Mining Periodic Patterns from Non-binary Transactions

Jhimli Adhikari
Department of Computer Science
Narayan Zantye College, Bicholim
Goa - 403 529, India
jhimli_adhikari@yahoo.co.in

Abstract: *Pattern with time period is more valuable because it can better describe objective knowledge. Previous studies on periodic patterns from market basket data focus on patterns without considering the items with their purchased quantities. But in real-life transactions, an item could be purchased multiple times in a transaction and different items may have different quantity in the transactions. To solve this problem, we incorporate the concept of transaction frequency (TF) and database frequency (DF) of an item in a time interval. Our algorithm works in two phases. In first phase we mined locally frequent item sets along with the set of intervals and their database frequency range and second phase mines the two types of periodic patterns (cyclic and acyclic) from the list of intervals. Experimental results are provided to validate the study.*

Keywords: Data Mining, Locally Frequent Items, Non-Binary Transaction, Periodic Pattern, Temporal Interval

Received: 12 May 2018, Revised 17 June 2018, Accepted 29 June 2018

DOI: 10.6025/jic/2018/9/4/144-156

1. Introduction

Pattern recognition and interestingness measures are two important and as well as interesting topics at the heart of various types of data mining problems. Temporal data mining is concerned with the analyses of data to find out patterns and regularities from a set of temporal data. In this context, association rule mining considering local frequent patterns (Yin, et al. 2014) cyclic patterns (Hu & Chiang, 2011) asynchronous periodic patterns (Liu, et al. 2017) and calendar-based periodic pattern (Adhikari, 2014; Adhikari & Rao 2013) are some interesting temporal patterns reported in recent times. For the effective management of business activities, we often wish to discover knowledge from time-stamped data. The supermarket transactional data is an example of temporal data. Periodicity detection is a process for finding temporal regularities within the time-stamped database.

Periodicity search, that is, search for cyclic patterns in time-related datasets, is an important data mining problems with many applications. Periodic pattern analysis not only helps in understanding the behaviour of the data but also contributes in predicting the future trends of the data. Problems related to periodicity search are stated as problems of finding patterns' occurrence at regular intervals. Literally, the concept emphasizes on two aspects of the problem, namely pattern and interval. Each cyclic pattern

is associated with temporal information pertaining to its durations of periodic appearances in a database. Little work has been reported on the area of mining periodic patterns from non-binary transactions. In this article we not only mine periodic patterns but also quantity from the database.

2. Motivation

Previous studies (Mazarbhuiya et al. 2012; 2016) on mining periodic patterns from transaction data focus on discovering patterns from binary transactions without considering the items with their purchased quantities. Authors studied the problem of cyclic and acyclic nature from a list of time intervals associated with a locally frequent itemset and devised a method to extract all frequent itemsets which are periodic in nature. Therefore, the patterns extracted from the databases show only periodicity but do not provide any information about the quantity. However, most of the real-life databases are non-binary, in sense that an item could be purchased multiple times in a transaction. Thus, periodic patterns in a binary database might have limited usage. When an item is purchased multiple times in a transaction then the existing techniques proposed by Mazarbhuiya et al. (2012; 2016) might not be adequate for mining periodic patterns, since they are based on a binary database. Thus, exploring periodic patterns with their purchased quantities along with the time periods may discover information useful to improve the quality of business decisions. It could help companies to project the right products on the right time. These facts and the limitations of existing algorithms motivated us in developing a method of mining such interesting periodic patterns from non-binary transactional data. We divide these studies into following two phases:

i) Finding locally frequent itemsets along with list of intervals and quantity,

ii) Mining periodic patterns from the list of intervals.

Rest of the paper is organized as follows. We discuss related work in Section 3. In Sections 4 to 9 we introduce associated concepts of periodic patterns and non-binary transactions. We state our problem and periodic patterns in Sections 10 and 11 respectively. In Section 12, we design algorithms for mining periodic patterns from temporal database. Experimental results are provided in Section 13.

3. Related works

Finding periodic patterns has been widely investigated in various domains as temporal patterns. Recently researchers introduced different types of periodic patterns such as quasi periodic frequent pattern (Kiran & Kitsuregawa, 2013) and acyclic patterns (Mazarbhuiya, 2016). But little work has been reported on mining periodic pattern from non-binary transactional data.

A frequent pattern is said to be *quasi-periodic-frequent* if most of its occurrences are periodic in a database. Here authors relaxed the constraint that a pattern must appear periodically throughout the database, and introduced a class of user-interest-based frequent patterns. In this paper we design an algorithm which mines cyclic as well as acyclic patterns from non-binary transactions.

Adhikari (2014) proposed an algorithm to mine calendar-based periodic patterns from non-binary transactions. The algorithm mined frequent itemsets along with quantity and list of intervals. Then periodicity (weekly, monthly, yearly) of the patterns was checked. In this article we extend the previous work and mine cyclic as well as acyclic periodic patterns along with quantity from non-binary transactions.

Barreto & Antunes (2014) mined periodic regularities from temporal data. In this connection authors proposed a constraint-based sequential mining method, called ConstraintPrefixSpan, for mining three types of periodic regularities: cyclic, converging and diverging.

Mahanta et al. (2008), Adhikari & Rao (2013) mined calendar-based periodic patterns from binary transactional data. Therefore, the patterns extracted from the databases show only periodicity but do not provide any information about the quantity.

Nishi et al. (2013) proposed an algorithm for the periodic pattern mining in time series databases which does not rely on the user for the period value and can detect all types of periodic patterns. The proposed algorithm facilitates the user to generate

different kinds of patterns by skipping intermediate events in a time series database and find out the periodicity of the patterns within the database. Our work is different, since we handle with transactional time stamped database to extract periodic patterns.

