A Novel Technique for Temporal Frequent Itemset Mining

Md. Ghouse Mohiuddin[1], Dr. L.Srinivasa Reddy[2]
[1]*Asst. Prof., Department of Computer Science, Palamuru University, Mahabubnagar*
[2]*Associate Professor, Dept. of Information Technology, CBIT, Hyderabad*

Abstract: **Pattern mining is a powerful tool for analyzing big datasets and extracting interesting patterns from the dataset. Temporal datasets include time as an additional parameter as time-stamps that affects the data mining results. Traditional data mining techniques of finding frequent itemsets consider the static data sets and the instigated rules are relevant across the whole dataset. However, this is not the case in temporal data because, in temporal data, there are certain itemsets that are frequent over a specific period of time but would not be extracted by traditional data mining methods since their support is very low over the whole dataset. Our aim is to extract such patterns with their time intervals. In this paper, we propose a method that is able to extract different types of patterns that may exist in the temporal dataset and it is not needed by the user to specify the time periods in advance. Here we consider the time stamps as hierarchical data structures and our algorithm extracts the periodic patterns along with the time intervals.**

Key Words: **Frequent Itemset, Local Temporal Frequent Itemset, Set Superimposition, Time-Cube, Basic Time-Cube.**

I. INTRODUCTION

Data mining is the process of exploring and analyzing data from different perspectives, using automatic or semi-automatic techniques to extract knowledge or useful information and discover correlations or meaningful patterns and rules from large databases. Using these patterns, it is possible for business enterprises to identify new and unexpected trends, and subtle relations in the data and use them to increase revenue and cut costs. Data mining has been proven to be advantageous in many areas. Data mining involves many different techniques and algorithms to accomplish different tasks. All these algorithms attempt to fit a model to the data. The algorithms examine the data and determine a model that is closest to the characteristics of the data being examined.

One of the most important characteristics missed by traditional data mining systems is their capability to record and process variable time-related aspects of real-world databases. In simple words, traditional data mining techniques lack the ability to analyze variations of data over time and treat them as ordinary data. Examples of temporal data include stock market data, banking data, production data, maintenance data, web mining, and point-of-sale records. Temporal data mining which mines or extracts knowledge and patterns from temporal databases is an extension of data mining with the capability to include time attribute analysis. Due to the importance and complexity of the time attribute, a lot of different kinds of patterns are of interest. Analysis of transactional data is one of the most important applications of data mining. Finding the association or correlations among items in transactions was first proposed by Agarwal et. Al..[1]. For example, in a given transactional database of a supermarket, we may have {milk, bread} bought together with the support of 25%. It means 25% of all transactions contain milk and bread together. Databases that evolved from the transactions in a supermarket, banks, and departmental stores are all inherently related to time. These are called temporal databases that contain time-stamping information. One important extension to frequent pattern mining is to include temporal features in the dataset. For example, milk and bread may be purchased together in 80% of all transactions which occurred between the time interval 7 am to 9 am, while their support in the whole database is 25%. In fact, interesting patterns can be extracted from a specific time interval, therefore the time interval during which they are observed is important.

From this, it can be concluded that different patterns can be obtained from different time intervals. Discovering such patterns may bring useful knowledge. The discovery of such frequent itemsets and association rules has been discussed in the literature. In this context, sequential association rules [2], Cyclic Association rules [3], Mining Temporal Association Rules [5&6], and calendar-based association rules [15&21] are some interesting studies in recent years.

The problem is to find valid time intervals during which frequent patterns hold and the discovery of periodicities that patterns include. In this paper, we conducted a study on developing an efficient algorithm to mine frequent patterns and their related time intervals from transactional databases. Here we first introduce the new notation Time Cube (TC) to consider time hierarchies in the mining process. Then we propose a new algorithm based on two thresholds, support and density as a novel threshold. Frequent patterns are discovered and those with neighbouring time intervals are merged.

The remaining of this paper is organized as: Section 2 provides a related work of the various solutions proposed for temporal frequent itemset mining, and Section 3 describes the terms, definitions, and terminology used in this paper. Section 4 gives the proposed Algorithms & procedures, Section5 describes the Implementation and results, and Section 6 concludes the study with future research directions.

II. RELATED WORK

The first study related to the association rules discovery is presented by [1]. This paper introduces the problem of mining a large collection of basket datatype transactions for association rules between sets of items with some minimum specified confidence and presents an efficient algorithm for this purpose. It has two parts finding frequent itemsets and generating association rules. The problem of discovering association rules that display regular cyclic variations over time was first proposed by [2]. The cyclic patterns are those patterns occurred after every regular cyclic variation over time. Here two algorithms are proposed: the sequential algorithm and interleaved algorithm. This technique extracts hourly, daily, monthly, and quarterly patterns. The pruning technique is applied to improve the performance of the algorithm. This cyclic analysis helps in trend analysis and market forecasting. It should be noted that by their methods, each cycle rule holds in every cycle with no exception. However, in real life patterns are not perfect. The Discovery of interesting patterns in association rules is presented by [3], where the author proposes a new technique based on the technique proposed by Ozden et al. This technique uses user-defined temporal patterns for rule discovery. Calendar algebra is used to process the time cycle. But this technique requires prior knowledge of calendar data expression. In a real-life scenario, the product purchase history for a defined cyclic period varies continuously. The product sold in one cycle may or may not be present in the next cycle with the same frequency. Hence cyclic pattern discovery is not an appropriate solution for pattern discovery.

