CHAPTER – IV

FINDING PERIODIC AND SEQUENTIAL PATTERNS

4.1 INTRODUCTION

The algorithm [2] described in section-3.3 of previous chapter extracts for each frequent item set a sequence of time intervals in which the item set is frequent. For certain item sets the associated time intervals may have some nice property. For example an item set may be periodic. Periodic patterns may be of different types. In section-4.2 of this chapter, we discuss some of these patterns and devise algorithms for extracting such patterns. We then report some results obtained.

The above study is intra-item set study because we are trying to find patterns within the time intervals for a single item set. Next we also carry out inter-item set study. Patterns may be seen among the time intervals associated with different item sets. For example the time intervals of two different item sets may be in some sequence. In section-4.3 of this chapter, we report on our study made in this direction. In section-4.4, we give a brief conclusion of this chapter.
4.2 PERIODIC PATTERNS

4.2.1 INTRODUCTION

In the algorithm [2] proposed in section-3.3 of chapter-III, for each locally frequent item set, a list of time intervals is extracted in each of which the set is frequent. In this part of our study, our concentration is not on the item sets but on the time intervals associated with the item sets. We try to find patterns in the time intervals associated with each item set. The problem we address now is to find periodic patterns if they exist. The first type of periodic pattern that we want to extract is to find item sets for which the time gaps between successive intervals are almost equal. For example some item sets may become frequent at every alternate year. The proposed method for finding such patterns is discussed in section-4.2.3 of this chapter. We also discuss some variations of such pattern in the same section.

If the time-stamps stored in the transactions are calendar-dates, then we may have yearly, monthly, daily patterns. Examples of such patterns are some item sets may be frequent in mid of every year, some item sets are frequent in the first half of every month and some item sets are frequent in every morning etc. The discovery of periodic patterns was studied in [6], and [54]. In [6], discovery of cyclic association rules (i.e., the association rules that occur periodically over time) was studied. However, periodicity has limited expressiveness in describing real life concept such as the first business day of every month since the distances between two consecutive such business days are not constant. In general, their
model does not deal with *calendric* concept like year, month, day, etc. In [54], the work of [6] was extended to treat user-defined temporal patterns. Although the work of [54] is more flexible than that of [6], it only considers the association rules that hold during the user-given time intervals described in terms of *calendar algebraic expression*. So the method requires user's prior knowledge about the temporal patterns. In [59], the author proposes a method for discovering temporal association rules with relaxed requirement of prior knowledge. Instead of using cyclic or user-given *calendar algebraic expressions*, they use *calendar schemas* as frameworks for discovering temporal patterns. But in our case, no prior information is required and same algorithm can be suitably modified for finding different types of periodic pattern if such patterns exist. In section-4.2.3, we report in details the work done in this regard.

In computing periodicity of patterns, keeping the time intervals is also important. In the same section (i.e. section-4.2.3), we propose a method to store the time intervals, which will turn out to be *fuzzy intervals*.

Next, we discuss about the pattern holding in sufficient number of intervals. For example some item set may be frequent in not all *Sundays* but say 80% of the *Sundays* throughout the life-span of the item set. We call such item sets partially periodic. The partially periodic pattern-mining problem was studied in [59]. In [59], the authors discuss about a method of extracting temporal association rules with respect to *fuzzy match* i.e. association rule holding during "enough" number of intervals given by the corresponding calendar pattern. They called the percentage parameter as match ratio. The mining of partial periodic patterns in
time series database is studied by Han et al [27]. Partial periodic patterns specify the behaviour of the time series at some but not all points in time. Although we are interested to extract similar type of patterns but our approach is different from earlier approaches. In our proposed method the same algorithm can be used to find fully and partially periodic patterns. In section-4.2.3, we report on our study made in this regard. In section-4.2.4, we report some results obtained.

Next, we discuss about the clustering of frequent patterns where each frequent pattern is associated with a fuzzy time interval describing the period of the pattern. In section-4.2.5, we briefly report on our study made in this regard.

4.2.2 DEFINITIONS AND KNOWN RESULTS

DEFINITION 1: ALMOST EQUAL INTERVALS

For each locally frequent item set extracted by algorithm [2], a list of time intervals is kept in which the set is frequent where 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. end – start gives the length of the time interval. Given two intervals [start₁, end₁] and [start₂, end₂] if the intervals are non-overlapping and start₂ > end₁ then start₂ – end₁ gives the distance between the time intervals. Similarly two intervals [start₁, end₁] and [start₂, end₂] are said to be almost equal in length if the length of the both intervals are equal up to a small variation say λ% i.e. (end₁− start₁)+λ%of (end₁− start₁) is equal to (end₂− start₂) or (end₂− start₂)+λ%of (end₂− start₂) is equal to (end₁− start₁) where λ is specified by user.
DEFINITION 2: SET SUPERIMPOSITION

When we overwrite, the overwritten portion looks darker for obvious reasons. The set operation union does not explain this phenomenon. After all

\[ A \cup B = (A-B) \cup (A \cap B) \cup (B-A) \]

And in \((A \cap B)\) the elements are represented once only.

Baruah [21] introduced an operation called superimposition denoted by \((S)\). If \(A\) is superimposed over \(B\) or \(B\) is superimposed over \(A\), we have

\[ A (S) B = (A-B) (+) (A \cap B)^{(2)} (+) (B-A) \]

Where \((A \cap B)^{(2)}\) are the elements of \((A \cap B)\) represented twice, and (+) represents union of disjoint sets.

To explain this, an example has been taken.

If \(A= [a_1, b_1]\) and \(B= [a_2, b_2]\) are two real intervals such that \(A \cap B \neq \emptyset\), we would get a superimposed portion. It can be seen from (1)

\[ [a_1, b_1] (S) [a_2, b_2] = [a_{(1)}, a_{(2)}] (+) [a_{(2)}, b_{(1)}]^{(2)} (+) [b_{(1)}, b_{(2)}] \]

where

\[ a_{(1)} = \min(a_1, a_2) \]
\[ a_{(2)} = \max(a_1, a_2) \]
\[ b_{(1)} = \min(b_1, b_2) \]
\[ b_{(2)} = \max(b_1, b_2) \]
(2) explains why- if two line segments are superimposed, the common portion looks doubly dark [21]. The identity (2) is called fundamental identity of superimposition of intervals.

Let now, \([a_1, b_1]^{(1/2)}\) and \([a_2, b_2]^{(1/2)}\) be two fuzzy sets with constant membership value \(1/2\) everywhere (i.e. equi-fuzzy intervals with membership value \(1/2\)). Applying (2) on the two equi-fuzzy intervals we can write

\[
[a_1, b_1]^{(1/2)}(S)[a_2, b_2]^{(1/2)} = [a_1(a_2), a_1(b_2)]^{(1/2)}(+) [a_2(a_1), a_2(b_1)]^{(1/2)}(+) [a_1(b_1), a_1(b_2)]^{(1/2)}
\] … (3)

To explain this we take the fuzzy intervals \([1, 5]^{(1/2)}\) and \([3, 7]^{(1/2)}\) with constant membership value \(1/2\) given in figure-4.1 and figure-4.2. Here \([1, 5] \cap [3, 7] = [3, 5] \neq \emptyset\)

\[1/2\]

\[1\]

\[3\]

\[5\]

\[7\]

Fig-4.1

\[1/2\]

\[3\]

\[7\]

Fig-4.2

\[1/2\]

\[1\]

\[3\]

\[5\]

\[7\]

Fig-4.3

77
If we apply superimposition on the intervals then the superimposed interval will be consisting of $[1, 3]^{(1/2)}$, $[3, 5]^{(1)}$ and $[5, 7]^{(1)}$. Here the membership of $[3, 5]$ is (1) due to double representation and it is given in figure-4.3.