Chanda et al. (2017) proposed a weighted periodic pattern mining algorithm that can mine three types of weighted periodic pattern, (i.e. single, partial, full) in a single run from time series databases. In this article we mine two periodic patterns (cyclic and acyclic) along with quantity in a single run from non-binary transactions.

Kiran et al. (2016) have suggested a greedy search technique to determine the periodic interestingness of a pattern. The technique discovers all periodic-frequent patterns by eliminating aperiodic patterns based on suboptimal solutions.

4. Preliminaries

This section introduces few terminologies required to get a clear perception on non-binary transaction data, locally frequent itemsets and periodic patterns.

Non-binary Transaction Database

Transactional database is a collection of transactions along with the related time stamps. Let *TIMT* be the type of a database such that a Transaction in the database might contain an Item Multiple Times. A transaction in *TIMT*-type database *DB* containing p items could be stored as follows: $\{i_1(n_1), i_2(n_2), \ldots, i_p(n_p)\}$, where item i_k is purchased n_k numbers at a time in the transaction, for $i = 1, 2, \ldots, p$. In this article, a database refers to a *TIMT*-type database if the type of the database is unspecified.

Adhikari and Rao (2008) proposed some measures related to the *TIMT* database. Each itemset X in a transaction is associated with the following two attributes: transaction itemset frequency (*TIF*) and transaction itemset status (*TIS*). These two attributes are defined as follows:

$TIF(X, \tau, DB) = m$, if X occurs m times in transaction τ in DB

$$TIF(X, \tau, DB) = \begin{cases} 1, for\ X \in \tau, \text{ and } \tau \in DB \\ 0, for\ X \notin \tau, \text{ and } \tau \in DB \end{cases}$$

Also, each itemset X in DB is associated with the following two attributes: transaction frequency (*TF*), and database frequency (*DF*). These two attributes are defined as follows:

$$TF(X, D) = \sum_{\tau \in D} TIS(X, \tau, DB)$$

$$DF(X, D) = \sum_{\tau \in D} TIF(X, \tau, DB)$$

Let *DB* be a *TIMT* -type database shown in Example 1, where $x(\eta)$ denotes item x purchased η numbers at a time in the corresponding transaction. Consider the following *TIMT*-type database *DB* in Example 1. We refer this database time to time to explain various concepts.

Example 1. *DB* consists of 25 transactions. *DB* is ordered in the ascending order of the time-stamps and within a transaction the items are alphabetically sorted. *TID* of each transaction represents its sequential occurrence order with respect to a particular timestamp. Each record contains items purchased along with their quantity as well as the date of the transaction and transaction identifier *TID*. Time of a transaction is omitted because our data analysis is not associated with the time component of a transaction.

With respect to above database we elaborate locally frequent itemsets, lifespan of an itemset, transaction frequency (*TF*) and database frequency (*DF*) of an itemset in the following section.

5. Basic Concepts, Definitions and Notations

Locally frequent itemsets are itemsets that are frequent in certain time intervals and may or may not be frequent throughout the

TID	time-stamp	items(quantity)	*TID*	time-stamp	items(quantity)
T1	28/5/2000	{*a*(30),*b*(3), *c*(22),*g*(5)}	T14	2/10/2000	{*c*(7), *d*(8),*f*(3)}
T2	8/6/2000	{*a*(22), *c*(10),*d*(2), *e*(3)}	T15	15/10/2000	{*a*(9), *c*(7),*f*(3)}
T3	15/6/2000	{*c*(18), *d*(2), *f*(5)}	T16	27/10/2000	{*a*(12), *c*(15),*g*(2)}
T4	22/6/2000	{*a*(17),*c*(23), *f*(8)}	T17	3/11/2000	{*a*(5), *c*(9), *d*(7)}
T5	28/6/2000	{*a*(7), *c*(20), *g*(8)}	T18	18/11/2000	{*a*(6), *b*(7), *c*(9)}
T6	15/7/2000	{*e*(9), *g*(7)}	T19	29/11/2000	{*d*(7), *e*(5), *g*(4)}
T7	26/7/2000	{*b*(11), *e*(4),*f*(12)}	T20	12/12/2000	{*b*(5), *d*(4)}
T8	1/8/2000	{*a*{10},*c*(5), *d*(2), *e*(3)}	T21	23/12/2000	{*a*(5),*b*(3),*c*(6),*f*(1)}
T9	17/8/2000	{*a*(11), *c*(11), *d*(4)}	T22	29/12/2000	{*b*(4), *c*(7), *e*(8)}
T10	25/8/2000	{*a*(10), *b*(13), *e*(3)}	T23	6/1/2001	{*a*(7), *c*(5), *e*(4)}
T11	1/9/2000	{*b*(5), *c*(14),*f*(7)}	T24	14/1/2001	{ *a*(2), *c*(1), *d*(7)}
T12	8/9/2000	{*a* (16), *c* (2), *e* (7), *g* (10)}	T25	20/1/2001	{*a*(1), *c*(3), *g*(4)}
T13	20/9/2000	{*b* (5), *c*(7), *e* (12)}			

