Mining Frequent Itemset for Temporal Data using Hybrid Algorithm

A.Monika
CSE Department
TRP Engineering College
Trichy,TamilNadu

A.Niveda Rajlakshmi
CSE Department
TRP Engineering College
Trichy,TamilNadu

M.Pavithra
CSE Department
TRP Engineering College
Trichy,TamilNadu

S K. Karthika
Assistant Professor
CSE Department
TRP Engg College
Trichy,TamilNadu

Abstract—Temporal data are data with time-stamping information. The result of data mining also depends on time stamp information. Traditional techniques assume that datasets are static and the rules being used are relevant for the entire dataset. In this paper, we are trying to improve the efficiency of mining frequent itemsets on temporal data. To solve this problem we produce a list based data structure and pruning technique that allow us to mine frequent itemset from a uncertain frequent pattern.

Keywords—Uncertain Frequent Pattern, Temporal Data, Time Stamp, Itemsets

DATA MINING is the process of discovering interesting patterns and knowledge from voluminous amounts of data [1]. One of the most important applications of data mining is the analysis of transactional data. Databases which originate from transactions in a supermarket, bank, department stores and, etc., are all inherently related to time. These are called temporal databases which are databases that contain time-stamping information [3]. Capturing the co-occurrence of items in transactions was first proposed by Agrawal et al. [2]. For example, given a transactional database of a supermarket, we may have {milk, bread} bought together with support of 20%. It means that 20% of all transactions contain milk and bread together. One important extension to frequent pattern mining is to include a temporal dimension [4]. For example, milk and bread may be ordered together in 80% of all transactions between 7 and 9 A.M while their support in the whole database is 20%. In fact, interesting patterns are often related to the specific period of time; therefore, the time during which they can be observed is important.

<table>
<thead>
<tr>
<th>ID</th>
<th>ITEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bread, milk</td>
</tr>
<tr>
<td>2</td>
<td>milk, rusk</td>
</tr>
<tr>
<td>3</td>
<td>milk, biscuit</td>
</tr>
</tbody>
</table>

From the above, the conclusion is we can obtain different pattern for different time intervals. Discovering such patterns may lead to useful knowledge. The discovery of such association rules has been discussed in the literature. In this context, sequential association rules [5], time intervals for association rules [6], [7] and calendar-based association rules [4], [8] are some interesting studies in recent years.

In this paper, we conduct a study on developing an efficient algorithm for mining frequent patterns and their related time interval from transactional database. Then a new algorithm is proposed based on two thresholds, support, and density as a novel threshold. Frequent itemsets are discovered and neighbouring time intervals are merged.

The rest of the paper is organized as follows. In Section II, we discuss some related work in comparison with our work. In Section III, the proposed algorithm is presented in details. The notion of TCs is described and the essence of a density threshold is illustrated with an example. In Section IV, we present the experimental evaluation of our algorithms using synthetic datasets and give analysis on experimental results. In Section V, we conclude the paper with some discussions.