Let $[x_i, y_i]$, $i=1,2,...,n$, be $n$ real intervals such that $\bigcap_{i=1}^{n} [x_i, y_i] \neq \emptyset$. Generalizing (3) we get

$$[x_1, y_1]^{(1/n)} (S) [x_2, y_2]^{(1/n)} (S) ... (S) [x_n, y_n]^{(1/n)}$$

$$= [x_{(1)}, y_{(1)}]^{(1/n)} (+) [x_{(2)}, y_{(2)}]^{(2/n)} (+) ... (+) [x_{(n)}, y_{(n)}]^{(n/n)}$$

$$(+) ... (+) [y_{(n+1)}]^{(1)} (+) [y_{(n+2)}]^{(2/n)} (+) ... (+) [y_{(n+1)}]^{(1/n)}$$

$$= [y_{(n+1)}]^{(1/n)} (+) [y_{(n+2)}]^{(2/n)} (+) ... (+) [y_{(n+1)}]^{(1/n)} ... (4)$$

In (4), the sequence $\{x_{(i)}\}$ is formed of the sequence $\{x_i\}$ in ascending order of magnitude for $i=1,2,...,n$ and similarly $\{y_{(i)}\}$ is formed of the $\{y_i\}$ in ascending order [21]. Observe that in (4) the membership values of $[x_{(r)}, x_{(r+1)}]^{(r/n)}$, $r=1,2,...,n-1$ look like an empirical probability distribution function and the membership values of $[y_{(r)}, y_{(r+1)}]^{(r/n)}$, $r=1,2,...,n-1$ look like the values of an empirical complementary probability distribution function or empirical survival function.

**DEFINITION 3: POSSIBILISTIC MEAN AND POSSIBILISTIC VARIANCE OF A FUZZY NUMBER**

Let $F$ be a family of fuzzy number and $A$ be a fuzzy number belonging to $F$. Let $A_\alpha=[a_1(\alpha), a_2(\alpha)], \alpha \in [0, 1]$ be an $\alpha$-cut of A. Carlsson and Fuller [8], [9] defined the interval-valued possibilistic mean of fuzzy number $A \in F$ as
\[ M(A) = [ M*(A), M^*(A) ] \] ...(5)

where the lower possibilistic mean value of \( A \) is given by

\[ M_*(A) = \frac{1}{2} \int_0^1 \alpha a_1(\alpha) d\alpha \]

Similarly, the upper possibilistic mean value of \( A \) is given by

\[ M^*(A) = \frac{1}{2} \int_0^1 \alpha a_2(\alpha) d\alpha \]

And the possibilistic variance of \( A \in F \) is given by

\[ \text{Var}(A) = \frac{1}{2} \int_0^1 \alpha (a_2(\alpha) - a_1(\alpha))^2 d\alpha \] ...(6)

Thus the variance of \( A \) is defined as the expected value of the squared deviations between arithmetic mean and the endpoints of its \( \alpha \)-cuts.

And the standard deviation is defined as

\[ \sigma_A = \sqrt{\text{Var}(A)} \] ...(7)

Before proceeding further we review the theorem given by Carlsson and Fuller [8].

**THEOREM 4.2.1:** The variance of a fuzzy number is invariant to shifting.

**Proof:** Let \( A \in F \) be a fuzzy number and let \( \theta \) be a real number. If \( A \) is shifted by value \( \theta \), then we get a fuzzy number, denoted by \( B \), satisfying the property
\( B(x) = A(x - \theta) \) for all \( x \in \mathbb{R} \). Then from the relationship \( B_\alpha = [a_1(\alpha) + \theta, a_2(\alpha) + \theta] \), we find

\[
\text{Var}(B) = \frac{1}{2} \int_0^1 \alpha((a_2(\alpha) + \theta) - (a_1(\alpha) + \theta))^2 \, d\alpha
\]

\[
= \frac{1}{2} \int_0^1 \alpha(a_2(\alpha) - a_1(\alpha))^2 \, d\alpha
\]

\[
= \text{Var}(A)
\]

### 4.2.3 EXTRACTION OF PERIODIC PATTERNS

One way to extract these sets is to find the time gap between any two consecutive frequent time intervals of the same set. If the time gaps between consecutive intervals are found to be \textit{almost equal} in length and also the lengths of the frequent intervals are found to be \textit{almost equal} (the definition of \textit{almost equal} in length is given in section 4.2.2) then we call these frequent sets as periodic frequent sets. Now to find out such type of periodicity for each frequent item set we proceed as follows. If the first frequent interval is \textit{almost equal} in length with second frequent interval then we see whether the time gap between the first and the second time interval is \textit{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 of frequency is \textit{almost equal} in length with the third interval of frequency, we proceed further or
otherwise stop. In general if the average lengths of the first \((n-1)\) frequent intervals is \textit{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 periodic patterns if such patterns exist. We describe below the algorithm for extracting periodically frequent item sets.

4.2.3.1 ALGORITHM FOR EXTRACTING PERIODICALLY FREQUENT ITEM SETS

ALGORITHM 4.1

\[
\text{for each frequent item set iset do} \\
\text{(L \leftarrow list of time intervals for iset)} \\
\text{t1 = L.get();} \\
\text{// t1 is now pointing to the first interval in L} \\
t1s = t1.stime(); \\
t1e = t1.etime(); \\
11 \leftarrow t1e-t1s \\
t2 = L.get(); \\
t2s = t2.stime(); \\
t2e = t2.etime(); \\
12 \leftarrow t2e-t2s \\
\text{if not almostequal(11,12,sgma) then} \\
\text{\quad \{report that iset is not periodic in nature} \\
\text{\quad \quad Continue} \\
\text{\quad \quad \quad \quad /* go for the next frequent item set */} \\
\} \\
n=1 \\
\text{avgts} \leftarrow t2s-t1e \\
\text{avglen} \leftarrow (11 + 12)/2 \\
\text{flag} = 0 \\
\text{while (tint = L.get()) != null)} \\
\text{\{ ts = tint.stime();} \\
\text{\quad te = tint.etime();} \\
\text{\quad tg \leftarrow ts-t2e} \\
\text{\quad if almostequal(tg, avgts, sgma) then} \\
\text{\quad \quad avgts \leftarrow (n*avgts + tg)/(n+1)} \\
\text{\quad else} \\
\text{\quad \quad \{flag = 1; break;\}} \\
\text{\quad len \leftarrow te - ts} \\
\]
if almostequal(len, avglen, sigma) then
    \[ \text{avglen } \left(\frac{(n+1)\cdot \text{avglen} + \text{len}}{n+2}\right) \]
else
    \{ flag = 1; break; \}
    \( t2 \leftarrow t \)
    \( n \leftarrow n + 1 \)
\}
if (flag == 1)
    report that iset is not periodic in nature
else
    report that iset is periodic in nature with average time gap avgtg and average period length avglen
\}

Here the function \( \text{stime}() \) returns the starting time of the corresponding time interval and the function \( \text{etime}() \) returns the ending time. The function \( \text{get()} \) returns a pointer to the next node of the current node in a linked list. It has been defined in a class used for iterating through a linked list (details given in chapter-III).

