CHAPTER VI

PERIODICITY DETECTION

6.1 INTRODUCTION

In a time-series containing closing stock prices of a particular company, stock price peaks may occur during certain time-intervals. Similarly, in a time-series containing sales records, the phenomenon of panic reversal of sales may be observed during certain time-intervals. Events such as a volcanic eruption, tropical storms across a particular region etc also occur during a time-interval, then stop for sometime, again occur during another time-interval and so on. Given the sequence of time-intervals in which a pattern/event like this has occurred and assuming that the pattern/event under study has hierarchical timestamps, a method is proposed in this chapter to detect partial and total periodicities of the pattern/event at different levels of the timestamp hierarchy. For example, if S is the sequence of time-intervals in which a particular pattern/event X has occurred and if the timestamps associated with X are in year/month/day format, then using the method proposed in this chapter, it is possible to detect partial as well as total, yearly and monthly periodicities of X.

In Section 6.2, some important definitions relevant to this chapter are introduced. A precise definition of the problems addressed in this chapter is given in Section 6.3. In Section 6.4, algorithms are presented to detect periodicities of a pattern/event that occurs in certain time-intervals. Experimental results are reported in Section 6.5. Section 6.6 summarizes the contributions made in this chapter.

6.2 PRELIMINARIES

DEFINITION 6.1 *Time-span of a sequence of time-intervals.* If t_{min} and t_{max} are respectively the smallest and the largest timestamps in a sequence of time-intervals, then the domain $[t_{min}, t_{max}]$ is defined to be the time-span of that sequence of time-intervals. TS(Q) is used to denote the time-span of a sequence Q of time-intervals.

DEFINITION 6.2 *Calendar schema.* A calendar schema $(c_1, c_2, ..c_n)$ is a set of n calendar units (e.g. year, month, day, hour etc) arranged in a hierarchy of n levels , viz. level 1, level 2, ...,level n. The calendar unit c_1 is at level 1, c_2 is at level 2 and so on. Note that $n > 1$. In a calendar schema, level 1 is considered to be the highest level and level n is taken to be the lowest level. A single calendar unit at a level k in a calendar schema consists of an integral number m_p of calendar units at level p, for $p > k$. It may be noted that the value of m_p is not always fixed. For example, a calendar schema can be (year, month, day) – the calendar unit year is at level 1, month is at level 2 and day is at level 3. It may be further noted that one year contains 12 months; one year contains 365 or 366 days and one month contains either 28 or 29 or 30 or 31 days.

DEFINITION 6.3 *Hierarchical timestamp.* A hierarchical timestamp in a n-level calendar schema S is given by $t_1/t_2/.../t_n$, where t_i (an integer) represents the value of the calendar unit at level i in S. For example, a hierarchical timestamp in the calendar schema (year, month, day) can be 2000/2/21. In this case, $t_1 = 2000$, $t_2 = 2$ and $t_3 = 21$. For a hierarchical timestamp in a calendar schema, t_i for $i > 1$ and $i \leq n$ will lie in the range $[p_i, q_i]$, where p_i and q_i are integers whose values may depend on $\{t_1, t_2,..t_{i-1}\}$. For example, for any hierarchical timestamp in the calendar schema (year, month, day), the range of t_2 is [1, 12], whereas the range of t_3 is either [1, 28] or [1, 29] or [1, 30] or [1, 31]. Similarly, for any hierarchical timestamp in the calendar schema (hour, minute, second), the ranges of both t_2 and t_3 are [0, 59]. *In this chapter, the terms hierarchical timestamp and timestamp are used interchangeably.*

Please note: A hierarchical timestamp in a n-level calendar schema S may be specified by permuting the different levels of S For example, a hierarchical timestamp 2000/3/15 in the calendar schema (year, month, day) may be specified as 3/15/2000 in month/day/year format. It is trivial to translate a timestamp in a given format into another format. ***Throughout this chapter, it is assumed that hierarchical timestamps are specified only as defined in Definition 6.3.***

DEFINITION 6.4 *Period.* In a n-level calendar schema S, a period is defined to be a single calendar unit at some level k, for k < n. For example, in a calendar schema (year, month, day), a period can be either 1 year or 1 month.

