number corresponding to itemset I. And then this number can be used instead of itemset I.

$$M_I = \prod_{\forall I_P \in I} E(I_P) \quad (4)$$

With this encoding, instead of maintaining all tmv (I_p,T_q) for every item and transaction, the value M_I can be stored for every transaction. Example 1 shows the result of using this technique.

Example 1: Convert vertical database layout

Table 1

Binary representation

T_{ID}	I_1	I_2	I_3	I_4
1	1	0	0	1
2	0	0	0	1
3	1	1	1	0
4	0	0	1	1

Table 2

Prime number form

I_p	E_{Ip}
I_1	7
I_2	5
I_3	3
I_4	2

Table 3

Single number representation

T_{ID}	M
1	14
2	2
3	105
4	6

In the case of large databases, encoding an item to one prime number may cause some problems especially in computing the value M (which involves multiplication) for any transaction. As a solution, the database can be divided into smaller parts vertically by having correlated items in one part. Then every part of the database is encoded to one column. Frequent itemset mining is done independently and association rules in every part are discovered. After an encoding of this form, the database has a smaller number of columns which leads to better performance and efficiency.

2.2 Anti-Apriori Algorithm

All Apriori-like algorithms for itemset mining start from finding frequent 1-itemset. In these algorithms, finding frequent itemset is done in bottom up manner. Different from these algorithms, a new algorithm called as anti-Apriori is used, in which the discovery of frequent itemset is done in up to down style. It means that the large frequent itemset are found at first and then all of their subsets (that are certainly frequent) are extracted. [2]. In this technique, it is supposed that any frequent itemset must be at least one time occurs in the transactions lonely (without any other items that are not member of that itemset).

In other words, if itemset (I_1, I_2, I_3) is frequent, the itemset at least in one transaction without any other items, such as shown in table 4.

Table 4

Frequent itemset presentation

Itemset	I_1	I_2	I_3	……
T_{id}	1	1	1	0000000000

In this method for every transaction T_q, the GCD (greatest common divisors) between M_{Tq} and M_T corresponding to other transactions are computed and frequencies of these greatest common divisors are stored in GCD-set. GCD-set is the candidate for frequent set. For any GCD in GCD-set, if its frequency is above the required threshold, it will be selected and inserted into the FGCD-set (frequent GCD itemset). For every transaction maintained, a set is denoted as GCD_{Tid}, composed of GCDs and frequency of any GCD. For example, if GCD-set and frequency between first transaction and other transactions is equal to GCD1={(42,3), (6,8), (21,2), (15,4), (105,1)} and the required threshold for support is equal to 7 and then the set (6,8) has a frequency equal to 8, greater than 7, and then 6 is inserted into FGCD-set.

Discovering association rules is based on all FGCD, which has been found in the previous phase. Measure M corresponds to any frequent itemset maintained in FGCD-set. Every measure M in FGCD-set is decomposed into the multiplication of prime number and each prime number corresponds to one item. The itemset that corresponds to M is identical and frequent, and all subset of it must be frequent. Every M in FGCD-set is decomposed into a candidate head Y and a body X=M/Y. This algorithm iteratively generates candidate heads $_{k+1}$of k+1 size, starting with k=1. If the head and the body are interesting and valuable, the confidence *C*

of the rule X=>Y is computed as the quotient of the supports for the itemset. *C* =Support(M) /Support(X) (Support(X) is computed by counting the number of M_{Tid} that can be divided by X). If any rule has a *C* greater than or equal to the given threshold for confidence, the rule will be appended into association rules.

2.3 Apriori, AprioriTid, and AprioriHybrid

The Apriori and AprioriTid algorithms generate the candidate itemsets to be counted in a pass by using only the itemsets found large in the previous pass, without considering the transactions in the database. The basic intuition is that any subset of a large itemset must be large. Therefore the candidate itemsets having k items can be generated by joining large itemsets having k-1 items, and deleting those that contain any subset that is not large. This procedure results in generation of a much smaller number of candidate itemsets. The AprioriTid algorithm has the additional property that the database is not used at all for counting the support of candidate itemsets after the first pass [5]. Rather, an encoding of the candidate itemsets used in the previous pass is employed for this purpose. In later passes, the size of this encoding can become much smaller than the database, thus saving much reading effort. Based on the observations of Apriori and AprioriTid, a hybrid algorithm which is called as AprioriHybrid uses Apriori in the initial passes and switches to AprioriTid when the candidate itemset at the end of the pass will fit into the memory.