We may have some patterns where the time gaps are \emph{almost equal} in length but the duration of the intervals of frequency are not equal even in the approximate sense. We may also have some patterns where the time-gaps are not equal in length but durations of the intervals are \emph{almost equal}. The above algorithm can be modified accordingly to find such patterns.

Now if the time-stamps stored in the transactions are the time hierarchy of the type \emph{hour day month year}, then we may have the following types of periodic patterns:

i) **Yearly periodic pattern:** Patterns that appear in yearly basis are called yearly pattern. For example in a *market-basket dataset* we normally see that the sales of *cold drinks* go up during summer. So in each year in the summer the item *cold drink* (or the different brands of it, if these are kept in that form) will become frequent. But this kind of periodicity is of a special type. For example, here if we ignore the year of the date associated with the transactions and consider only the month, day and hour (in such cases the ordering of the transactions will be taken into account so that a month and a day appearing earlier than another such pair in the sequence of transactions will indicate that the transactions appearing first has taken place earlier than the other), then the periods in which an item set is frequent will have large overlapping. This may happen in case of hourly, daily and monthly patterns too.

ii) **Monthly periodic pattern:** To extract monthly pattern, we do not consider month and year in the time hierarchy and check whether there exists any monthly pattern or not. An example of such pattern is that an item set is frequent in the first part of every month.

iii) **Daily periodic pattern:** To extract daily pattern, we do not consider day, month and year in the time hierarchy and check whether there exists any daily pattern or not. The method of extracting the daily pattern is similar to that of monthly pattern. An example of such pattern is that a set is frequent in the first half of every day.
For storing information of the periods of these periodic frequent sets we propose a method that is similar to the concept of set superimposition defined in [21] (the Definition of Superimposition of sets is given in section- 4.2.2). It is also true that such periods normally will have non-empty intersections. For example in our country the winter season (when the temperature goes down) starts around November/December and continues till February/March. These cut off times are not always same. It varies from year to year but even then these durations never have empty intersections. For example it will never happen that in some year the winter season started after March. Under this assumption and the assumption that the items are uniformly distributed in the period in which it is frequent (which is reasonable as we considering locally frequent sets, ignoring the time periods in which an item set appears rarely), we propose our representation scheme below.

With each periodic pattern we define a term called ‘certainty’ of the pattern in time-slots. Considering all the periods in which the pattern holds, we use a concept similar to the concept of superimposition of sets defined in [21], to define the certainty of the patterns in subintervals of the interval obtained by taking union of the intervals in such a way that the certainty of the pattern in the overlapping subintervals is more than the certainty in the other subintervals.

Suppose a pattern holds in $[t_1, t_1']$ and $[t_2, t_2']$ where the intersection $\cap_{i=1}^{2}[t_i, t_i'] \neq \emptyset$. Let $t^{(1)}$, $t^{(2)}$ be the ordered values of $t_1$, $t_2$ and $t^{(0)}$ and $t^{(2)}$ be the ordered values of $t_1'$ and $t_2'$ and since the intervals are overlapping we have $t^{(2)} < t^{(0)}$. Now we define the certainty of a pattern in the intervals $[t^{(1)}, t^{(2)}]$, $[t^{(2)}, t^{(0)}]$ and
\([t^{(1)}, t^{(2)}]\) as \(\frac{1}{2}, 1\) and \(\frac{1}{2}\) respectively. We use the following notation to describe this

\[
[t_1, t'_1] (S) [t_2, t'_2] = [t^{(1)}, t^{(2)}]^{1/2} [t^{(2)}, t^{(1)}] [t^{(1)}, t^{(2)}]^{1/2}
\]

... (1)

where \( (S) \) is the symbol used for denoting the operation of superimposition of sets.

Let \([t_1, t'_1], i = 1, 2, \ldots, n\) be \(n\) time intervals such that \(\bigcap_{i=1}^{n}[t_i, t'_i]\neq \phi\). Then generalizing (1) we have

\[
[t_1, t'_1] (S) [t_2, t'_2] (S) \ldots (S) [t_n, t'_n] =
\]

\[
[t^{(1)}, t^{(2)}]^{1/n} [t^{(2)}, t^{(3)}]^{2/n} [t^{(3)}, t^{(4)}]^{3/n} \ldots \ldots [t^{(r)}, t^{(r+1)}]^{r/n} \ldots [t^{(n)}, t^{(n)}]^{n/n}
\]

where \(\{t^{(i)}\}_{i=1}^{n}\) is the sequence obtained from \(\{t_i\}_{i=1}^{n}\) by sorting in the ascending order and \(\{t'^{(i)}\}_{i=1}^{n}\) is obtained from \(\{t'_i\}_{i=1}^{n}\) by sorting in the ascending order. For any given time-stamp, we can easily calculate the certainty of the pattern in that time-stamp by using the following relation.

For any time-stamp \(t\) in \([t^{(n)}, t^{(1)}]\) certainty value is 1.

For \(t < t^{(1)},\) and \(t > t^{(n)}\) certainty value is 0.
For \( t^\text{(n)} \), \( \text{certainty}(t) = \frac{r-l}{x} \) if \( t^{(r-1)} \leq t < t^{(r)} \).

And for \( t > t^{(1)} \), \( \text{certainty}(t) = \frac{r-l}{x} \) for \( t^{(r-1)} < t < t^{(r)} \).

It may so happen that a pattern is seen twice (or more) in a year. Then obviously the time-periods will not have non-empty intersection. But here the intervals will pile up in two (or more) time-periods within a year. Then the above formula can be applied to each of these periods individually.

It is easy to see that for the periodic patterns of the above type, the union of all the periods together with the certainty function defined above gives rise to a fuzzy interval. So for each periodic pattern we have a fuzzy interval (sometimes more than one fuzzy time intervals) describing the periods of the pattern.

Again for each such periodic pattern keeping the number of \textit{superimposed} intervals is also important. For example for a yearly pattern if the number of \textit{superimposed} intervals is equal to the number of years in the life-span of the item set (the life-span of an item set may not be same as the life-span of the whole dataset) then the periodic pattern is full. But some such periodic patterns may not be full. We may calculate the ratio of the total number of \textit{superimposed} intervals to the total number of years (if we are extracting yearly patterns) in the life-span of the item set and select only those patterns for which this ratio is sufficiently large. We call such patterns as partially periodic patterns. A value of 1 for this ratio will indicate that the pattern is full.
We mentioned earlier that the partially periodic pattern-mining problem was studied in [27], [59]. Although we want to extract similar patterns but our approach is different from earlier approaches. We apply the superimposition of equi-fuzzy intervals, to compute the number of superimposed intervals. And dividing this number with the total number of such intervals in the life-span of the corresponding frequent item set, we can get match ratio, it is required to find the percentage of periodicity.