I. RELATED WORK

Association rule was first proposed by [2]. It has two part, finding frequent itemsets and generating association rules. The major and time consuming part of the algorithm is discovering frequent itemsets and generating association rules is straightforward. Therefore, in our literature review, we consider association rules the same as frequent itemset mining. Classification association rule uses class tables and maximum support count which is efficient and eliminates infrequent item sets. But duplicate data elimination is difficult [9]–[11], context-based association rule uses association rule and accuracy, frequency and performance is high but sequential and parallel construction of table is difficult due to time factor [12]–[14], negative association rule [15], fuzzy association rule [16], generalized association rule [17]–[19] are some of the research areas in this field. Among these extensions, the time attribute of the transactions has attracted many researchers to discover frequent itemsets over time computational time is low. The problem of discovering association rule that display regular cyclic variation overtime was first proposed by Ozdenetal. [20]. Two new algorithms were presented, sequential algorithm and interleaved algorithm, to discover hourly, daily, weekly, etc., patterns. Some pruning techniques also used to improve the performance of the algorithms. It should be noted that by their method, each cyclic rule holds in every cycle with no exception. However, in real life, patterns are not perfect. Therefore, Han et al. [21] proposed partial periodic pattern mining in temporal database. Discovered association rules show behavior in some but not all points in time. The work of Ozdenetal in [20] was extended by Ramaswamy et al. in [22] to discover user-defined temporal patterns in association rules. The idea of using calendar algebra was proposed to describe interesting patterns, however, it requires user’s prior knowledge to define calendar...
expressions. In [6], Ale and Rossi proposed the idea of extracting rules over a specific period of time that is shorter than the whole database. The lifetime of each item was used to define time intervals. The concept of temporal support was introduced for the first time and algorithm a priori was modified to incorporate time. Li et al. [4], [23] proposed an approach to discover association rule holds either in all or some time intervals. Instead of using cyclic [20] or user-given calendar algebraic expressions [22], calendar schema is used to restrict the meaningful time intervals. Since items have different exhibition periods, algorithm progressive-partition-miner (PPM) was proposed to discover patterns in such databases [24], [25]. The basic idea of PPM is to first partition the database in light of exhibition periods of items and then progressively accumulate the occurrence count of each candidate 2-itemset based on the intrinsic partitioning characteristics. It’s worth noting that exhibition period of items in [24] and [25] is the same as lifetime of items in [6]. The study of [24] and [25] was further extended [26], considering the importance of time intervals. Using the same concept proposed in [6] and [24], algorithm segmented-progressive-filter (SPF) was introduced in [27] to first segment the database into sub databases in such a way that items in each sub database will have either starting time or ending time. Then, for each subdatabase, SPF progressively filters candidate 2-itemsets with cumulative filtering thresholds either forward or backward in time. Lee and Lee [28] also used calendar algebra to discover association rules. Since human being tends to be uncertain, fuzzy set theory incorporated to help the construction of desired time intervals. This is similar to the work of Ramaswamy et al. [22], but it is more flexible which is because of the fuzzy sets and fuzzy operators. Junheng [29] added some modification to the algorithm SPF [27] to increase efficiency. Results show that its performance outperform from prior methods. Huang et al. [30] explore the previous studies [27], [29] to remedy drawbacks of their algorithms. Algorithm TWAIN was presented to find association rules that are absent when the whole range of the database is evaluated altogether. Mahanta et al. [8] presented a method which is able to extract different types of periodic patterns that may exist in a temporal dataset. The user does not need to specify the periods in advance. Set operation called set superimposition was used for storing periods associated with item-sets. Adhikari et al. [31] enhanced the study of [8] by proposing a new data structure for storing and managing patterns. They also introduced minor changes to their algorithm to improve its performance. Lee et al. [32] proposed an efficient algorithm for discovering fuzzy periodic association rules. They investigated the problem of discovering regular time intervals, i.e., the periodicity. To overcome the difficulties of finding precise time interval, fuzzy calendar was proposed. Their method scan the database at most twice for discovering association rules and related time periods. Matthews et al. [33] used genetic algorithm to find temporal association rules for the first time. Genetic algorithm was employed to simultaneously search the rule space and temporal space. They used the strength of evolutionary algorithms in searching for association rules and optimizing parameters of induction process (support and confidence values). They further extended their works for finding fuzzy temporal association rules where Duplication can be minimised to very little amount and provides good response time [34], [35]. The problem of it is it doesn’t handle very large number of data.

[37] proposed a new form of association rule, i.e., association rule with time windows. The main purpose of their study was to find the time intervals for association rules which may be arbitrary in length and not user specified. They further optimized the process of finding time windows by mathematical modeling [7]. Saleh and Masseglia [38] deal with this problem from another viewpoint. The concept of solid itemset mining was proposed to find the subsets of database that contain frequent itemsets and then the proposed algorithm was developed based on that. The problem is that usage of tree structure and other nodes uses more memory. Like above studies, the challenge was to find the optimal time interval.