Table 1. *TIMT* dataset *DB*

lifespan of the itemset. For example, some items may appear in the transaction for a certain period and then disappear for a long period and then appear again. Extracting those itemsets together with the time slots in which they are frequent is important. The lifespan of an itemset is defined as the time period between the first transaction and the last transaction containing the itemset. For example, the lifespan of itemset {*f*} in Table 1 is 15/6/2000 to 23/12/2000. The itemset might not be frequent throughout the lifespan as its sale heavily depends on season and festival. Thus, our objective is to mine all the intervals where the itemsets are frequent. In this context, Adhikari and Rao (2013) designed an algorithm that dynamically extracts all the frequent itemsets along with the period where the itemsets are frequent. We define transaction frequency (*TF*), and database frequency (*DF*) with respect to an interval $[t_1, t_2]$ as follows:

$$TF_{[t1,\,t2]}(X, |DB|) = \sum_{\tau \in D} TIS_{[t1,\,t2]}(X, \tau, |DB|) \quad (1)$$

$$DF_{[t1,\,t2]}(X, |DB|) = \sum_{\tau \in D} TIF_{[t1,\,t2]}(X, \tau, |DB \quad (2)$$

Here $|DB|$ denotes the total number of transactions in time interval $[t_1, t_2]$. Based on the frequency of an itemset in a database, we define transaction support (*tsupp*) and database support (*dsupp*) with respect to an interval $[t_1, t_2]$ of an itemset as follows.

Definition 1. Let *X* be an itemset in *TIMT* type database *DB* and it is frequent in an interval $[t_1, t_2]$. Transaction support of *X* in an interval $[t_1, t_2]$ is computed from Eq. (1) as follows:

$$tsupp_{[t1,\,t2]}(X, DB) = TF_{[t1,\,t2]}(X, DB) / |DB|$$

Definition 2. Let *X* be an itemset in *TIMT* type database *DB* and it is frequent in an interval $[t_1, t_2]$. Database support of *X* in an interval $[t_1, t_2]$ is computed from Eq. (2) as follows:

$$dsupp_{[t1,\,t2]}(X, DB) = DF_{[t1,\,t2]}(X, DB) / |DB|$$

Let us consider *DB* as a *TIMT*-type database shown in Example 1, where *x*(*ç*) denotes item *x* purchased *ç* numbers at a time in the corresponding transaction. The concept of *TF* and *DF* has been elaborated with respect to the *TIMT* database presented in Table 1. For example, itemset {*e*} has *TF* and *DF* values of 10 and 58, respectively, in database *DB*. Here, *TF* represents the traditional

frequency measure of binary transaction dataset, which could be used to compute the support of an itemset. Here one must note that *dsupp* will be more than 1. Since our data analysis requires quantity of an itemset we consider *DF* rather than *dsupp*.

6. The proposed Approach

A frequent pattern (Agrawal, 1993) is an itemset whose frequency in database is larger than or equal to minimum support (*minsupp*). With respect to *TIMT* type database *DB* given in Example 1, we mine the intervals where itemsets are frequent. We always consider closed time intervals of the form $[t_1, t_2]$ where t_1 and t_2 are time-stamps. We say that a transaction is in the time interval $[t_1, t_2]$ if the timestamp of the transaction say t is such that $t_1 < t < t_2$. We compute the transaction support and database frequency of an itemset in a time interval $[t_1, t_2]$ as given in previous section. Periodic itemsets with quantity are mined from list of intervals.

7. Interesting Intervals

To mine the intervals where itemsets are locally frequent the intervals must satisfy some constraints. Let itemset X be frequent in time intervals $[t_i, t_i']$, $i = 1, 2, \ldots, n$. The intervals are interesting if length of the intervals is larger than or equal to *mininterval*, and transaction support (*tsupp*) is larger than or equal to *minsupp*. Here *mininterval* is the minimum period length of a time interval. Each interval should be of sufficient length; otherwise, a pattern appearing once in a transaction also becomes frequent in an interval. Let *maxgap* be the user-defined maximum gap (time units) between the current time stamp of a pattern and the time stamp of the pattern when it was last seen. If the gap between the current time stamp of a pattern and the time stamp of the pattern when it was last seen is greater than *maxgap*, then a new interval is formed for the pattern with the current time stamp as the start of the interval. In addition, the previous interval of the pattern was ended when it was last seen. For this purpose, we shall impose the restriction that the gap between two intervals should be greater than or equal to *maxgap*. Therefore, every interesting interval must satisfy the constraints such as *minsupp*, *mininterval*, and *maxgap*. These constraints can assure that the itemsets must occur in a reasonable period of time. Since, we have taken time of the transaction as calendar date (dd/mm/yy) *mininterval*, and *maxgap* are considered as number of days.

8. Occurrence of an Itemset in an Interval

Let $X = \{x_1, x_2, \ldots, x_k\}$ be an itemset in database *DB*. Also, let t be a transaction in *DB*. Let item x_i be purchased η_i numbers at a time in τ, for $i = 1, 2, \ldots, k$. Then, $TIF(X, \tau, DB) = minimum\ \{\eta_1, \eta_2, \ldots, \eta_k\}$ was proposed by Adhikari & Rao (2008). Based on the concept of *TIF* authors proposed database frequency *DF* as follows: $DF(X, DB) = \Sigma_{\tau \in DB} TIF(X, \tau, DB)$.

Adhikari & Rao (2008) mined association rules (Agrawal & Srikant, 1994) from *TIMT* database and therefore the quantity of every itemset in each transaction has significant impact on another itemset. But this concept might not be effective for this particular application since after mining the interesting intervals, our interest is to compute the number of occurrences of itemsets in that specific interval. This knowledge provides retailer, meaningful information about inventory management. Therefore, we focus on the total number of occurrence of an itemset in an interval rather than a single transaction. We store individual number of occurrences for an itemset with higher level in that period and it is more meaningful.