In general, we may have the following partially periodic patterns viz. i) Yearly partially periodic pattern, ii) Monthly partially periodic pattern and iii) Daily partially periodic pattern.

i) **Yearly partially periodic pattern**: To extract yearly partially periodic pattern, we do not consider year in the time hierarchy and check whether there exists any yearly partially periodic pattern or not. This can be done by computing the number of superimposed intervals associated with an item set then computing the number of years within the life-span of that item set. The ratio of the number of superimposed intervals to the number of years within the life-span of the item set is the match ratio, which is the percentage of the periodicity and its value 1 will represent fully yearly periodic pattern. An example of such pattern is that an item set is frequent in 80% of the summer.

ii) **Monthly partially periodic pattern**: To extract monthly partially periodic pattern, we do not consider month and year in the time hierarchy and check whether there exists any monthly partially
periodic pattern or not. Here the match ratio is computed dividing the total number of *superimposed* intervals associated with an item set with the total number of months within the life-span of that item set.

iii) **Daily partially periodic pattern:** To extract daily partially periodic pattern, we do not consider day, month and year in the time hierarchy and check whether there exists any daily partially periodic pattern or not. Here the match ratio is computed dividing the total number of *superimposed* intervals associated with an item set with the total number of days within the life-span of that item set.

Below we discuss an algorithm for extracting one of such patterns say yearly partially periodic pattern.

### 4.2.3.2 ALGORITHM FOR EXTRACTING PARTIALLY PERIODIC PATTERNS

For each frequent item set having a sufficiently long list of time intervals (a minimum threshold is used) in which the set is locally frequent, we apply the following algorithm to extract periodic (such as yearly, monthly, daily etc.) nature (if any) of the interval list and use the method of set (interval) superimposition to store information about the periods. We use set superimposition if the periods have overlapping of sufficient (a threshold is used) length. In this way (as we have shown above [section-4.2.3.1]), we get fuzzy interval associated with each such periodically frequent item sets. During the
execution of the algorithm a list of superimposed time intervals is kept. Each superimposed interval is a fuzzy interval of the form

\[
[t^{(1)}_1, t^{(2)}_1]^{1/n} [t^{(2)}_2, t^{(3)}_2]^{2/n} [t^{(3)}_3, t^{(4)}_3]^{3/n} \ldots \ldots [t^{(r)}_r, t^{(r+1)}_r]^{r/n} \ldots \ldots \\
[t^{(n)}_n, t^{(1)}_1]^{n-1/n} [t^{(1)}_1, t^{(2)}_2]^{n-2/n} \ldots \ldots [t^{(n-2)}_{n-2}, t^{(n-1)}_{n-1}]^{2/n} [t^{(n-1)}_{n-1}, t^{(n)}_n]^{1/n}
\]

For each such interval we keep the intermediate points \( t^{(1)}, t^{(2)}, t^{(3)}, \ldots, t^{(n)}, t^{(1)}, t^{(2)}, t^{(3)}, \ldots, t^{(n)} \). To check whether a new interval has non-empty intersection of sufficient length with each of the intervals associated with a superimposed interval, we compute the intersection of the new crisp interval with the core of the corresponding fuzzy interval. If this intersection is of sufficient length then this new interval is superimposed on the already maintained superimposed interval and a new fuzzy interval is obtained for the resultant superimposed time interval. Initially the list of superimposed intervals is empty. During the execution of the algorithm a complete pass is made through the time-interval list of an item set. When it moves to a new time interval it checks whether the time interval can be superimposed on any of the superimposed intervals already obtained. If it is then the superimposition process is carried out i.e. the corresponding fuzzy time interval is updated. If it does not superimpose with any of the already obtained superimposed intervals (kept as a list) then this interval is added as a new entry to the list. At the end we study each of the superimposed intervals. For each superimposed interval the number of intervals superimposed is found. If the pattern that we are looking for is yearly patterns then we divide the number of intervals with total number of years in the life-span of the item set. If this ratio is 1 then it is a fully yearly pattern. If the ratio is less than 1 but
sufficiently large (such as 90% or 80%) then we call it a partially periodic (yearly) pattern with the ratio being called as match ratio. In the same way if we are looking for monthly pattern then it is divided by the total number of months in the life-span of the item set. We have already mentioned that for finding yearly pattern we remove the year component from the corresponding dates. For monthly pattern the year and month components are removed. The algorithm assumes that this process is done as a preprocessing step.

For representing each superimposed interval of the form
\[ \left[ t^{(1)}, t^{(2)} \right]^{1/n} \left[ t^{(2)}, t^{(3)} \right]^{2/n} \left[ t^{(3)}, t^{(4)} \right]^{3/n} \ldots \left[ t^{(r)}, t^{(r+1)} \right]^{r/n} \]
\[ \left[ t^{(n)}, t^{(1)} \right]^{1/n} \left[ t^{(1)}, t^{(2)} \right]^{n-1/n} \ldots \left[ t^{(n-2)}, t^{(n-1)} \right]^{2/n} \left[ t^{(n-1)}, t^{(n)} \right]^{1/n} \]

We keep two arrays of real numbers, one for storing the values \( t^{(1)}, t^{(2)}, t^{(3)}, \ldots, t^{(n)} \) and the other for storing the values \( t^{(1)}, t^{(2)}, t^{(3)}, \ldots, t^{(n)} \), each of which is a sorted array. Now if a new interval \([t, t']\) is to be superimposed on this interval we add \( t \) to the first array by finding its position (using binary search) in the first array so that it remains sorted. Similarly \( t' \) is added to the second array. With each superimposed interval the number of intervals superimposed is also kept. If this is known then the value of the certainty function can be easily computed for each piece of time interval of the form \([t^{(i)}, t^{(i+1)}]\).

Data structure used for representing a superimposed interval is

```c
struct superinterval {
    int arsize, count;
    short *l, *r;
};
```

90
Here $arsize$ represents the maximum size of the array used, $count$ represents the number of equi-fuzzy intervals superimposed, and $l$ and $r$ are two pointer pointing to the two associated arrays. During the execution of the algorithm if $count$ exceeds $arsize$ then new memory space is allocated for the arrays pointed to by $l$ and $r$, the old values are copied to the new space and $arsize$ is updated.