To overcome the problem of cyclic pattern discovery as emerged in [3], the periodic pattern

discovery was proposed by [4]. In this technique, segment-wise periodicity is defined with a fixed length period. The periodicity-based association rules extract good results for some time periods but not for all. This study reveals that the data cube provides an efficient structure and a convenient way for interactive mining of multiple-level periodicity. It is important to extend the method for mining segment-wise periodicities for an arbitrary length period.

For temporal association rules extraction, [5] proposes a new technique. This technique stated that every item set or rule has a specific lifespan. This lifespan is defined in the database. This technique extracts the association rules from a specific time period less than the database time period. The lifetime of each item was used to define time intervals. The concept of temporal support was introduced for the first time and the apriori algorithm was modified to incorporate time.

A new technique Progressive Partition Miner proposed by [6], explores a new problem of mining general temporal association rules in publication databases. In this work author tried to focus on two problems: 1) Lack of exhibition period for each item and 2) Lack of equitable support for each item. This technique initially portioned the publication database in light of exhibition periods of items and then progressively calculate the frequency count of each candidate-2-itemset based on intrinsic partitioning characteristics. The exhibition of the period is the same as the life span of an item presented in [5]. This algorithm is also designed to employ a filtering threshold in each partition to early prune out those cumulative infrequent candidate 2-itemsets. The feature that the number of candidates 2-itemsets generated by the PPM algorithm is very close to the number of frequent 2-itemsets allows applying the scan reduction technique to effectively reduce the number of database scans. In [7] the author introduced a weighted model of transaction-weighted association rules in a time-variant database. He proposed an efficient progressive weighted miner (PWM) algorithm to perform the mining. In this algorithm, the importance of each transaction period is first reflected by a proper weight assigned by the user. Then, PWM partitioned the time-variant database in light of weighted periods of transactions and performed weighted mining. The algorithm PWM is designed to progressively accumulate the itemset counts based on the intrinsic partitioning characteristics and employ a filtering threshold in each partition to early prune out those cumulatively infrequent 2-itemsets. With this design, the algorithm PWM is able to efficiently produce weighted association rules for applications where different time periods are assigned with different weights and lead to results of more interest. Using the same concept proposed in [5] and [6], the algorithm segmented-progressive-filter (SPF) was introduced in [8], to first segment the database into sub-databases in such a way that items in each sub-database will have either a common starting time or common ending time. Then, for each sub-database SPF progressively filters candidate 2-itemsets with cumulative filtering thresholds either forward or backward in time. This feature allows SPF of adopting the scan reduction technique by generating all candidate k-itemset(k>2) from candidate 2-itemset directly. Junheng-Haung [9] proposed a Segment Progressive Filter (SPFA) algorithm by extending the SPF algorithm proposed in [8]. SPFA significantly outperforms other schemes which are extended from prior methods in terms of execution time and scalability. The advantage of SPFA becomes even more prominent as the size of the database increases. Huang et al.[10], explore the previous studies [8], [9] to remedy the drawbacks of their algorithms. Algorithm TWAIN – two end association miner was presented to find association rules that are absent when the whole range of the database is evaluated altogether. This algorithm not only generates frequent patterns with more precise frequent exhibition periods but also discovers some interesting frequent patterns. TWAIN employees start time and end time of each item to provide precise frequent exhibition periods while progressively handling itemsets from one partition to another. Along with one scan of the database, TWAIN can generate frequent 2-itemsets directly according to the cumulative filtering threshold. Then TWAIN adopts the scan reduction technique to generate all frequent k-itemset (k>2) from the generated frequent 2-itemsets. Experimental results show that this algorithm performs better than the previous algorithm in the quality of frequent patterns execution time, I/o cost, CPU overhead, and scalability.

A Genetic algorithm is proposed by [11], to find temporal association rules. This algorithm simultaneously searches the next rule in rule space and temporal space. This technique discovers more frequent item sets over a short time interval of the transaction dataset. This approach does not require prior partitioning. In [12], the authors investigate the problem of maximum frequent time-window selection (MFTWS) that appears in the process of discovering association rules time-windows (ARTW). They formulate the problem as a mathematical model using integer programming which is a typical combination problem with a solution space exponentially related to the problem size.

A variable neighbourhood search (VNS) algorithm is developed to solve the problem with near-optimal solutions. Computational experiments are performed to test the VNS algorithm against a benchmark problem set. The results show that the VNS algorithm is an effective approach for solving the MTFWS problem, capable of discovering many large frequent-1 itemsets with time-windows (FITW) with a larger time-coverage rate than the lower bounds, thus laying a good foundation for mining ARTW. In [13], the author proposed a new form of an association rule, i.e., an 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. The main contribution of this work includes three aspects. First, the part-time association rule, the new concept of association rule with time windows (ARTM) together with the new concept of frequent itemsets with time windows (FITW), is presented as a more general form of association rule to substitute the old term temporal association rule. Second, a new framework for mining ARTMs on real-time transaction databases is proposed. The new framework is an extension of the support confidence framework, where a new criterion i.e. minimum time window(minwin) is added to judge an ARTW holding or not. Third, an efficient algorithm of TW-Apriori is developed for efficiently generating FITWs from real-time transaction database. They further optimized the process of finding time windows by mathematical modelling [12].