9. Equality of Intervals / gaps

Periodic pattern is a pattern that repeats itself with a specific period in a given sequence. In this article we are dealing with locally frequent itemset. For each locally frequent itemset extracted by algorithm (Adhikari, 2014) a list of interesting time intervals is maintained. Locally frequent patterns in time-stamped data will be periodic when it repeats after certain gap and it is frequent for same length of interval. But in real life transactions, it might be difficult to get same length / gap for all the intervals. This incident occurs more commonly in real world database. Thus, we incorporate the concept of almost equal intervals / gaps. Each interval is represented as [*start, end*] where *start* gives the starting time-stamp of the time interval and *end* gives the ending time-stamp of the time-interval. As the interval is closed, $(end - start) + 1$ gives the length of the time interval. Given two intervals for the same itemset $[start_1, end_1]$ and $[start_2, end_2]$ it is always $start_2 > end_1$ and $(start_2 - end_1) - 1$ gives the gap length or distance between two time intervals. To extract periodic frequent itemsets we find the time gap between any two consecutive interesting time intervals. If the time gaps between consecutive intervals are found to be almost equal in length and also the lengths of the frequent intervals are found to be almost equal then we call these frequent sets as periodic frequent sets. Let $leng_1$, $leng_2$ be the

of the length intervals $[start_1, end_1]$ and $[start_2, end_2]$ respectively. Intervals $[start_1, end_1]$ and $[start_2, end_2]$ are said to be almost equal in length if $|leng_1 - leng_2| < \delta$ where δ is variation of length specified by user. Similarly, g_1, g_2 be the gap of two intervals and they are said to be almost equal in length if $|g_1 - g_2| < \delta$.

Example 2. We refer to the database of Example 1 to illustrate the various concepts discussed above. Let us consider the pattern $\{a, c\}$. Let the value of *maxgap*, *mininterval* and *tsupp* be 30 days, 10 days and 0.6 respectively. Let the value of variation of length $\delta = 7$ days. Table 2 shows all the intervals where the pattern $\{a, c\}$ is frequent.

In addition to support, we also compute the database frequency *DF* of an itemset for an interval. Here pattern $\{a, c\}$ is frequent in four intervals.

Interval	***mininterval*(days)**	***maxgap*(days)**	***tsupp***	**{*a*(quantity), *c*(quantity}**
[28/05/2000-28/06/2000]	32	-	0.8	{*a*(76), *c*(93)}
[01/08/2000-08/09/2000]	39	33	0.6	{*a*(47), *c*(32)}
[15/10/2000-18/11/2000]	34	37	1.0	{*a*(32), *c*(40)}
[23/12/2000-20/01/2000]	29	35	0.8	{*a*(15), *c*(22)}

Table 2. Locally frequent pattern $\{a, c\}$ along with quantity

10. Problem Statement

To find out the periodicity for each frequent itemset we proceed as follows. If the first interesting interval is almost equal in length with second interval then we see whether the time gap between the first and the second time interval is almost equal in length with the time gap between the second and third periods. If it is, then we take the average of the first two time gaps and see whether it is almost equal to the time gap between the third and the fourth periods. If the average length of the first two intervals is almost equal in length with the third interval, we proceed further or otherwise stop. In general if the average lengths of the first (n-1) interesting intervals is almost equal to the length of the n-th frequent interval and the average of first (n-2) time gaps are almost equal to the (n-1)-th time gap, then the average of n frequent intervals is compared with (n+1)-th frequent interval and that of the first (n-1) time gaps is compared with the n-th time gap. This way we can extract cyclic patterns. Based on above discussion, we state our problem as follows.

Let *DB* is a *TIMT*-type transactional database where each record contains items purchased along with their quantity as well as the date of the transaction and transaction identifier *TID*. *DB* is ordered in the ascending order of the time-stamps. Given the user-defined thresholds minimum support (*minsupp*), minimum interval length (*mininterval*) and maximum gap between two intervals (*maxgap*), and variation of length ($\ddot{a}$) discover the periodic itemsets along with quantity in *DB*.

11. Different Periodic Patterns

In above we define the equality between two time intervals associated with a locally frequent itemsets as follows: two time intervals are said to be almost equal if their lengths are equal up to a small variation otherwise they are unequal. Based on length of intervals and gap between two intervals we mine two types of periodic patterns (i) cyclic: where the cyclicity is defined in terms of equality among the intervals associated with an itemset as well as that of the time gaps. Here time gap is the gap between two consecutive time intervals associated with the frequent itemset (ii) acyclic: where the lengths of time intervals associated with it is almost equal but at least any two of their time gaps is unequal.

In Example 2, an analysis of pattern $\{a, c\}$ is presented in different intervals. In this article, we might not require to store the support of all the intervals because here we are dealing with the *TIMT* database. It is obvious that the pattern is frequent in all the intervals. Therefore, instead of storing support values, one could think of storing the number of occurrences of an itemset in all the intervals. Let there be n database frequency (*DF*) of a pattern corresponding to n intervals. While storing the number

of occurrences of an itemset for all the intervals, we use here the range measure for a set of values. Here, the lowest range and the highest range are formed with the minimum and maximum of *n* intervals.

In Example 2 we consider first two intervals for pattern $\{a, c\}$. Two intervals are almost equal as $|32 - 39| = 7 < \delta$. Then average of first two intervals 35.5 is compared with third interval (34) i.e. $|35.5 - 34| = 1.5 < \delta$. Therefore third interval is also equal to first two intervals. At last fourth interval (29) is compared to the average of three intervals i.e. 35 and $|35 - 29| = 6 < \delta$. Thus all four intervals are almost equal for pattern $\{a, c\}$. Similarly, all three *maxgaps* are also almost equal. Thus, pattern $\{a, c\}$ is a cyclic periodic pattern as all the intervals and gaps between the intervals are almost equal.