**ALGORITHM 4.2**

```plaintext
for each locally frequent item set $s$ do
   $L$ ← list of time intervals associated with $s$
   $Ls$ ← list of superimposed intervals initially set to null
   $lt = L.get();$
   // It is now pointing to the first interval in $L$
   $Ls.append(lt);$  
   while ($(lt = L.get()) != null)$
      $flag = 0;$
      while ($(lst = Ls.get()) != null)$
         if $(\text{compsuperimp}(lt, lst))$
            $flag = 1;$
            if $(flag == 0)$ $Ls.append(lt);$ 
      }
   }

compsuperimp($lt, lst)$
   if ($|\text{intersect}(lst, lt)| > \text{thdl}$)
      superimp($lt, lst);  
      return 1;
   }
   return 0;
}
```

The function $\text{compsuperimp}(lt, lst)$ first computes the intersection of $lt$ with the core of the $lst$ i.e. the time piece (slot) for which the certainty function value is 1. If the intersection length is greater than $\text{thdl}$ (a threshold given by the user) it superimposes $lt$ by calling the function $\text{superimp}(lt, lst)$ which actually carries on the superimposition process by updating the two lists associated as described earlier. The function returns 1 if $lt$ has been superimposed on the $lst$ otherwise returns 0.
Let $n$ be the average size of interval lists maintained with the item sets. Let $m$ be the average number intervals superimposed in one place. For each time interval of an item set a pass is made through the list of superimposed intervals to check whether it can be superimposed on any of the existing superimposed intervals. Here each superimposed interval is a fuzzy set as shown in section-4.2.2. For this the intersection of the core of the fuzzy set and the current time interval is to be computed and this require $O(1)$ time. If the interval superimposes then the time boundaries are to be inserted in the two-sorted arrays maintained for the fuzzy set. Although searching in a sorted array of size $m$ can be done $O(\log m)$ time, inserting it in the current place requires $O(m)$ time. Two such insertions will take $O(2m)$ i.e. $O(m)$ time. Thus for one item set this process will require $O(npm)$ time where $p$ is the average size of the lists of superimposed intervals. Now $p = O(n)$ and $m = O(n)$, thus the overall complexity is $O(n^3)$. This will have to be done for all locally frequent item set.

### 4.2.4 RESULTS OBTAINED

In table below we give some of the periodic patterns extracted from retail dataset $\text{minthd}1= 5$ days, $\text{minthd}2=15$ days and $\text{sigma}=3\%$

<table>
<thead>
<tr>
<th>Levels</th>
<th>Frequent item sets (Periodic Patterns)</th>
<th>Time intervals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level-1</td>
<td>36</td>
<td>[2-1-2000, 3-2-2000]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[5-3-2000, 2-4-2000]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1-5-2000, 7-6-2000]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[4-7-2000, 1-8-2000]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[8-9-2000, 10-10-2000]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[6-11-2000, 12-12-2000]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1-1-2001, 3-2-2001]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2-3-2001, 8-4-2001]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[7-5-2001, 8-6-2001]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[7-7-2001, 5-8-2001]</td>
</tr>
</tbody>
</table>

92
In the above table we see that the item \{170\} is associated with the time intervals [2-1-2000, 15-2-2000], [12-1-2001, 20-2-2001], [4-1-2002, 22-2-2002] which is a yearly pattern. If we remove years from the calendar dates, we get the superimposed interval as follows:

\[
\text{[2nd January, 4th January]}^{(1/3)}\text{[4th January, 12th January]}^{(2/3)}\text{[12th January, 15th February]}^{(1)}\text{[15th February, 20th February]}^{(2/3)}\text{[20th February, 22nd February]}^{(1/3)}
\]

In the following table some of the partially periodic patterns along their match ratios are given.
TABLE-4.2: Partially Periodic Patterns

<table>
<thead>
<tr>
<th>Levels</th>
<th>Partially Periodic Patterns</th>
<th>Time Intervals</th>
<th>Match Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level-1</td>
<td>1344</td>
<td>[3-3-2000, 17-3-2000]</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[5-4-2000, 16-4-2000]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1-5-2000, 20-5-2000]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3-8-2000, 14-8-2000]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1-10-2000, 20-10-2000]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[6-11-2000, 18-11-2000]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[4-1-2001, 15-1-2000]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2-3-2001, 18-3-2001]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[6-8-2001, 16-8-2001]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[5-10-2001, 17-10-2001]</td>
<td></td>
</tr>
<tr>
<td>Level-2</td>
<td>12929</td>
<td>[8-7-2001, 15-8-2001]</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[10-9-2001, 12-10-2001]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[12-12-2001, 16-1-2002]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[26-3-2000, 19-4-2000]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[26-9-2000, 26-10-2000]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[28-10-2000, 3-12-2000]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[10-12-2000, 24-1-2001]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16-5-2001, 19-6-2001]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[28-6-2001, 25-7-2001]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[29-10-2001, 10-12-2001]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[29-12-2001, 14-2-2002]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16-2-2002, 27-3-2002]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[7-4-2002, 6-5-2002]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[4-6-2002, 18-7-2002]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[28-8-2002, 5-10-2002]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[2-2-2003, 19-3-2003]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[27-1-2000, 28-2-2000]</td>
<td>0.555</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[27-8-2000, 31-9-2000]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[3-2-2001, 14-3-2001]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[11-4-2001, 7-5-2001]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[6-6-2001, 11-7-2001]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[5-9-2001, 9-10-2001]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[6-12-2002, 5-1-2003]</td>
<td></td>
</tr>
</tbody>
</table>

4.2.5 CLUSTERING OF PERIODIC PATTERNS

The algorithm [2] discussed in section-3.3 of chapter-III gives all the locally frequent item sets along with the lists of time intervals. Each frequent item set is associated with a list of time intervals where it is frequent. In the previous section, we discussed about the periodicity of the frequent item sets based on their interval lists. We also discussed that while extracting the calendric periodic
frequent patterns, we have fuzzy time intervals describing the periods of the patterns. In this way we can have some periodic patterns associated with fuzzy time intervals describing their periods. In this section we briefly discuss about the clustering of such periodic patterns.

Consider all those periodic patterns having fuzzy time intervals describing their periods and each of these periodic patterns are associated with one fuzzy time interval (sometimes more than one) describing its periods. We want to find clusters among the periodic patterns. Any statistical measure of fuzzy time intervals for example Possibilistic Mean and Variance of fuzzy number [8] (the definitions are given in section-4.2.2) can be used to define the similarity measure for clustering. Since the variance of fuzzy intervals are invariant to shifting [8], two different frequent patterns having the same value of variance for the fuzzy intervals describing their periods can be considered to be similar and hence they may belong to same cluster. This can be used to find clusters among the patterns. After defining the similarity measure any standard algorithm developed for clustering can be used to find the clusters among the periodic patterns.

4.3 SEQUENTIAL PATTERNS

4.3.1 INTRODUCTION

The algorithm [2] described in section-3.3 of chapter-III extracts for each frequent item set a sequence of time intervals in which the set is frequent and we have called such frequent set as locally frequent set. In section-4.2 of this chapter, we have done intra-item set study i.e. we tried to find patterns within the time
intervals for a single such item set. However there may exist some patterns in the time intervals of different item sets. In this chapter we are going to devise a method to extract patterns among different item sets based on their interval lists. Thus the study is completely inter-item set study. For example the time intervals of two different item sets may be in some sequence. This is one type of sequence mining.