### A. Proposed Algorithm

We propose a List-based UNcertain frequent pattern mining Algorithm (LUNA), which is an efficient and exact method for mining uncertain frequent patterns based on novel data structures and mining techniques that can guarantee the correctness of the mining results without any false positives. The main contributions of this paper are as follows:

1. Proposing a new paradigm for efficiently mining uncertain frequent patterns from uncertain databases.
2. Devising novel data structures based on list forms that can store uncertain data more efficiently compared to tree structures of previous approaches.
3. Proposing an algorithm that can extract exact results of uncertain pattern mining without any false positives using the suggested data structures.
4. Suggesting new pre-pruning techniques that can improve the mining performance of the proposed algorithm by preventing useless mining operations from being caused, and effective strategies that can speed up the mining operations without any additional memory consumption.

#### LUNA algorithm

In this section, we describe an overall procedure of LUNA, the proposed exact UPF mining algorithm that applies the suggested data structures and mining techniques as well as various performance improving techniques. Fig. 10 shows the overall pseudo code of the proposed algorithm. LUNA first scans a given uncertain database, D, and computes the expSup value for each item composing D (lines 01–03). After that, the algorithm discards items with lower expSup values than a given minSup and calculates a support ascending order of the remaining items (lines 04–05). Moreover, these items are extracted as 1-length UPF s (line 06). After a series of initialization processes for UP-Lists are conducted (line 07), the second database scanning operations are performed to construct UP-List s (lines 08–10). In this process, tuple data and Max information for each UP-List are updated according to the method mentioned in Section 3.4.1. Thereafter, the algorithm performs a series of works for constructing CUP-List s from the generated UP-List s (lines 11–26). Once an UP-List is selected (line 11), the algorithm initializes a set of CUP-List s, C , and a prefix itemset, pref (line 12), and sets the item of the currently selected UP-List to pref (line 13). Then, another UP-List after the currently selected one is chosen again (line 14).
After calculating an overestimated expSup value from the selected two UP-Lists, the algorithm performs subsequent tasks for constructing a CUP-List from them if the overestimated value is not smaller than minSup (lines 15–21). In the middle of generating the CUP-List, it can be possible to prune it according to the proposed pruning techniques. The proposed algorithm improves its mining efficiency by utilizing the existential probability information of the prefix. If expSup of the constructed CUP-List is higher than or equal to minSup (line 17), the CUP-List is stored into a set of CUP-List s for the next mining task (line 18) and the pattern corresponding to the list is extracted as a valid UFP (line 19).

After a series of works for the UP-List selected first, u0, are finished, if the number of CUP-List s contained in C is higher than 1, sub-procedure LUNA_growth is called to mine UFP s with longer lengths recursively. If the algorithm conducts the above operations with respect to all the UP-List s, it can provide a complete set of UFPs from the given uncertain database.

The sub-procedure, LUNA_growth, is recursively called to construct CUP-List s and extract UFP s with 3 or longer lengths (lines 28–43). The overall flow of this sub-procedure is similar to the part between lines 11 and 26 in the main procedure. The major differences between them are that (1) the prefix information is continually updated to perform the mining operations more efficiently for each call of the sub-procedure (line 30) and (2) CUP-List s for longer patterns are recursively generated from previous CUP-List s until there is no more CUP-List to be combined (lines 31–42).