Table 3 shows $\{a, c\}$ is periodic in nature with average interval length (*avginvlen*) 33.5 days and average gap length (*avgaplen*) 35 days. It also shows minimum and maximum occurrence of *a* in four intervals 15 and 76 respectively. Similarly, minimum and maximum occurrence of *c* in four intervals is 22 and 93 respectively. This knowledge provides the retailer meaningful information about inventory management.

itemset	***avginvlen*(days)**	***avgaplen*(days)**	**Occurrence of *a***	**Occurrence of *c***	**Type of periodic pattern**
$\{a, c\}$	33.5	35	15…76	22…93	cyclic

Table 3. Analysis of pattern $\{a, c\}$

12. Algorithms for Mining Periodic Patterns

Mining periodic patterns from non-binary transactional database can be decomposed into the following steps:

1. Find the locally frequent items with quantity and intervals from *TIMT* database using Algorithm 1

2. Find the frequent itemsets of higher level with quantity and intervals using Algorithm 2

3. Mine the periodic itemsets using Algorithm 3.

Algorithm 1 and 2 were proposed by Adhikari (2014). The pseudocode of the algorithm is presented below.

Algorithm 1. Mine locally frequent items with quantity and their intervals

Procedure *MiningFrequentItems_One* (*DB, maxgap, mininterval, minsupp*)
***Inputs*:** *DB, maxgap, mininterval, minsupp*
***Outputs*:**
S: Locally frequent items with quantity and their intervals.
01: read a transaction $t \in DB$;
02: **while** not end of the transaction database *DB* **do**
03: **for** all the items present in *DB* **do**
04: **if** the item is seen first time in the transaction **then**
store the time as firstseen and lastseen;
05: **else** update lastseen;
06: form an interval;
07: **if** intervals satisfy *maxgap*, *mininterval* and *minsupp* **then** add occurrence of items of all transaction in that interval;
08: **else** discard the interval;
09: **end for**
10: **end procedure**

At line 1, the first transaction of database is read. Line 4 checks whether the item is first time seen in the transaction, and the necessary assignment is done at line 5. At line 6, the interval is constructed and the interestingness of the interval is checked at line 7. If the interval satisfies the criteria, then the occurrence of the items in that interval is also stored.

We shall now present below Algorithm 2, which uses the locally frequent itemsets obtained by Algorithm 1 and apriori property (Agrawal & Srikant, 1994). An array *level_1* is used to generate the candidate sets at the second level. Then, array *level_2* is used to generate candidate sets at the third level, and so on. We apply pruning using conditions at line 6 to eliminate some itemsets at the next level. This pruning step ensures that the size of the itemsets at the current level is one more than the size of an itemset at the previous level. In addition, pruning is applied using user-defined thresholds such as *maxgap*, *mininterval*, and *minsupp* at line 13. The data structure $S1$ is used to store higher-level frequent itemsets.

Algorithm 2. Mine locally frequent itemsets with quantity at higher level and their intervals

Procedure *MiningHigherLevelItemsets* (*DB*, *S*)

***Inputs*:** *DB*, *S*

***DB*:** Database to be mined

***S*:** Partially constructed data structure containing locally frequent itemsets of size one with quantity

***Outputs*:** Locally frequent itemsets at higher levels, their intervals and quantity in array $S1$

01: **let** L_1 = set of elements at *level_1* of *S* with quantity; **let** $k = 2$;

02: **while** $L_{k-1} \neq \phi$ **do**

03: $C_k = \phi$;

04: **for** each itemset $l_1 \in L_{k-1}$ **do**

05: **for** each itemset $l_2 \in L_{k-1}$ **do**

06: **if** $((l_1[1] = l_2[1]) \wedge \ldots \wedge (l_1[k\text{-}2] = l_2[k\text{-}2]) \wedge L_1 (l_1[k\text{-}1] < l_2[k\text{-}1]))$ **then**

07: $c = l_1 \bowtie l_2$; $C_k = C_k \bigcup c$;

08: **end if** {06}

09: **end for** {05}

10: **end for** {04}

11: **for** each element $c \in C_k$ **do**

12: construct intervals for *c* as mentioned in Algorithm 1;

13: **if** the intervals corresponding to *c* satisfy *maxgap*, *mininterval* and *minsupp* **then**

14: add *c* and the intervals to *level_k* of $S1$;

15: add individual occurrence of *c* for the intervals;

16: **end if** {13}

17: **end for** {11}

18: increase *k* by 1;

19: **let** L_k = set of elements at *level_k* of $S1$;

20: **end while** {02}

end procedure

Algorithm 3 is presented below to compute average interval length *(avgleng)* / average gap length (*avgap*) every time for each pattern. It also compares between *avgleng* / *avgap* and current interval length (*currleng*) / current gap (*currgap*). While checking equality of interval / gap, user-defined threshold δ is used. We use two arrays *intv* and *g* for storing length of interval / gap and occurrence range of itemset. At the end algorithm reports all the cyclic and acyclic periodic patterns. Collections *C* and *A* stores cyclic and acyclic periodic itemsets, respectively.