The sequence-mining problem is extensively studied in [23], [37], [38], [46], and [51]). It is a topic in its own right and many application domains such as DNA sequence, signal processing, and speech analysis require mining of sequence data even though there is no explicit temporality in the data. Discovering sequential patterns from a large dataset of sequences has been recognized as an important problem in the field of knowledge discovery and data mining. Here the problem is to mine causal relations between events. An event is a non-empty, disordered collection of items. The sequence mining problem is discussed briefly in section-2.7.2.1 of chapter-II.

In this section, we are going to propose a method of sequence mining in a temporal dataset. In our case the events are replaced by item sets, which are locally frequent. Thus we define sequence mining problem as causal relation between the locally frequent sets and our proposed algorithm takes as input the output of algorithm-3.1 proposed in chapter-III. As we mentioned that the algorithm-3.1 gives all the locally frequent item sets along with the list of time intervals where the set are frequent which are our sequences of size one or 1-sequences, these are the frequent sequences of level-1. In the interval-lists of
locally frequent sets there may exist *sublists*, which may follow some pattern. For example, the first interval in *sublist* of item set say $A$ starts before the starting of the first interval of *sublist* of another item set say $B$, the second interval in the *sublist* of the $A$ starts before the starting of the second interval of *sublist* of $B$ and so on. Using this we go on enlarging the size of the sequences using bottom up approach. So the algorithm is basically a level-wise algorithm. Since the sequential patterns we have studied involves only the starting time-stamps of the time intervals associated with the item sets, from now onwards instead of list of time intervals we shall deal with list of time-stamps where the time-stamps are starting time-stamps of time intervals associated with some item set.

### 4.3.2 DEFINITIONS AND NOTATIONS

Let $F_1 = \{A_1, A_2, \ldots, A_m\}$ be a set of locally frequent item sets obtained by using algorithm-3.1 discussed in chapter-III, where each $A_i$ has an associated list of time-stamps.

**DEFINITION 1: SEQUENCE**

We define a sequence of frequent sets as $s = (A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_p)$ where each $A_i$ is locally frequent item set obtained by the method discussed in chapter-III. We call a sequence of size-$p$ or $p$-sequence if the length of the sequence is $p$. So obviously the sequences of size-$1$ are nothing but the locally frequent sets themselves.
DEFINITION 2: SUBSEQUENCE

We define a subsequence of a sequence having size less than the size of the sequence and having same ordering. So deleting some item sets from a sequence can form a subsequence of the sequence.

DEFINITION 3: FREQUENCY OF A SEQUENCE

Let $A_i$ and $A_j$ be two locally frequent sets. From the sequence of time-stamps associated with the two item sets $A_i$ and $A_j$, we extract two subsequences of time stamps $LA_i$ and $LA_j$ of same size satisfying the following conditions:

Let $s_{tk}[LA_i]$ denote the time-stamp of the $k$-th interval in the subsequence $LA_i$, $i,k$ being positive integers.

i) $s_{tk}[LA_i] \leq s_{tk}[LA_j]$ for $k=1,2,...,l$ where $l$ is the length of $LA_i$,

ii) $s_{tk+l}[LA_i] \geq s_{tk}[LA_j]$ for $k=1,2,...,l-1$ and

iii) $LA_i$ and $LA_j$ are maximal sequences satisfying i) and ii)

Then frequency of $A_i \rightarrow A_j$ is defined as the ratio between the $Supp(A_i \rightarrow A_j)$ to the $Supp(A_i)$, where

$Supp(A_i \rightarrow A_j)$ = number of time-stamps $s_{tk}[LA_i]$ satisfying the conditions i), ii) and

iii)

$Supp(A_i)$ = number of time-stamps associated with $A_i$.

Thus the frequency of $(A_i \rightarrow A_j)$ is given by
In general, we define the frequency of a sequence $A_1 \rightarrow A_2 \rightarrow A_3 \ldots \rightarrow A_p$ as follows:

Let $\{LA_i\}_{i=1,2,\ldots,p}$ be the subsequences of time-stamps of same size extracted from the sequence of time-stamps associated with the locally frequent sets $\{A_i\}_{i=1,2,\ldots,p}$ that satisfies the following conditions:

Let $st_k[LA_i]$ denote the time-stamp of the $k$-th interval $LA_i$ $k$ being a positive integer and $i=1,2,\ldots,p$.

1. $st_k[LA_1] \leq st_k[LA_2] \leq \ldots \leq st_k[LA_p]$ for $k=1,2,\ldots,l$ where $l$ is the length of $LA_1$

2. $st_{k+1}[LA_1] \geq st_k[LA_p]$ for $k=1,2,\ldots,l-1$ and

3. $\{LA_i\}_{i=1,2,\ldots,p}$ are maximal sequences satisfying [1] and [2]

Then frequency of $A_1 \rightarrow A_2 \rightarrow A_3 \ldots \rightarrow A_p$ is defined as the ratio between the $Supp(A_1 \rightarrow A_2 \rightarrow A_3 \ldots \rightarrow A_p)$ to the $Supp(A_1)$, where

$Supp(A_1 \rightarrow A_2 \rightarrow A_3 \ldots \rightarrow A_p) = \text{length of } LA_1 \text{ satisfying the conditions [1] and [2]}$

$Supp(A_1) = \text{length of the sequence of time-stamps i.e. the number of time-stamps associated with } A_1$.

Thus $fr(A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \ldots \rightarrow A_p) = \frac{Supp(A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \ldots \rightarrow A_p)}{Supp(A_1)}$
DEFINITION 4: FREQUENT SEQUENCE

A sequence of item sets is said to be frequent if its frequency exceeds some user-specified threshold $\sigma$.

DEFINITION 5: MAXIMAL FREQUENT SEQUENCE

A sequence $s = (A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \ldots \rightarrow A_p)$ of item sets is said to be maximal sequence if it cannot be enlarged or it is not a subsequence of any other frequent sequence.

THEOREM 1:

A sequence $s = (A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \ldots \rightarrow A_n)$ of item sets is frequent then any of its subsequence of size-$(n-1)$ starting with $A_1$ is frequent.

Proof: Suppose that sequence $s = (A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \ldots \rightarrow A_n)$ of item sets is frequent.

$$r(A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \ldots \rightarrow A_n) \geq \sigma,$$

for a given $\sigma$, $0 \leq \sigma \leq 1$

$$\Rightarrow \frac{\text{Supp}(A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow \ldots \rightarrow A_n)}{\text{Supp}(A_i)} \geq \sigma \quad \text{......} \quad (1)$$

Now according to the definition of $\text{Supp}(A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_n)$, we can extract $n$ subsequences of time-stamps say $LA_1$, $LA_2$, ..., $LA_n$ of equal length say $l$ where $LA_i$ denotes the list time-stamps associated with the item set $A_i$ for $i=1,2,...,n$ where these are subsequences of the lists of starting time-stamps associated with the item sets $A_1, A_2, ..., A_n$ respectively. The subsequences satisfy the following three conditions:

(i) $\text{st}_k[LA_i] \leq \text{st}_k[LA_{i+1}]$ for $i=1,2,...,n-1$ and $k=1,2,...,l$. 