In [14], the author presented a novel technique to identify Calendar-based (Annual, Monthly & Daily) periodicities of and interval-based temporal patterns. An interval-based temporal pattern is a pattern that occurs across a time interval, then disappears for some time, again reoccurs across another time interval, and so on and so forth. Given the sequence of time-interval in which an interval-based temporal pattern has occurred, they proposed a method for identifying the extent to which the pattern is periodic with respect to a calendar cycle. For this, a function called occurrence function is defined for a time stamp. A generalized algorithm was developed for computing the occurrence function at any time stamp in either a discrete or continuous domain. They have also developed another algorithm for finding the local maxima of the occurrence function. It had shown in this paper that, how these two algorithms could be used to determine calendar-based periodicities of an interval-based temporal pattern in either discrete or continuous patterns. The work proposed in [15] applies knowledge discovery techniques on a series of huge datasets obtained over a partition that contains many transactions in a consecutive time period, instead of applying them to the whole database. In addition, instead of extracting rules throughout the whole timeline, the rules are extracted from consecutive time intervals with different time granularities. The result for that will be developing a more efficient approach for mining temporal association rules on large data sets. The main contribution of this paper is to modify the method of partitioning the database by selecting the biggest partition that contains a large number of transactions instead of periodically time partitioning, in order to avoid multi-scanning of the database. In addition to it, the proposed system applies the Apriori Algorithm only to the selected partition aimed to reduce the time of extraction of association rules from the dataset.

In the research paper [16] Saleh and Masseglia deal with the problem of the time periods which

may contain frequent items might be because of the arbitrary division of the data. They introduced the definition of solid itemsets, which represent coherent and compact behaviour over specific periods, and the concept of SIM- Solid Itemset Mining was proposed to find the subsets of the database that contains frequent itemsets. Mazaher Ghorbani & Masoud Abessi, in [17], extended the concept of [16] and proposed a new technique to mine frequent itemsets over temporal data. In this paper, the author proposed an efficient algorithm to mine frequent patterns and their related time intervals from the transactional database. He first presents a new concept of time cubes (TC), to consider time hierarchies in the mining process. Then a new algorithm is proposed based on two thresholds, support and density as a novel threshold. Frequent itemsets are discovered and those with neighbouring time intervals of the same frequent itemsets are merged. This technique assumes that patterns are either hold in either some or all-time intervals. This technique proposes a time cube analysis of frequent patterns. The whole dataset is divided into the number of time cubes such as (hour, day, month), (day, month, year), etc. and analysis is done using an Apriori algorithm over these time cube data. It uses the temporal support value concept. In [18] the author, make an effort to enhance conventional rule mining by introducing temporal soft sets. They define temporal granulation mappings to induce granular structures for temporal transaction data. Using this notion, they define temporal soft sets and their Q-clip soft sets to establish a novel framework for mining temporal association rules. A number of useful characterizations and results are obtained, including a necessary and sufficient condition for the fast identification of strong temporal association rules. By combining temporal soft sets with NegNodeset-based frequent item set mining techniques, they develop the negFIN-based soft temporal association rule mining (negFIN-STARM) method to extract strong temporal association rules.

In [19] the authors present an integrated approach that can be used to write efficient codes for pattern mining problems. The approach includes, (1) cleaning datasets with the removal of infrequent events, (2) presenting a new scheme for time-series data storage,

(3) exploiting the presence of prior information about a dataset when available, (4) utilizing vectorization and multicore parallelization. They present two new algorithms, FARPAM (FAst Robust PAttern Mining) and FARPAMp (FARPAM with prior information about prior uncertainty, allowing faster searching). The algorithms are applicable to a wide range of temporal datasets. They implement a new formulation of the pattern searching function which reproduces and extends existing algorithms (such as SPAM and RobustSPAM), and allows for significantly faster calculation. The algorithms also include an option of temporal restrictions in patterns, which is available neither in SPAM nor in RobustSPAM. The search algorithm is designed to be flexible for further possible extensions.

From the above discussion, it is observed that the time intervals have to be specified by the user. In this paper our proposal is completely different from the previous methods for finding temporal patterns, here we consider time as a hierarchical data structure and then extract the local periodic patterns from these hierarchical levels along with the time intervals without specifying the time intervals in advance by the user. Our method will extract the time time-intervals automatically while scanning the dataset in which the patterns are frequent.

III. DEFINTION & TERRMINOLOGY

Let I = $\{i_1, i_2, . . . i_n\}$ be a set of items and D be a database of transactions. Each transaction tr is associated with an identifier transaction ID (TID), a time stamp TS and a set of items. For all transactions, $T_{tr} \in$ TS, where TS is the total time span of the database. Let t_{st}, $t_{et} \in$ TS where t_{st} is the start time and t_{et} is the end time be the time interval in each time hierarchy, and also it is clear that $t_{st} < t_{et}$. For example, (2,6) Month shows the time interval between the 2nd and 6th months.