Algorithm 3. Mining periodic itemsets

Procedure *Periodic_Patterns* (*S*1, δ)
Inputs:
*S*1: a data structure locally frequent itemsets at higher levels, their intervals and quantity
δ: user-defined threshold to compare equality of interval / gap
Outputs: cyclic and acyclic periodic itemsets
01: **let** $k = 1$; **let** L_1 = elements at *level*_1 of *S*; **let** $C = \phi$; $A = \phi$;
02: **while** $L_k \neq \phi$ **do**
03: **for** each element *l* of L_k **do**
04: **let** $i_1, i_2, \ldots, i_r$ be the intervals corresponding to *l*;
05: Compute length of the intervals and gap between the intervals from *S*1;
06: Store minimum and maximum quantity as range;
07: **for** each interval **do**
08: compute average length of the interval;
09: **if** $|avgleng - currleng| < \delta$ **then**
10: add *l*, intervals, occurrence range in *intv*;
11: **end if** {09}
12: **end for** {07}
13**:** **for** each gap **do**
14: compute average gap between the interval;
15: **if** $|avgap - currgap| < \delta$ **then**
16: add *l*, gaps in *g*;
17: **end if** {15}
18: **end for** {13}
19: **if** all the intervals are equal for *l* **then**
20: **if** all the gaps between the intervals are equal **then**
21: *l* is cyclic; add the occurrence range to *C*;
22: **else** *l* is acyclic; add the occurrence range to *A*;
23: **end if** {19}
24: **end for** {03}
25: increase *k* by 1;
26: **let** L_k = set of elements at *level k* of *S*;
27: **end while** {02}
28: **for** each element $l \in C \cup A$ **do**
29: display *l*, their occurrence information;
30: **end for** {28}
end procedure

In Algorithm 3 itemsets are processed level-wise. At line 26, we move on to the next level. Intervals and gaps between the intervals of each itemset are processed in lines 4 – 6. Lines 7 – 18 check the equaliy between intervals / gaps for an itemset. Periodic itemsets (cyclic and acyclic) are displayed in lines 28 – 30.

13. Experimental Results

We have carried out several experiments to study the efficiency and effectiveness of our approach. All the experiments have been implemented on a 2.4 GHz, core i3 processor with 4 GB of memory, running Windows 7 HB, using Visual C++ (version 6.0) software. We present experimental results using *T10I4D100K* (see Reference 14) database. The characteristics of the database are given in Table 4.

Let *DB*, *NT*, *ALT, AFI,* and *NI* be the given database, the number of transactions, average length of a transaction, average frequency of an item, and the number of items, respectively.

DB	***NT***	***ALT***	***AFI***	***NI***	**Size in Megabytes**
T10I4D100K	1,00,000	11.10	1276.12	870	3.83

Table 4. Database characteristics

Due to the unavailability of the *TIMT* database, we have applied two pre-processing techniques on the database. First, we have attached time stamps as calendar date for the transactions because the records in these databases contain only items purchased in transactions. Therefore, a program was written to incorporate temporal features in the dataset. We assume that each year contains 365 days and the market remains open on all days. We discuss here the issue of handling items that are measured in continuous scale. Consider the item milk in a departmental store. Let there be four types of milk packets: 0.5, 1, 1.5, and 2 kL. The minimum packaging unit could be considered as 1 unit. Thus, 3.5 kL of milk could be considered as 7 units of milk. If an item is present in a transaction, then the number of occurrences of the items is generated randomly between 1 and 5. Thus, a binary transactional database gets converted into a time-stamped non-binary transaction database. The process is repeated to generate the datasets of different sizes for conducting experiment. We have given the maximum number of transactions and minimum number of transactions in such a way that the lifetime of each size of dataset is almost one year. Pre-processed datasets are shown in Table 5.

Data size(No. of transactions)	**Minimum transactions /day**	**Maximum transactions /day**	**Starting date, ending date**
10,000	20	30	1/1/2010, 31/12/010
20,000	40	60	1/1/2010, 28/12/2010
30,000	70	90	1/1/2010, 2/1/2011
40,000	100	125	1/1/2010, 30/12/2010
50,000	125	150	1/1/2010, 27/12/2010
60,000	160	185	1/1/2010, 31/12/2010
80,000	200	230	1/1/2010, 1/1/2011
1,00,000	250	300	1/1/2010, 29/12/2010

Table 5. Pre-processed datasets

The usage of constraints is application specific. Depending on the patterns that the user is targeting and also on the nature of the dataset, an optimal setting can be defined for the constraints. This setting will result in the elimination of undesired patterns and at the same time provide further pruning. On the other hand, the requirement of an organization might determine an important parameter for mining periodic patterns. The distribution of items in databases also matters in selecting the right values of

mininterval and maxgap. For a sparse database maxgap could be longer, and it could be even longer than mininterval provided minsupp remains small.

In Figure 1 and Figure 2 number of cyclic and acyclic itemsets obtained by our method *for the different sizes of* datasets is given. We observe that if the number of transactions is increased keeping other parameters constant like, minsupp, mininterval, maxgap and ä the number of cyclic itemsets increases.