100
\[(ii) \quad st_{k+1}[L_{A_i}] \geq st_{k+1}[L_{A_j}] \quad \text{for} \quad i=1,2,\ldots,n-1 \quad \text{and} \quad k=1,2,\ldots,l-1\]

\[(iii) \quad \text{the sequences are maximal sequences satisfying (i) and (ii)}\]

where \(st_k[L_{A_i}]\) denotes the \(k\)-th time stamp of the list \(L_{A_i}\).

Now let us consider any subsequences \(B_1\rightarrow B_2 \rightarrow \ldots \rightarrow B_m\) of \(A_1\rightarrow A_2 \rightarrow A_3 \rightarrow \ldots \rightarrow A_n\) starting with \(A\) i.e. \(B_1=A_1\).

Now since each \(B_i\) for \(i=2,3,\ldots m\) is some \(A_j\) for \(j=2,3,\ldots n\), we associate with each \(B_i\) a list of time-stamps say \(L_{B_i}\) where \(L_{B_i}\) is \(L_{A_j}\) if \(B_i\) is \(A_j\). Now it follows from above that the \(B_1\rightarrow B_2 \rightarrow \ldots \rightarrow B_m\) satisfy the following two conditions:

\[(i) \quad st_k[L_{B_i}] \leq st_k[L_{B_{i+1}}] \quad \text{for} \quad i=1,2,\ldots,m-1 \quad \text{and} \quad k=1,2,\ldots,l\]

\[(ii) \quad st_{k+1}[L_{B_i}] \geq st_k[L_{B_{m}}] \quad \text{for} \quad i=1,2,\ldots,m-1 \quad \text{and} \quad k=1,2,\ldots,l-1\]

This implies that \(\text{Supp}(B_1\rightarrow B_2 \rightarrow B_3 \ldots \rightarrow B_{n-1}) \geq \text{Supp}(A_1\rightarrow A_2 \rightarrow A_3 \ldots \rightarrow A_n)\)

\[\Rightarrow \frac{\text{Supp}(B_1\rightarrow B_2 \rightarrow \ldots \rightarrow B_{n-1})}{\text{Supp}(B_1)} \geq \frac{\text{Supp}(A_1\rightarrow A_2 \rightarrow \ldots \rightarrow A_n)}{\text{Supp}(A_1)}\]

\[\geq \sigma \quad [\text{using (1)}]\]

since \(\text{Supp}(A_1) = \text{Supp}(B_1)\)

\[\Rightarrow fr(B_1\rightarrow B_2 \rightarrow B_3 \ldots \rightarrow B_{n-1}) \geq \sigma\]

\[\Rightarrow s' = (B_1\rightarrow B_2 \rightarrow B_3 \ldots \rightarrow B_{n-1})\] is a frequent sequence of item sets.

Hence every subsequence of a frequent sequence having same starting item set is frequent.
4.3.3 EXTRACTING SEQUENTIAL PATTERNS

4.3.3.1 ALGORITHM PROPOSED

The algorithm proposed for the above sequence mining problem is a level-wise algorithm. In the first level, we will consider all the locally frequent item sets generated by the algorithm [2] discussed in section-3.3 of chapter-III and these are our frequent 1-sequences and we denote the set by \( F_1 \). Each frequent set in \( F_1 \) is associated with a list of time-stamps. Considering all possible pairs of disjoint frequent sets from the 1-sequences we generate a set of candidate 2-sequences. Then the frequency of every candidate is computed going through the list of time-stamps maintained for frequent 1-sequences and using the \textit{Definition} of section-4.3.2. Those candidates having frequency greater than the \textit{min_threshold} \( \sigma \) are kept leaving others. These will be the set of frequent sequence in the second level we denote these as \( F_2 \). From \( F_2 \), we can generate candidates for third level as follows: If \( A_1 \rightarrow A_2 \) and \( A_1 \rightarrow A_3 \) are two members of \( F_2 \) then the corresponding candidates for third level will be \( A_1 \rightarrow A_2 \rightarrow A_3 \) and \( A_1 \rightarrow A_3 \rightarrow A_2 \). Then the frequency of each candidate is computed and compared with \textit{min_threshold} \( \sigma \) to get \( F_3 \). From \( F_3 \) we can get candidates for fourth level as follows: If \( A_1 \rightarrow A_2 \rightarrow A_3 \) and \( A_1 \rightarrow A_2 \rightarrow A_4 \) are any two frequent 3-sequences, then the corresponding candidates for fourth level are \( A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow A_4 \) and \( A_1 \rightarrow A_2 \rightarrow A_4 \rightarrow A_3 \). Before going to compute the frequency of each candidate we must ensure that all its subsequences of size one less than it having same starting item set are frequent. For example, for the sequence \( A_1 \rightarrow A_2 \rightarrow A_3 \rightarrow A_4 \) to be frequent, the 3-sequences...
\( A_1 \rightarrow A_2 \rightarrow A_3, A_1 \rightarrow A_2 \rightarrow A_4 \) and \( A_1 \rightarrow A_3 \rightarrow A_4 \) must belong to \( F_3 \). Thus pruning step actually begins at fourth level.

In general, from \( i \)-th level frequent sequences we can generate \((i+1)\)-th level candidates in the following way: For any two frequent \( i \)-sequences \( s_1 = (A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_i) \) and \( s_2 = (B_1 \rightarrow B_2 \rightarrow \ldots \rightarrow B_i) \) such that \( A_1 = B_1, A_2 = B_2, \ldots, A_{i-1} = B_{i-1} \) and \( A_i \neq B_i \) \((A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_i \rightarrow B_i) \) and \((A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow B_i \rightarrow A_i) \) will be two candidates for \((i+1)\)-th level. In this way we generate all the candidates. Then the set of candidates is pruned. Then the frequency of each of the remaining candidates is computed and compared with \textit{min_threshold} \( \sigma \) to get \( F_{i+1} \). The process continues till no candidate is generated or a particular level becomes empty. The algorithm for sequence mining is given below.

**ALGORITHM 4.3**

\textit{Algorithm to find frequent 2-sequences of the type \( A_2 \) is followed by \( A_i \) i.e \( A_1 \rightarrow A_2 \)}

\begin{verbatim}
\{\( LA1 = \) list of time-stamps associated with the item set \( A_1 \),
\( LA2 = \) list of time-stamps associated with the item set \( A_2 \),
\( LIA1 = LIA2 = \) null; \( count = 0 \);
\( lpA1 = LA1.get() \); // \( lpA1 \) will now point to the first node of \( LA1 \).
\( lpA2 = LA2.get() \); // \( lpA2 \) will now point to the first node of \( LA2 \).
while(\( lpA1 != \) null \&\& \( lpA2 != \) null) do
\{ \( t1 = \) lpA1->ts();
\( t2 = \) lpA2->ts();
\( if(t1 < t2)\)
\{ \( if((temp = \) lpA1.get())!=null)
\{ \( if(temp->ts() >= t2)\)
\{ count++; \( lpA1 = temp;\)
\( lpA2 = \) LIA2.get();
\( LIA1.append(t1); LIA2.append(t2);\)
\} \}
else \( \{lpA1 = temp; continue;\}\) \}
else \( \{count++; lpA1 = temp; LIA1.append(t1); LIA2.append(t2); continue;\}\) \}
\}
\( lpA2 = LA2.get() ;\)
\}
\}
\end{verbatim}

103
This algorithm will give the value of the $\text{Supp}(A_1 \rightarrow A_2)$ and it is divided by $\text{Supp}(A_1)$ to get $f_r(A_1 \rightarrow A_2)$ which is required to find whether a sequence $(A_1 \rightarrow A_2)$ is frequent or not.