We define the local support of an itemset in a time interval [t1,t2] as the ratio of the number of transactions in the time interval containing the itemset to the total number of transactions in that time-interval for the whole dataset D. We use the notation $Supp_{[t1,t2]}(X)$ to denote the support of item set X in that time interval. Here we define the Time Interval as BTC-Basic timecube. The Support of an itemset X in BTC is calculated as:

$$Support_{BTC}(X) = N(X)^{Cube} / |N^{Cube}| . \quad (1)$$

Given a Threshold (σ) we say that an itemset X is frequent in the BTC if :

$$SUP_{BTC}(X) >= (\sigma/100)*TC \quad (2)$$

Where TC denotes the total number of transactions in D that are in the BTC.

We say that the set is locally frequent in BTC. We say that an association rule X=>Y, where X & Y are item sets, holds in the BTC if and only if given threshold 9,

$$SUP_{btc}(XUY) / SUP_{BTC} (X) >= \sigma/100 \quad (3)$$

and XUY is frequent in BTC. In this case, we say that the confidence of the rule is Conf.

For each locally frequent itemsets extracted by the algorithm, a list of Time intervals are maintained in TC-Time Cube. Where each interval is represented as [str-dt, end-dt], where str-dt is the starting Timestamp of the Time Interval and end-dt is the ending timestamp of the time interval. (end-dt – std-dt) gives the length of the time interval (TP). Given two intervals [std-dt1,end-dt1] & [std-dt2,end-dt2], if the intervals are non-overlapping and std-dt2>end-dt1, then std-dt2-end-dt1 gives the distance between the time-interval.

Minimum support is a threshold to evaluate itemsets. Since records are not equally distributed in time intervals, very few records may occur on some occasions. Therefore, generated patterns may not be valid, since there is not enough evidence to show that they exist for that time interval, which causes the overestimating problem. In order to overcome this overestimating problem, we have used another threshold which is called density.

Let us explain the necessity of density with an example. Consider itemset XY with $Sup(X, Y)^{4-9} = 8 /12 \geq 60\%$ Where 60% is the minimum support. However, we can observe from the database that XY is only frequent during the period 4 to 7 rather than the whole range of 4 to 9, which is an overestimated time period. Density not only ensures the validity of the patterns but also filters out time intervals with few records which cause overestimating the time periods. The density of a time interval is calculated as follows:

$$Avg = N /N_{BTCs} \quad (4)$$

$$Density = \alpha \times Avg. \quad (5)$$

where N is the total number of records or transactions, N_{BTCs} is the number of basic time cubes, and therefore, Avg. is the average number of transactions per BTC. A user-specified parameter $\alpha \in [0, 1]$ is used to determine the desired density using equation 4. α is called density rate. For example, if a dataset contains 2000 records and 10 BTCs, then the average number of records per basic time-cube is 200. With parameter $\alpha = 0.5$, cubes less than 100 records are filtered.

Therefore, an itemset is called frequent if and only if for each Time-Cube(TC), it satisfies the following conditions.

1) $X \overset{Cube}{=\Rightarrow} Y$ has a support greater than or equal to the minimum support threshold which is defined by the user.

2) The time interval (Time cube) must be dense to ensure the validity of the rules.

3.1 Set Superimposition:

In this work, we have used an operator called superimposition denoted as (S), which was proposed by (Baruah, 1999). If set A is superimposed over the set B or B is superimposed over A then, we have:

$$A(S)B = (A-B)\ (+)(A \cap B)^2(+)(B-A) \qquad (6)$$

Where $(A \cap B)^2$ are the elements of $(A \cap B)$ represented twice, and (+) represents the UNION of disjoint sets. To explain this, let us take an example below:

If A=[a1,b1] and B=[a2,b2] are two time intervals such that A intersect B != {}, we could get a superimposed portion, such as:

$$[a1,b1](S)[a2,b2] =$$

$$[(a_{(1)},a_{(2)})\ (+)\ [a_{(2)},b_{(2)}]^2(+)(b_{(1)},b_{(2)}] \qquad (7)$$

Where

$a_{(1)} = \min(a1,a2) \quad a_{(2)} = \max(a1,a2)$

$b_{(1)} = \min(b1,b2) \quad b_{(2)} = \max(b1,b2)$

IV. PROPOSED ALGORITHMS

At the time of constructing locally frequent sets, with each locally frequent set, a list of time intervals (TC) is constructed in which the itemset is frequent. Here we use two thresholds mintp1 and mintp2 as user input values. During execution, while making a pass through a dataset, if for a particular itemset the time gap between its current timestamp and the time when it was last seen(before the current timestamp) is less than the value of mintp1 then the current transaction is included in the current time interval(BTC) under consideration, otherwise, a new time-interval is started with the current timestamp as the starting point. The support count of the itemset in the previous time interval is checked to see whether it is frequent in that interval or not and if it is then it is added to the list (TC) maintained for that set. Also for the locally frequent itemsets, a minimum period length(mintp2) given by the user and minden(Minimum Density) is checked against the values of the time-interval (BTC). If the time-gap (TP) of the BTC and the number of transactions of the BTC is greater than the mintp2 and minden then these values are considered. If mintp2 is not used then an item appearing only once in the whole dataset will also become locally frequent.