III. THE PROPOSED METHOD FOR DISCOVERY OF TEMPORAL ASSOCIATION RULES

The problem of finding association rules can be done as follows: (1) Generate all combinations of items that have fractional transaction support above a certain threshold, called *minsup*. Those items are called *large* itemsets, and all other combinations whose support is below the threshold are called *small* itemsets. (2) Use the large itemsets to generate the association rules. For every large itemset *l*, find all non-empty subsets of *l*. For every such subset *a*, output a rule of the form $a \Rightarrow (l\text{-}a)$ if the ratio of *support* (*l*) to *support* (*a*) is at least *minconf*. If an itemset is found large in the first step, the support of that itemset should be maintained in order to compute the confidence of the rule in the second step efficiently.

3.1 Temporal Association Rule Mining

An entire temporal database can be used for mining association rules. However, the resulting rules may not be interesting since data accumulated over a longer time span is used. Customer behavior and preferences change over time. Observation of these changes provides useful information for various decision-making purposes. A temporal database provides the opportunity to observe these changes, by mining various subsets of a temporal database.

This approach aims at detection of the changes in the association rules. The changes in association rules occur over periods of time, and include a decrease (increase) in the support (confidence) of an association rule and addition (removal) of itemsets from a particular itemset. In order to observe how a frequent pattern fluctuates within certain periods, it is necessary to extract association rules from datasets accumulated over consecutive time periods. In this process, the vital parameters are the length of interval, through which the set of association rules will be extracted, and the minimum support and confidence for these rules. The changes in association rules can be observed through three steps. 1. Determining the time period, 2.Identifying temporal association rules and 3.Presentation of association rules with changes.

In the first step, the length of the time interval and the number of such intervals are decided. Then, from the database, subsets of temporal data for each interval are extracted. For example, the association rules may be searched for weekly, monthly and so on until the specified time and date. In the second step**,** the association rules are discovered within a certain period, and this process is repeated for the whole database for consecutive time periods. The algorithm for this is as follows:

1 Initialize ‘start’
2 end = start + L
3 **while** start ≤end **do**
4 **begin**
5 R [start, end] = R <X, f_I> [start, end]
6 compute A[start, end]
7 start = start + L + y
8 end = start + L
9 **end**

Algorithm1: Algorithm for repeated rule mining

The method undertaken is to retrieve the data for the subintervals from the original interval. Datasets for the subinterval are smaller and fits the memory easily. The operations done for one subinterval are done in a

sequence for all subintervals. This method utilizes the results of the smaller datasets. This lends itself to the use of parallel processing techniques. The Partition algorithm proposed in can be efficiently used in this method.

3.2 Calculation of changes in Temporal-Association Rules

Individual itemsets change as time progresses. An itemset that was small can become large, large itemset can become small, or itemsets may remain large or small. Small itemsets that are moving towards large are defined as emerging. Large itemsets that are moving towards small are submerging. A small (large) itemset that becomes large (small), i.e. support is above (below) *minsup*, is said to have emerged (submerged). Therefore the following can be identified: (i) itemsets that are currently emerging (submerging), (ii) which of these itemsets have the potential to emerge (submerge) within the next time interval or n intervals.

Partitioning the Itemset Space partitions the space of itemsets. It can also be viewed as all the possible transitions for an itemset X from the original database at time‘t’ to incremented database at time ‘t+i’. The support count (*SC*) of an itemset is the number of transactions that an itemset satisfies. After extracting the patterns for small periods, the large itemsets for the larger time periods can be obtained by using the patterns extracted before. This is similar to what is present in the Partition algorithm. The only feature of importance is that the minimum support for small periods should be held small enough in order to prevent missing any large itemsets. The minimum support for larger time intervals should be adjusted for reliable rule mining. These parameters cannot be fixed earlier because they are user and application dependent.

IV. PERFORMANCE EVALUATION

The anti-Apriori algorithm in combination with the encoding method decreases the size of the database and also leads to reduction in the number of passes over the static database. The performance of AprioriHybrid relative to Apriori and AprioriTid for large datasets is as follows. AprioriHybrid performs better than Apriori in almost all cases. In general, the advantage of AprioriHybrid over Apriori depends on the size of the candidate itemset that declines in the later passes. If there is a gradual decline in the size, AprioriTid can be used for a while after the switch, and a significant improvement can be obtained in the execution time. By using this encoding method the efficiency of Apriori, AprioriTid, and AprioriHybrid can be improved significantly.