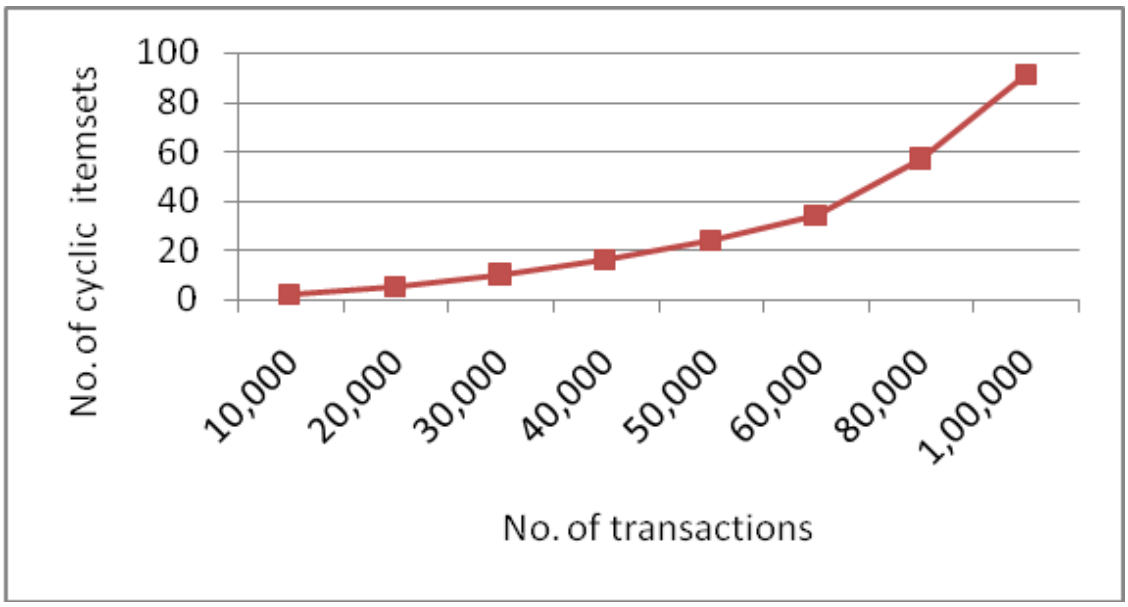

Figure 1. No. of cyclic itemsets vs. size of database (*T10I4D100K*) (*minsupp*=0.18, *mininterval* =30, *maxgap* =50, δ=10)

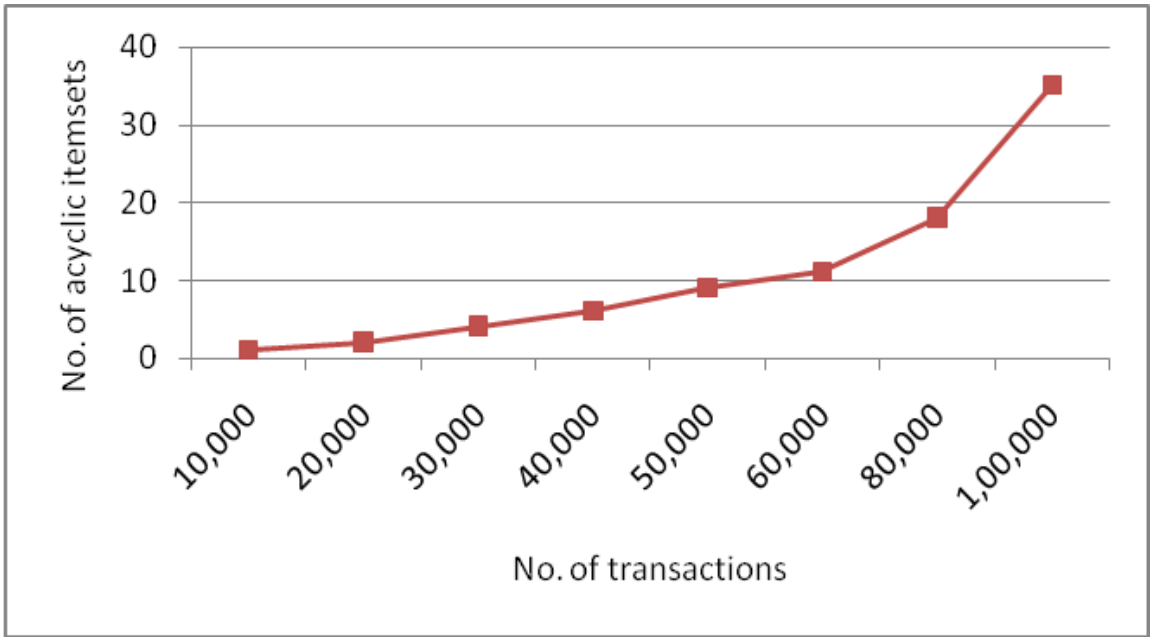

Figure 2. No. of acyclic itemsets vs. size of database (*T10I4D100K*) (*minsupp*=0.18, *mininterval*=30, *maxgap*=50, δ=10)

The number of interesting intervals could increase by lowering the thresholds *minsupp*, *mininterval*, *maxgap*. The scalability of the algorithms is measured with respect to database sizes. In Figure 3 the relationship between the database size and execution

time for mining periodic patterns has been shown. We observed that the number of patterns increases as the number of transactions increases. Thus, execution time increases with an increase in database size.