4.1. Procedure for Finding Locally Frequent Itemsets:

4.1.1. Pre-processing the Dataset:

If the dataset is non-temporal then we incorporate the temporal features i.e. Timestamps in the Dataset. For this, we have developed a program, which generates a list of timestamps when an initial timestamp value is given as input. It generates the timestamps from the initial timestamp value feed by the user to the end as per the number of records in the Dataset which is calculated by the program based on the input dataset. Program after merging these generated timestamps with the records of the dataset a new datafile is created which is used by the algorithm to generate the frequent itemset.

4.1.2. Compute L1- the set of Locally Frequent-1 Itemsets:

For each item, while scanning the dataset we record a timestamp called L-timestamp that corresponds to the time when the item was last scanned. When an item is found in a transaction and the timestamp is tmstp and the time-gap between L-timestamp and the tmstp is greater than the minimum threshold(mintp1) given, then a new time interval (BTC) is started by setting starting timestamp of the new time interval (BTC) as tmstp and ending timestamp of the previous BTC as L-timestamp. The previous BTC is added to the list (TC) maintained for that itemset provided that the duration of the BTC and the support of the itemset in BTC are both greater than or equal to the minimum thresholds specified for each. Otherwise, L-timestamp is set to tmstp, the counter maintained for counting transactions are increased appropriately and the process is continued. The Algorithm to compute L1-list of frequent-1 sets is given below.

Algorithm-1:

```
def : scan(D,C1):
Input:
   D= Temporal Datafile , C1=Candidate-1 Itemset,
mintp1, mintp2
Output:
            L1=Set of Frequent-1 Itemset with Time
Intervals
Method:
 n=len(C11)  # find the length of the C1
   for k in range(n):
       icount=0
       ptcount=0
       trcount=0
       L-timestamp=0
       F-timestamp=0
       for tr in range(len(D)):
           if C1[k].issubset(D[tr]):
               tstmp=tm[tr]
               if L-timestamp = 0:
                  F-timstamp=tstmp
                  L-timestamp=tstmp
                  icount=1
                  ptcount=1
                  trcount=1
                  elif (L-timestamp !=0):
       start=datetime.strptime(lastseen, '%d-%m-%Y')
       end=datetime.strptime(tstmp, '%d-%m-%Y')
                   nodays=(end-start).days
                   if(nodays<=mintp1):
                       L-timestamp= tstmp
                       icount=icount+1
                       trcount=trcount+1
                       ptcount=trcount
                   elif (nodays>=mintp2):
                       tc.append(C1[k])
                       tc.append(F-timestamp)
                       tc.append(L-timestamp)
                       tc.append(icount)
                       sup=round(icount/ptcount,2)
                       tc.append(sup)
                       firstseen=tstmp
                       lastseen=tstmp
                       icount=1
                       ptcount=1
                       trcount=1
           else:
               trcount=trcount+1
   n=5
   tc2=[]
   tc2=[tc[i:i+n] for i in range(0,len(tc),n)
#Return the L1-Frequent-1 itemset with time-
intervals
   return(tc2)
```

In the above algorithm three support counts icount, ptcount, trcount are maintained with each item. When an item is first scanned then these are initialized to 1. For each item while making a scan through the dataset when a transaction containing the item is found then icount for that item is increased. To check whether an item is frequent in a time interval the total number of transactions in that time interval will have to be counted. For this, with each item, two counts ptcount and trcount are kept. The value of trcount increases with each transaction, but ptcount changes its value only when a transaction containing an item is found within mintp1 from the current value of the L-timestamp and then it takes the value of ctcount. When an item is not seen for more than mintp, the time distance from L-timestamp, then the value of ptcount is used to compute the percentage support count of the item between F-timestamp and L-timestamp. If the count percentage of an item in a time interval is greater than the minimum support and time period, then only the itemset is considered as a locally frequent set and the locality is the time interval. When a new time interval is started for an item then these three count values are again initialized to 1.