Database encoding has been applied to a static database prior to the application of Apriori or anti Apriori. To make it scalable, the same has been applied to a dynamic database, which involves time constraints. The Apriori family and anti-Apriori algorithm combined with an encoding method have been applied for association rule mining on a temporal database. The encoding method reduced the size of the memory required as shown in fig1. The use of anti-Apriori reduced the number of scans on a temporal database.

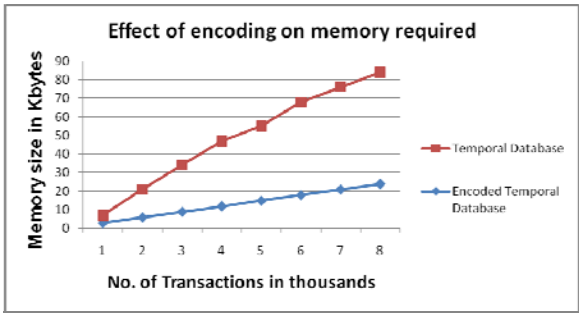

Fig.1. Effect of encoding on memory required

Considering the telecommunication temporal database, which addresses complaints, performance of the Apriori family of algorithms and the Anti-Apriori algorithm is as given in the following figures.

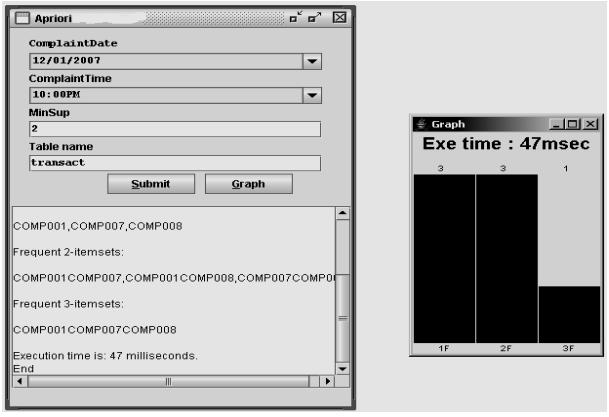

Fig .2. Performance of Apriori

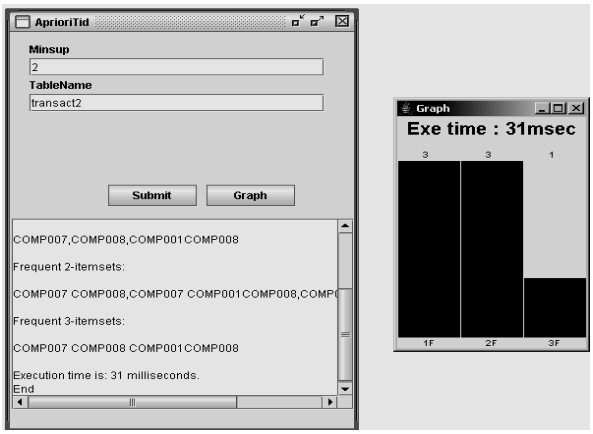

Fig .3. Performance of AprioriTid

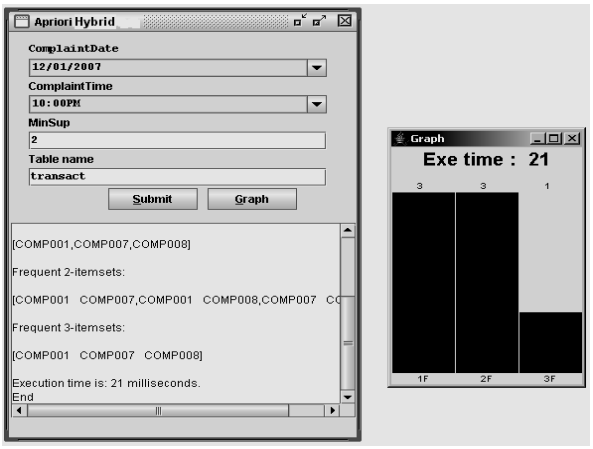

Fig. 4. Performance of AprioriHybrid

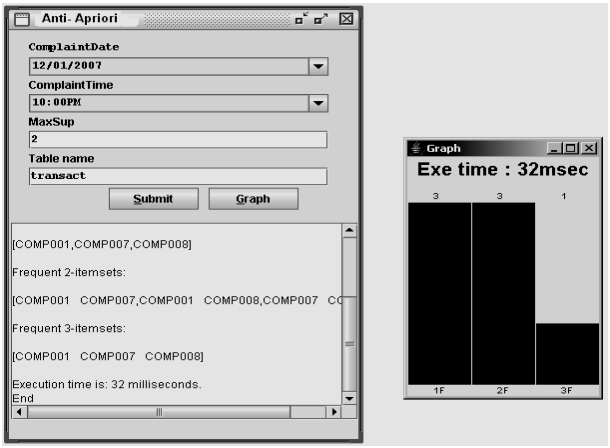

Fig .5. Performance of anti-Apriori

The various algorithms considered in this paper differ in their performance based on the time factor. The time is measured using WEKA as the tool in the association rule mining process.

The performance comparison in terms of the execution time for the various algorithms is shown in fig 6.

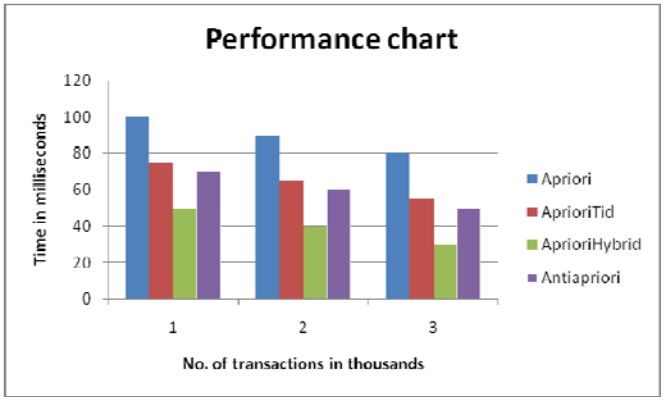

Fig.6. Performance comparison

The method for discovering temporal association rules that changes with time tracked the changed rules in an effective manner, but the effectiveness was dependent on the users and the applications.

V. CONCLUSION AND FUTURE WORK