Please note: The above definition (Definition 6 4) can be generalized further to also accommodate other types of periods, such as bi-monthly period, quarterly period, half- yearly period etc

DEFINITION 6.5 *Period instance* A period instance of a period defined at level k in a calendar schema S is a set of hierarchical timestamps in S. Each hierarchical timestamp in a period instance has the same value for the calendar unit at the i^{th} level in S, for $i \leq k$. However, the hierarchical timestamps in a period instance span the entire range of values of the calendar unit at the i^{th} level in S for $i > k$. Clearly then, a period instance is a time-interval of length equal to the length of the period. For example, for a yearly period in the calendar schema (year, month, day), period instances can be [2000/1/1, 2000/12/31], [1995/1/1, 1995/12/31] etc. Similarly, for a monthly period in the same calendar schema, period instances can be [2012/1/1, 2012/1/31], [2012/2/1, 2012/2/29], [1998/4/1, 1998/4/30] etc.

DEFINITION 6.6 *Stripped timestamp.* For a period defined at a level k in a calendar schema S, a hierarchical timestamp h in S can be mapped to a stripped timestamp strip(h). The stripped timestamp strip(h) is obtained by removing from h, the value of the calendar units at level 1, level 2,…,level k. For example, if a yearly period is considered, then the hierarchical timestamp 2008/2/20 in the calendar schema (year, month, day) maps to the stripped timestamp 2/20. Similarly, if a monthly period is considered, then the hierarchical timestamp 2009/3/27 in the same calendar schema maps to the stripped timestamp 27.

Please note: For a period defined with respect to a calendar schema, a stripped timestamp x may not have a corresponding hierarchical timestamp t with strip(t) = x in every period instance of the period. For example, if a yearly period is considered in the calendar schema (year, month, day), then the stripped timestamp 2/29 does not have a corresponding hierarchical timestamp t with strip(t) = 2/29 in the period instance [2001/1/1, 2001/12/31]. Similarly, if a monthly period is considered in the same schema, then the stripped timestamp 30 does not have a corresponding hierarchical timestamp t with strip(t) = 30 in the period instance [2011/2/1, 2011/2/28].

For a period defined with respect to a calendar schema S, the notion of certainty of a pattern/event (Definition 6.8) at a stripped timestamp x will be meaningful only if x has a corresponding hierarchical timestamp t with strip(t) = x in every period instance that intersects TS(Q). Here, Q is the sequence of time-intervals (with timestamps in S) in which the pattern/event has occurred. Hence, the following definition is introduced –

DEFINITION 6.7 *Common stripped timestamp domain (CSTD).* Given a period defined with respect to a calendar schema S and a sequence Q of time-intervals with timestamps in S, the common stripped timestamp domain (CSTD) comprises of every stripped timestamp x that has a corresponding hierarchical timestamp t with strip(t) = x in every period instance that intersects TS(Q). For example, if TS(Q) = [2009/5/12, 2012/11/30] and a yearly period is considered in the calendar schema (year, month, day), then the period instances that intersect TS(Q) are as follows – [2009/1/1, 2009/12/31], [2010/1/1, 2010/12/31], [2011/1/1, 2011/12/31] and [2012/1/1, 2012/12/31]. In this case, CSTD = [1/1, 2/28] $\cup$ [3/1, 12/31]. Similarly, if TS(Q) = [2000/2/18, 2000/10/25] and a monthly period is considered in the same calendar schema, then the period instances that intersect TS(Q) are [2000/2/1, 2000/2/29], [2000/3/1, 2000/3/31], [2000/4/1, 2000/4/30], [2000/5/1, 2000/5/31], [2000/6/1, 2000/6/30], [2000/7/1, 2000/7/31], [2000/8/1, 2000/8/31], [2000/9/1, 2000/9/30] and [2000/10/1, 2000/10/31]. In this case, CSTD = [1, 29].