4.1.3. Compute Lk – Frequent -k Itemsets with Time Intervals

After this Apriori candidate generation algorithm is used to find candidate frequent-2 itemset and then the pruning is applied. With each frequent-2 itemset, the list of time intervals is associated. In the candidate generation phase, the list of time intervals (TC) is empty. During the pruning phase, this list is prepared. The procedure for preparing is that when the first subset of itemset appearing in the previous level is found then that list is taken as the list of time intervals associated with the set. When the subsequent sets are found then the list is reconstructed by taking all possible pair-wise intersections of subsets one from each list. If this list becomes empty at any point in time or when a particular subset of the itemset under consideration is not found in the previous level then the set is pruned. The pairwise intersection of the interval list is taken for the following reasons. If the interval say[t1,t1'] and [t2,t2'] in which the itemsets say {A,B} is frequent then there exists two time periods [t1,t1'] and [t2,t2'] in which the itemsets {A} & {B} are respectively, frequent and [t,t'] subset [t1,t1] intersection [t2,t2']. Using this concept we described below the modified Apriori algorithm for the problem under consideration.

```
Algorithm2:
Modified Apriori
Initialize k=1
C1=Candidate-1 Itemset
L1=Frequent-1 Itemset with Time intervals (TC)
          #L1 is computed from Algorithm1
K=2
While(L(k-1) != {}) :
          Ck = Cand-Gen(L[k-1],k)
#Candidate generation by setting the TC =0
          Prune(Ck)
#drop all lists of time-interrvals(TC) maintained with the sets in Ck(Candidate itemset) which does not satisfy the threshold conditions.
freq-Gen(Ck,minsup,dbslen)

Compute Lk from Ck
#Lk can be computed from Ck using Algorithm1
K=k+1
Lk = Lk-1 U Lk-2 U….U Lk-i        (where i=1 to k)
Return (Lk)
# Set of Frequent-k itemsets
Algorithm3:

def freq-Gen(Ck,minsup,dbslen):
l = Ck
nopart=int(len(l))
if nopart != 0:
      a=dbslen/nopart
      den=a
 else:
     print("No. of Partitions are zero")
     print("Frequent Set cannot be generated")

   for i in range(len(l)):
      l2=l[i]
      st=datetime.strptime(l2[1], '%d-%m-%Y')
      et=datetime.strptime(l2[2], '%d-%m-%Y')
      tp=(et-st).days
      if (l2[4]>=minsup and tp>=3):
         l3.append(l2)
   return(l3)
Algorithm4:
def Cand-Gen(Lk, k):                    #creates Ck
lenLk = len(Lk)
for i in range(len(Lk)):
for j in range(i+1, lenLk):
   L1 = list(Lk[i])[:k-2]; L2 = list(Lk[j])[:k-2]
   L1.sort(); L2.sort()
   if L1==L2:             #if first k-2 elements are equal
   retList.append(Lk[i] | Lk[j])          #set union
  return (retList)
Algorithm5:
def prune(Ck,l5)
 for i in range(len(Ck)):
   temp1=Ck[i]
   for j in range(len(l5)):
     if l5[j][0].issubset(temp1):
        temp2=l5[j][1]
        temp3=l5[j][2]
          for k in range(j+1,len(l5)):
            if l5[k][0].issubset(temp1):
               temp4=l5[k][1]
               temp5=l5[k][2]
          dt1=datetime.strptime(temp2, '%d-%m-%Y')
           dt2=datetime.strptime(temp4, '%d-%m-%Y')
               gap1=abs((dt2-dt1).days)
          dt3=datetime.strptime(temp3, '%d-%m-%Y')
           dt4=datetime.strptime(temp5, '%d-%m-%Y')
               gap2=abs((dt4-dt3).days)
      if l5[k][0].issubset(temp1) and gap1<=2
             and gap2<=2:
d1=pairwise-intersection(temp2,temp3,temp4,temp5)
         d2.append(temp1)
         d2.append(d1)
         tp.append(d1)
   return(d2)
```

4.1.4. Explanation of the Algorithm with an Example

To explain the working of the algorithm, we have taken a small dataset consisting of timestamps and the list of items at the corresponding dates. We have assumed the dataset is collected for a period of two months i.e. 01-01-2021 to 28-02-2021 and the market is open all days of the week. The total number of items are 5, total number of transactions are 59. We have assumed the mintp1=2 days and the mintp2=5 days and min-support=50%. We executed the algorithm to find the locally frequent itemset. The dataset statistics are given below:

Database length= 182 Transactions
Total Number of frequent itemsets = 31
The Execution Time is : 0.859375 msec

Read the dataset to find the list of time intervals where the itemsets are frequent. Read the transactions one by one to find the F-timestamp, L-timestamp, and the support of the itemset in [F-timestamp, L-timestamp]. After the first scan of the dataset, we get the set local frequent-1 itemset along with the time interval as follows:

```
--------------------------------------------------------------|
|         The frequent one itemset is                        |
|-------------------------------------------------------------|
|Item   Starting Date   Ending Date  Item Count   Item
Support |
|-------------------------------------------------------------|
[{'curd'}, '30-01-2022', '02-02-2022', 4, 1.0]
[{'curd'}, '13-02-2022', '19-02-2022', 5, 0.71]
[{'curd'}, '19-03-2022', '25-03-2022', 5, 0.71]
[{'curd'}, '07-04-2022', '11-04-2022', 4, 0.8]
[{'curd'}, '17-04-2022', '20-04-2022', 3, 0.75]
[{'detergent'}, '30-01-2022', '02-02-2022', 4, 1.0]
[{'detergent'}, '11-02-2022', '19-02-2022', 5, 0.56]
[{'detergent'}, '19-03-2022', '25-03-2022', 4, 0.57]
[{'detergent'}, '07-04-2022', '11-04-2022', 3, 0.6]
[{'salt'}, '30-01-2022', '05-02-2022', 6, 0.86]
[{'salt'}, '11-02-2022', '19-02-2022', 6, 0.67]
[{'salt'}, '19-03-2022', '25-03-2022', 5, 0.71]
[{'salt'}, '07-04-2022', '11-04-2022', 4, 0.8]
[{'salt'}, '17-04-2022', '20-04-2022', 3, 0.75]
List L is  [[{'curd'}, {'detergent'}, {'salt'}]]
```