This paper presented the impact of the Apriori family of algorithms and the anti-Apriori algorithm on an encoded temporal database for association rule mining. Each of the algorithms had a different impact and produced effective results. The AprioriHybrid and anti-Apriori had better performances in terms of execution time. Therefore lower complexities of time and space had been obtained leading to better formalization of database metrics. The method for discovering temporal association rules that change over time produced effective results. This can still be improved by (i)optimizing the process of discovering association rules in each interval using better algorithms, and (ii) optimizing the choosing of the time interval size, leading to more advantageous openings in the formalization of database metrics.

VI. REFERENCES

[1] M.H. Dunham, Data Mining Introductory and Advanced Topics, Tsinghua University Press, Beijing, (2003).

[2] T.Wang, P.L.He, Database Encoding and an Anti-Apriori Algorithm for association Rules Mining, Machine Learning and Cybernetics, (2006) 1195-1198.

[3] A. Savasere, E. Omiecinski, and S. Navathe, An Efficient Algorithm for Mining Association Rules in Large Databases, In Proc. of the 21st VLDB'95, Zurich, Switzerland, (1995).

[4] R. Agrawal, T. Imielinski, and A. Swami, Mining Association Rules between Sets of Items in Large Databases, In Proc. of ACM SIGMOD'93, Washington, DC,(1993),207-216.

[5] R. Agrawal, R.Srikant, Fast algorithm for mining association rules, The International Conference on Very Large Data Bases, (1994) 487–499.

[6] H.Ning, H.Yuan, S.Chen, Temporal Association Rules in Mining Method, Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences, (2006).

[7] A.U.Tansel, S.P.Imberman, Discovery of Association Rules in Temporal Databases, Fourth International Conference on Information Technology (ITNG'07),IEEE computer society, (2007).

Corresponding author .Tel. +91 4288 274741;
fax: +91 4288 274745, E-mail
addresses:rc.balsubramanian@gmail.com
(C.Balasubramanian),
drkduraiswamy@rediffmail.com,
drkduraiswamy@yahoo.com(Dr.K.
Duraiswamy)