The algorithm also produces two lists of time-stamps $llA1$ and $llA2$ of length $count$ from the lists $lA1$ and $lA2$ such that $(llA1)_i <= (llA2)_i$, for $i = 1, 2, \ldots count$, where $(llA1)_i$ denotes the $i$-th time-stamp of $lA$ and similarly $(llA2)_i$ has the same meaning and $(llA1)_{i+1} > (llA2)_i$ for $i = 1, 2, \ldots count - 1$.

**ALGORITHM TO FIND SUPPORT OF SEQUENCES OF ANY LENGTH**

Associated with each frequent sequence of the form $A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_k$ we have $k$ number of lists of time-stamps satisfying [1], [2] and [3] of Definition 3 of section-4.3.2. While implementing the level-wise procedure, we maintain the two lists associated with the two extreme item sets $A_1$ and $A_k$. These two lists are needed for counting supports of candidate sequences in the next level, which are formed by joining this sequence with some other sequence of the same size.

**ALGORITHM 4.4**

Algorithm to compute support of $(k+1)$-sequences of the form $A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_k$, obtained by joining the two sequences $A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_{k-1} \rightarrow A_k$ and $A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_{k-1} \rightarrow A_k$ with $A_{k+1}$ being $A_k$.

- $lA1$—list of time-stamps associated with $A_1$ for the $k$-sequence $A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_k$
- $lAk$—list of time-stamps associated with $A_k$ for the $k$-sequence $A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_k$
- $lAkd$—list of time-stamps associated with the 1-sequence $A_k$

$\text{count} = 0$

$tl = lA1.getO(); tk = lAk.getO(); tkd = lAkd.getO();$

$\text{count} = 0$

$tl = lA1.getO(); tk = lAk.getO(); tkd = lAkd.getO();$

$\text{while}(t1 != null) && (tk != null) && (tkd != null)) do$

{if ($tk->ts() < tkd->ts()$)$
{count++; $lA1n.append(tl->ts());$
$\text{temp} = lAk.getO();$
$tl \text{~temp;}$

{if ($\text{temp} > tkd->ts()$)$
{count++; $lA1n.append(tl->ts());$
$\text{temp} = lAk.getO();$
$tl \text{~temp;}$

$\text{while}(t1 != null) && (tk != null) && (tkd != null)) do$

{if ($tk->ts() < tkd->ts()$)$
{count++; $lA1n.append(tl->ts());$
$\text{temp} = lAk.getO();$
$tl \text{~temp;}$
The function \( ts() \) returns the time-stamp of the corresponding node in the list.

**ALGORITHM 4.5**

\( F_2 \) is the set of frequent 2-sequence obtained using the Algorithm 4.3.

\( k = 3 \)

\[
\text{do while } F_{k-1} \neq \emptyset \\
F_k := \emptyset \\
C_k := \text{gen_candidate_sequences}(F_{k-1}) \\
\text{prune}(C_k) \\
\text{for all candidates } s_k \text{ in } C_k \text{ // } s_k \text{ is any sequence of size-}k \\
\text{if } \frac{\text{fr}(s_k)}{\text{fr}(s_k) \text{ is computed using the Algorithm 4.4}} > \sigma \}
\]

\[
F_k := F_k \cup \{ s_k \}
\]

\( k = k + 1 \)

\end do

**Answer:** \( \cup F_k \)

Candidate sequence generation with the given \( F_{k-1} \)

\[
\text{gen_candidate_sequences}(F_{k-1})
\]

\( C_k := \emptyset \\
\text{for all } (k-1)-\text{sequences } s_{k-1} \in F_{k-1} \\
\text{for all } (k-1)-\text{sequences } s'_{k-1} \in F_{k-1} \\
[ s_{k-1}[1] = s'_{k-1}[1], s_{k-1}[2] = s'_{k-1}[2], \ldots, s_{k-1}[k-2] = s'_{k-1}[k-2], s_{k-1}[k-1] = s'_{k-1}[k-1] ] \\
\text{then } s_k = (s_{k-1}[1] \rightarrow s_{k-1}[2] \rightarrow \ldots \rightarrow s_{k-1}[k-2] \rightarrow s_{k-1}[k-1] \rightarrow s_{k-1}[k]) \\
\text{and } s'_k = (s'_{k-1}[1] \rightarrow s'_{k-1}[2] \rightarrow \ldots \rightarrow s'_{k-1}[k-2] \rightarrow s'_{k-1}[k-1] \rightarrow s'_{k-1}[k]) \\
C_k := C_k \cup \{ s_k, s'_k \}
\]

**Pruning**

\[
\text{prune}(C_k)
\]

\( \text{for all } s_k \in C_k \\
\text{for all } (k-1)-\text{subsequences } s_{k-1} \text{ of } s_k \text{ having same starting item set} \\
\text{do} \\
\text{if } s_{k-1} \notin F_{k-1} \\
\text{then } C_k := C_k \setminus \{ s_k \}
\]

The algorithm gives all the frequent sequences along with the extracted lists of time-stamps.
4.3.4 RESULTS OBTAINED

Some of the sequential patterns obtained from the table-4.1 are given below:

<table>
<thead>
<tr>
<th>Sequences</th>
<th>Extracted list of time-stamps</th>
<th>Support value</th>
</tr>
</thead>
</table>

TABLE 4.3: Frequent sequences and associated time-stamps

4.4 CONCLUSION

In this chapter, we mainly focused on extracting patterns from the locally frequent item sets based on the associated interval-lists obtained by the method discussed in the previous chapter. First part of the chapter deals with different types of periodic patterns that may exist in the underlying dataset. In this part first of all we discussed about a method extracting periodic patterns based on the time
gap between any two consecutive time intervals and the length of each of the intervals in the interval-list associated with a locally frequent item set. Next we discussed about different types periodic patterns and partially periodic patterns that may exist if the time-stamps are taken as calendar-dates. We propose here an algorithm, which successfully uses a set operation called set superimposition to find match ratio, which is required to check whether a pattern is periodic or partially periodic. Again we propose a method of storing time intervals based on set superimposition, which turns out to be fuzzy time interval if the intervals have sufficient overlapping. In the next section we discussed about clustering of patterns based on their fuzzy time intervals using some statistical measure.

In the last part of this chapter we discussed about a method of finding sequential patterns based on their associated lists of time intervals. We define support of a sequence in terms of their lists of time intervals and propose an algorithm to extracts frequent sequences. The algorithm is a level-wise algorithm. We also report some of the experimental results in tabular form.