In the second level we get the candidate itemsets as follows:

```
--------------------------------------------------------------|
|         The Frequent-TWO itemset is                          |
|-------------------------------------------------------------|
|Item   Starting Date   Ending Date  Item Count   Item
Support |
|-------------------------------------------------------------|
[{'curd', 'detergent'}, '30-01-2022', '02-02-2022', 4,
1.0]
[{'curd', 'detergent'}, '13-02-2022', '19-02-2022', 4,
0.57]
[{'curd', 'detergent'}, '19-03-2022', '25-03-2022', 4,
0.57]
[{'curd', 'detergent'}, '07-04-2022', '11-04-2022', 3,
0.6]
[{'curd', 'salt'}, '30-01-2022', '02-02-2022', 4, 1.0]
[{'curd', 'salt'}, '13-02-2022', '19-02-2022', 5, 0.71]
[{'curd', 'salt'}, '19-03-2022', '25-03-2022', 5, 0.71]
[{'curd', 'salt'}, '07-04-2022', '11-04-2022', 4, 0.8]
[{'curd', 'salt'}, '17-04-2022', '20-04-2022', 3, 0.75]
[{'salt', 'detergent'}, '30-01-2022', '02-02-2022', 4,
1.0]
[{'salt', 'detergent'}, '11-02-2022', '19-02-2022', 5,
0.56]
[{'salt', 'detergent'}, '19-03-2022', '25-03-2022', 4,
0.57]
[{'salt', 'detergent'}, '07-04-2022', '11-04-2022', 3,
0.6]
```

In the third level we get the frequent-3 itemset as:

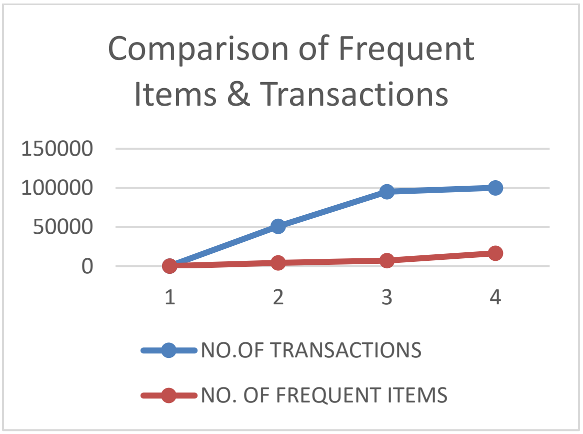
```
--------------------------------------------------------------|
|         The Frequent-THREE itemset is
|
|-------------------------------------------------------------|
|Item   Starting Date   Ending Date  Item Count   Item
Support |
|-------------------------------------------------------------|
[{'curd', 'salt', 'detergent'}, '30-01-2022', '02-02-2022',
4, 1.0]
[{'curd', 'salt', 'detergent'}, '13-02-2022', '19-02-2022',
4, 0.57]
[{'curd', 'salt', 'detergent'}, '19-03-2022', '25-03-2022',
4, 0.57]
[{'curd', 'salt', 'detergent'}, '07-04-2022', '11-04-2022',
3, 0.6]
```

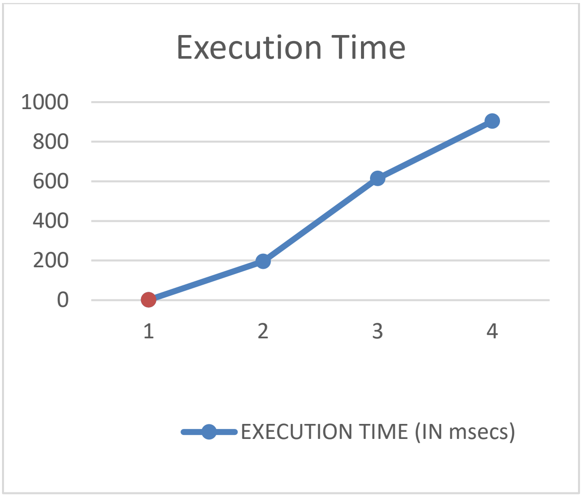
V. IMPLEMENTATION & RESULTS

The above algorithm we have implemented in python 3.10.2 on Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz on Windows 11 Home Single Language. We have taken the dataset from http://fimi.uantwerpen.be/data/ -FIMI – **Frequent Itemset Mining Dataset Repository.** We conducted the experiments on different datasets. The results and the details of the datasets are as follows:

Table1- Executed data Statistics

SNO	DATASET-NAME	NO.OF TRANSACTIONS	NO. OF FREQUENT ITEMS	EXECUTION TIME (IN msecs)
1	GROCERY.TXT	159	30	1.09375
2	KOSARK.TXT	50877	4174	194.90625
3	RETAIL.TXT	95050	7114	614.09375
4	T10I4D100K	100059	16375	903.46875





From the above Figures, it can be seen that in Figure 1, as the No. of Transactions increases the No. of Frequent items also increased. Similarly in Figure 2, it can be observed that a more number of transactions takes more amount of Execution time.

VI. CONCULSION

In this paper, we mainly focused on extracting local frequent itemsets along with Time Intervals. Here we have used a time hierarchy structure Time Cube to store the time intervals.