B. Equations

1. Expected Support of a Pattern (expSup)

Let \( X \) \( i_1, i_2, \ldots, i \), be a pattern composed of items belonging to \( F \). Then, the existential probability of \( X \) for each transaction \( T \), \( p(X, T) \), is computed as follows.

\[
p(X, T) = \forall i \in T \exists x \in X \ p(x, T)
\]

C. Figures and Tables

An example of uncertain datasets

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1.0</td>
<td>0.9</td>
<td>0.6</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>200</td>
<td>0.9</td>
<td>0.9</td>
<td>0.7</td>
<td>0.6</td>
<td>0.4</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>300</td>
<td>–</td>
<td>0.5</td>
<td>0.8</td>
<td>0.9</td>
<td>–</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>400</td>
<td>–</td>
<td>–</td>
<td>0.9</td>
<td>–</td>
<td>0.1</td>
<td>0.5</td>
<td>–</td>
</tr>
<tr>
<td>500</td>
<td>0.4</td>
<td>0.5</td>
<td>0.9</td>
<td>0.3</td>
<td>–</td>
<td>–</td>
<td>0.3</td>
</tr>
<tr>
<td>600</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0.9</td>
<td>0.1</td>
<td>0.6</td>
<td>–</td>
</tr>
<tr>
<td>700</td>
<td>0.9</td>
<td>0.7</td>
<td>0.4</td>
<td>0.6</td>
<td>–</td>
<td>0.9</td>
<td>–</td>
</tr>
</tbody>
</table>

Algorithm 3.1: Algorithm for mining frequent itemsets with LUNA Algorithm

![Fig. Story Board of the proposed algorithm.](image-url)
IV. COMPUTATIONAL STUDY

In this section, experimental study is carried out by applying the proposed algorithm. The performance of the algorithm is examined in the experiment on synthetic data. In the following, we describe the process of generating synthetic dataset.

A. Generation of Synthetic Data

A synthetic dataset is chosen rather than a real dataset so that a controlled experiment can be conducted to validate the efficacy of our approach. The IBM quest synthetic data generator [5] has been used to generate a dataset for experimentation. As the dataset is nontemporal, so it cannot be...
used directly. We have incorporated the temporal feature in the
dataset; so, it can be a temporal dataset and can be handled by
our algorithm. We developed a program for this. The program
takes starting date, ending date, and synthetic data as inputs
and randomly distributes data between the starting and ending
dates. Dataset features such as number of items, number of
transactions, average transaction length, and time span of the
dataset can be set by the user. As there is no guarantee that the
generated dataset contains pattern with temporal information,
augmentation to the original dataset is proposed. The
augmented pattern act as a target, and the aim of our algorithm
is to discover this target pattern. The augmentation method is
now described. The a priori algorithm is used to identify
frequent patterns on original dataset. One of the patterns is
selected and augmented in a new copy of the dataset. After
selecting a pattern, augmentation occurs by inserting it as a
transaction to the intended time interval with no additional
items; so, no unexpected correlations between items are
introduced. The result is a dataset that has an increase in the
selected pattern in the intended time interval. The objective is
to find this pattern by our algorithm.

B. Experiment on Augmented Dataset
The dataset is a synthetic database containing transactions
madeupby100itemsfrom01/01/2015-00:00:00to01/01/201600:00:00.Thenapatternchosenastarget
solution. The goal behind using such a large dataset for analysis
is to make sure that our proposed algorithm is capable of
handling any frequent itemset mining problem. Considering
time hierarchies of the time stamp, i.e., month, day, hour, a
specific time interval for augmenting a pattern is chosen. From
a priori, itemset {61,70,91} is a frequent pattern. We assume
this pattern is frequentbetween7and9A.M.,1stand2nddaysofthefirstthreemonths of 2015. This time interval contains 3×2×3 BTCs. For
each BTC, 100 transactions are augmented which include only
items 61, 70, and 91. Our algorithm should be able to find this
itemset and its temporal information. The proposed algorithm
is coded by Java programming language and is run on a
notebook computer using an Intel Core i7, 2.5 GHz CPU with
16 G of memory and running windows 10. As we expected, targetsolution, i.e., pattern {61,70,91} was
mined in the specified time interval at a reasonable time. Fig. 4
shows the relationship between the solution time and
minimum supports at different densities. As we can see, the
lower the minimum support, the higher the solution time is. It
is also worth noting that with support level less than 20%,
many other useful patterns is found in addition to the target
solution. Although there is not a considerable differences
between solution time with different density threshold, but
now we guarantee that data patterns found are valid and no
overestimating of the time periods has been occurred.
This article has been accepted for inclusion in a future issue of
this journal. Content is final as presented, with the exception
of pagination.