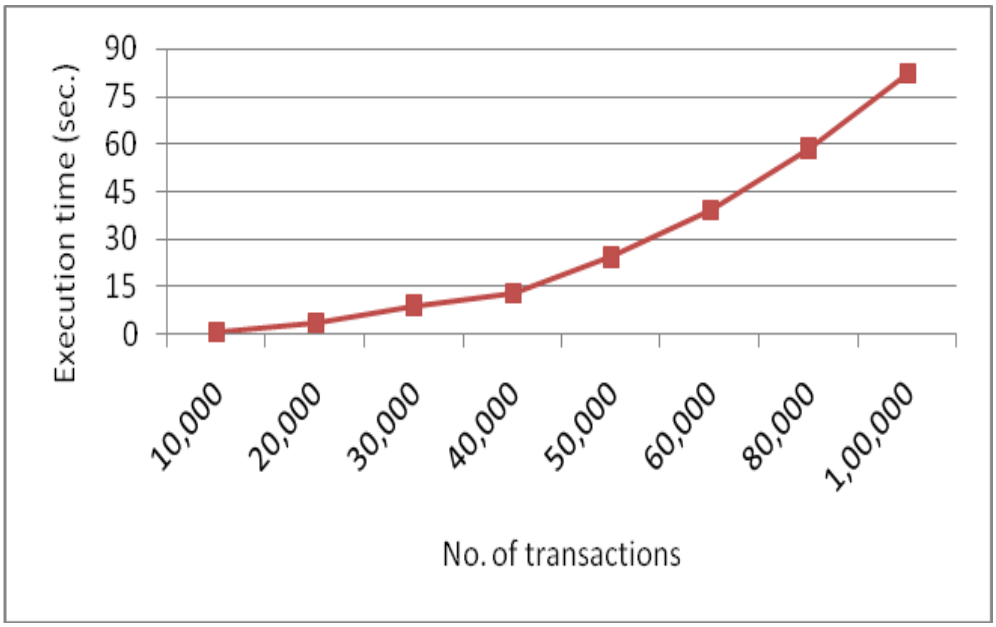

Figure 3. Execution time vs. size of database (*minsupp* = 0.18, *mininterval* =30, *maxgap*= 50)

The scalability of the algorithms is measured with respect to database sizes. In Figure 3 the relationship between the database size and execution time for mining periodic patterns has been shown. We observed that the number of patterns increases as the number of transactions increases. Thus, execution time increases with an increase in database size.

14. Conclusions

We have presented a method for mining periodic patterns from non-binary transactional databases. We mine locally frequent itemsets along with the set of intervals and their occurrences. The algorithm has been designed to mine frequent itemsets along with intervals and quantity. In addition, an algorithm has been designed to check type of periodic pattern. Experimental results also report whether a periodic pattern is cyclic or acyclic. With the growing size of the datasets, development of incremental periodic pattern mining algorithms has become a necessity. The proposed algorithm is incremental in nature. When new transactions are added it is not necessary to process the whole dataset again. Obviously the newly added transactions will have time-stamps that are after the last time-stamp of the earlier data. This pattern analysis can provide an understanding of sales patterns, help a company manage inventory, or plan promotion and marketing spending on specific product lines.

References

[1] Yin, K. C., Hsie, Y. L., Yang, D. L., Hung, M. C. (2014). Association rule mining considering local frequent patterns with temporal intervals. *Applied Mathematics and Information Sciences*, 8 (4) 1879-1890.

[2] Hu, Y. H. Chiang, I. C. (2011). Mining cyclic patterns with multiple minimum repetition supports, *In: F*uzzy Systems and Knowledge Discovery (FSKD), 8th International Conference (3) 1545- 1549 IEEE

[3] Liu, Jian., Wang, Zhenrui., Li, Mingyang., Zhang, Biao(2017). Detecting Asynchronous Periodic Patterns of Intervals in Temporal Sequence Data, IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER) 91-96

[4] Adhikari, J. (2014). Mining Calendar-based periodic patterns from non-binary transactions. *Journal of Intelligent Systems*, 29 (3) 277-29.

[5] Adhikari, J., Rao, P. R. (2013). Identifying Calendar-based Periodic Patterns. *Emerging Paradigms in Machine Learning*, S. Ramanna, L. Jain and R. J. Howlett, (Ed.). (329 – 357). Springer, Heidelberg.

[6] Mazarbhuiya, F. A., Shenify, M., Khan, A., Farooq, A. (2012). Finding cyclic frequent itemsets. *International Journal of Computer Science Issues*, 9 (6) 1694 - 0814.

[7] Mazarbhuiya, F. A. (2016). Discoving acyclic patterns. *International Journal of Advance Research in Computer Science and Management Studies*, 4 (1) 130-133.

[8] Kiran, R. U., Kitsuregawa, M. (2013). Discovering quasi-periodic-frequent patterns in transactional databases. *Big Data Analytics*, 8302 LNCS, 97-115.

[9] Barreto, A., Antunes, C. (2014) Finding periodic regularities on sequential data: converging, diverging and cyclic patterns, C3S2E 2014, 19, 1-4.

[10] Mahanta, A. K., Mazarbhuiya, F. A., Baruah, H. K. (2008). Finding calendar-based periodic patterns, *Pattern Recognition Letters.* 29, 1274 – 1284.

[11] Nishi, M. A., Chowdhury, F. A., Samiullah, Md. Jeong, B. S. (2013). Effective periodic pattern mining in time series databases. *Expert Systems with Applications*, 3015–3027.

[12] Chanda, A.K., Chowdhury F. A. Samiullah Md. Leung C. K. (2017). A new framework for mining weighted periodic patterns in time series databases. *Expert Systems with Applications,* 79. 207-224.

[13] Kiran, R. U., Kitsuregawa, M., Reddy, P. K. (2016). Efficient discovery of periodic-frequent patterns in very large databases, *Journal of Systems and Software*, 112, 110–121.

[14] Akter, S., Karim Md. R., Samiullah Md., Chowdhury F. A. (2018) Mining non-redundant closed flexible periodic patterns. *Engineering Applications of Artificial Intelligence*, 69. 1-23.

[15] Adhikari, A., Rao, P. R. (2008). Association rules induced by item and quantity purchased. *DASFAA*, 478 – 485.

[16] Agrawal, R., Imielinski, T., Swami, A. (1993). Mining association rules between sets of items in large databases. *Proceedings of the ACM SIGMOD Conference Management of Data*, p. 207-216.

[17] Agrawal, R., Srikant, R. (1994). Fast algorithms for mining association rules, *In:* Proceedings of 20 th Very Large Databases (VLDB) Conference, 487 – 499.

[18] Frequent itemset mining dataset repository, http://fimi.cs.helsinki.fi/data