Our Algorithm is successfully used a set operator called Set Superimposition. This Algorithm is able to partition and extract the Time intervals automatically from the dataset, as it is not needed to specify the dates to partition the dataset by the user which was required in the previous studies. As can be seen that this Algorithm generates more frequent items and takes more amount of time for Execution as the size of the data increases.

In the future, we try to overcome the pitfalls of this algorithm by:

- Reducing the amount of Execution Time
- Generate interesting Frequent Itemset and Association Rules.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," ACM SIGMOD Rec., vol. 22, no. 2, pp. 207–216, 1993.

[2] R.Agarwal & R.Srikanth, Mining Sequential Patterns, in Proc. IEEE 11th Intl. Conf. Data Eng. 1995

[3] B.Ozden, S. Ramaswamy, and A. Silberschatz, "Cyclic Association Rules," in Proc. IEEE 14th Int. Conf. Data Eng., 1998, pp. 412–421.

[4] S. Ramaswamy, S. Mahajan, and A. Silberschatz, "On the discovery of interesting patterns in association rules," in Proc. 24th Int. Conf. Very Large Data Bases, 1998, pp. 368–379.

[5] J. Han, W. Gong, and Y. Yin, "Mining segment-wise periodic patterns in time-related databases," in Proc. Int. Conf. Knowl. Discovery Data Mining, 1998, pp. 214–218.

[6] J. M. Ale and G. H. Rossi, "An approach to discovering temporal association rules," in Proc. ACM Symp. Appl. Comp.-vol. 1, 2000, pp. 294–300.

[7] C.-H. Lee, M.-S. Chen, and C.-R. Lin, "Progressive Partition Miner: An efficient algorithm for mining general temporal association rules," IEEE Trans. Knowl. Data Eng., vol. 15, no. 4, pp. 1004–1017, Jul./Aug. 2003.

[8] C-H. Lee, J.C. Ou, & M.S. Chen, "Progressive Weighted Miner: An efficient method for time-constraint mining", in Advances in knowledge Discovery and Data Mining. NewYork. Ny. USA: Springer,2003 pp 449-460.

[9] C.Y. Chang, M.S. Chen, and C.H. Lee, "Mining general temporal association rules for items with different exhibition periods," in Proc. IEEE Int'l. Conf. Data Mining, 2002, pp.59–66.

[10] W.-W. Junheng-Huang, "Efficient algorithm for mining temporal association rule," Int. J. Comput. Sci. Netw. Sec., vol. 7, no. 4, pp. 268–271, 2007.

[11] J.W. Huang, B.R. Dai, and M.S. Chen, "Twain: Two-end association miner with precise frequent exhibition periods," ACM Trans. Knowl. Discovery Data, vol. 1, no. 2, 2007, Art. no. 8.

[12] S. G. Matthews, M. A. Gongora, and A. A. Hopgood, "Evolving temporal association rules with genetic algorithms," in Research and Development in Intelligent Systems XXVII. New York, NY, USA: Springer, 2011, pp. 107–120

[13] Y. Xiao, Y. Tian, and Q. Zhao, "Optimizing frequent time-window selection for association rules mining in a temporal database using a variable neighbourhood search," Comput. Oper. Res., vol. 52, pp. 241–250, 2014.

[14] Y. Xiao, R. Zhang, and I. Kaku, “A new framework of mining association rules with timewindows on real-time transaction database,” Int. J. Innov. Comput., Inf. Control, vol. 7, no. 6, pp. 3239–3253, 2011.

[15] Mulla Dutta & Anjana Kakoti Mahanta, “Detection of Calendar-based periodicities of Interval based Temporal Patterns”, International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.2, No.1, January 2012.

[16] Abdel Rahman Mahmoud, Dr. Nagy Ramadan, and Abdel Moniem Helmy, “An Enhanced Algorithm for Association Rule Mining in Huge Temporal Database,” International Research Journal of Advanced Engineering and Science, Volume 4, Issue 3, pp. 254-261, 2019.

[17] B. Saleh and F. Masseglia, “Discovering frequent behaviors: Time is an essential element of the context,” Knowl. Inf. Syst., vol. 28, no. 2, pp. 311–331, 2011.

[18] Mazaher Ghorbani and Masoud Abessi, "A New Methodology for Mining Frequent Itemsets on Temporal Data", in IEEE Transactions on Engineering Management, Vol. 64, Issue. 4, pp. 566 - 573, Nov 2017

[19] Mining Temporal Association Rules with Temporal Soft Sets Xiaoyan Liu, 1 Feng Feng, Qian Wang, Ronald R. Yager, Hamido Fujita, and Jose´ Carlos R. Alcantud, Hindawi Journal of Mathematics Volume 2021, Article ID 7303720, 17 pages

[20] Fast implementation of pattern mining algorithms with time stamp uncertainties and temporal constraints, Sofya S. Titarenko, Valeriy N. Titarenko, Georgios Aivaliotis and Jan Palczewski, Titarenko et al. J Big Data (2019) 6:37, Springer. https://doi.org/10.1186/s40537-019-0200-9

[21] Anjana Kakoti Mahanta, Fokrul Alom Mazarbhuiya, Hemath K. Barauh, Finding Calendar-based Periodic Patterns, 0167-8655/$ Elsevier.B.V. 2008.01.020.