Fig. 4. Solution time of the proposed algorithm against
different minimum supports (a) α = 30%. (b) α = 50%.

C. Experiment on Real Dataset
As the density threshold is proposed in this study and is
defined for the proposed algorithm, we could not compare the
results of our method with others. But instead, we use a real
dataset of a grocery store to compare the results of this section
with Section IV-B. The dataset is available from website at
http://fimi.ua.ac.be/data/. We use the same procedure which is
proposed in Section IV-A to change the nontemporal dataset to
temporal one. The total number of items is 331 and total
number of transactions is 9835. We run the program with
minimum support 50% and α equal to 20%. Total of 88 patterns
was found by our algorithm in 1222 seconds
(20.3 minutes). As we can see, runtime of the algorithm heavily
depends on minimum support threshold and number of items.
Increasing in the number of items and transactions, adds more
complexity to the problem and though runtime of the
algorithm increases. As we can see in Table III, soda, bottled
water, and whole milk were old together during specific time
interval. Therefore, many marketing strategies can be applied
to increase the profit of the store. For example, according to
the cross-selling strategy, items with less interest from the
customers perspective can be placed between frequent patterns
in specific time intervals to increase the selling rate.

D. Analysis of Scalability
In this section, for analyzing the scalability and fair
comparison of the performance of the algorithm, several
experiments are carried out and then we compare the results
of our algorithm with [8]. Several experiments are designed
and in each experiment, the objective is to find a target solution in
an augmented dataset. The obtained results from numerous
runs of the algorithm with different number of items and
transactions at different periods of time is shown in Table IV.
In all of the experiments, the minimum support is 25% and
density rate is 50%. Also the augmented target is the same for
all experiments. We can see how the proposed algorithm
perform well in almost all of the experiments. By comparing
experiments 1 and 2, we can infer that solution time increases
when number of items and average transaction length increase
which is reasonable. Comparing experiments 3 and 4 and also 5
and 6, it is seen that solution time increases, when the time
period is increased. This seems logical since increasing the
time interval, increases the solution space. We also compare
the results of our algorithm with the work of [8]. They
considered a small dataset consisting of time-stamp and items
sold at the corresponding dates which is collected over two
months, i.e., January 2000–February 2000. The total number of
items is 5, total number of transactions is 60. The dataset
is given in Table V. In order to compare the results, we use the
same minimum support, i.e., 50%, and run the program. The
runtime of our algorithm is 0.079 seconds and results are
shown in Table VI. In terms of the number of detected
patterns, the number of patterns detected by [8] is less than
ours. In terms of the quality of the detected patterns, the results
show that some of the patterns extracted by [8] are not valid.
For example, [8] stated that itemset {1,2} is frequent in period
[1–2] [4–9]. However, the results of our algorithm indicate
that this itemset is frequent in Time stamps Items Time stamps
Items Time stamps Items the period [4–9] in the first month,
and in the period [3–10] in the second month. Even with more
accurate survey of dataset, it is clear that the tiny time frame in
which itemset {1,2} become frequent is [5–8] and not [4–9].
Our proposed algorithm is able to detect this defect and has
extracted more accurate patterns.
V. CONCLUSION

In this paper, we studied the mining of frequent itemsets along with their temporal patterns. Some patterns are held during some time intervals while others may happen periodically. The main feature of our proposed algorithm is that a new notion of TCs is presented to consider time hierarchies in data mining process. It enables us to find different kinds of temporal patterns for small or medium-sized datasets, it can find the solution in less than one minute. We applied our algorithm to market basket dataset with time stamps. From managerial viewpoint, results of our algorithm can help managers to make better decisions. Duplicate data are eliminated easily. It is easier to implement in large databases. Performance is high compared to original apriori

REFERENCES