DEFINITION 6.8 *Certainty of a pattern/event at a stripped timestamp in CSTD.* Given a period defined with respect to a calendar schema S and a sequence Q of time-intervals (with timestamps in S) in which a particular pattern/event has occurred, a certainty function (cert()) is defined on CSTD, whose value at a stripped timestamp x in CSTD is given by

$$cert(x) = |\{ t \mid \text{the pattern/event occurs at } t \text{ and } strip(t) = x\}| \,/\, np$$

Here, np is the total number of period instances that intersect TS(Q). Now, in every period instance that intersects TS(Q), there is at most one timestamp t such that the pattern/event occurs at t and strip(t) = x. Also, if a period instance does not intersect TS(Q), then there is no timestamp t in that period instance where the pattern/event occurs. Hence clearly, $|\{ t \mid$ the pattern/event occurs at t and strip(t) = x$\}| \leq$ np and so, $0 \leq cert(x) \leq 1$. ***It may be noted that cert(x) = 1 indicates a full periodicity of the***

pattern/event at x, whereas 0 < cert(x) < 1 yields a partial periodicity at x. If the value of cert(x) is less than 1 but greater than or equal to a user-defined threshold, then the pattern/event is said to be almost fully periodic at x. The certainty of the pattern/event at a stripped timestamp x in CSTD is defined to be cert(x).

Example 6.1 Suppose an event occurs during the following time-intervals [2000/4/24, 2000/4/30], [2000/8/14, 2000/8/18], [2001/8/10, 2001/8/20], [2002/8/15, 2002/8/25] and [2003/8/2, 2003/8/22]. All timestamps are in year/month/day format. Determine the certainty of the event on 18th August and 25th April.

Solution: Let Q represent the sequence of time-intervals in which the event has occurred. Now, TS(Q) = [2000/4/24, 2003/8/22]. The calendar schema and period under consideration here are (year, month, day) and one year respectively. As such, there are four period instances that intersect TS(Q) – viz. [2000/1/1, 2000/12/31], [2001/1/1, 2001/12/31], [2002/1/1, 2002/12/31] and [2003/1/1, 2003/12/31]. Since the event occurs on four timestamps – viz. 2000/8/18, 2001/8/18, 2002/8/18 and 2003/8/18 that map to the stripped timestamp 8/18 (i.e. 18th August),

$$\text{cert}(8/18) = \frac{4}{4} = 1$$

This implies that in a span of four years, the event occurs on 18th August in every year. In other words, the event is fully periodic on 18th August.

Next, the certainty of the event on 25th April is computed. Since the event occurs on only one timestamp – viz. 2000/4/25 which maps to the stripped timestamp 4/25 (i.e. 25th April),

$$\text{cert}(4/25) = \tfrac{1}{4} = 0.25$$

This implies that in a span of four years, the event occurs on 25th April in only one year i.e. the event is partially periodic on 25th April.

6.3 PROBLEM STATEMENT

Given a period defined with respect to a calendar schema S and a sequence Q of time-intervals (with timestamps in S) in which a particular pattern/event has occurred, it is required

(i) to determine cert(x) at a given $x \in CSTD$ and

(ii) to find the local maxima of the certainty function in CSTD

Significance of the proposed work: From the value of cert(x) at a given $x \in CSTD$, it can be ascertained whether the pattern/event under study is fully or partially periodic at x. The value of cert(x) can also indicate whether the pattern/event is almost fully periodic at x. The determination of local maxima of the certainty function in CSTD helps in detecting the regions viz. stripped timestamps in CSTD at which the pattern/event is either fully or almost fully periodic.

6.4 MINING PERIODICITIES OF A PATTERN/EVENT

Previously in [MMB08], a $O(n^3)$ method was proposed for mining periodicities of a pattern/event that occurs in certain time-intervals. In this section, a much more time-efficient method for the same problem is presented. Certain preprocessing steps are described in Section 6.4.1. In Section 6.4.2, an alternative formula for cert(x) at a given

$x \in CSTD$ is established. This formula is used by the algorithms presented in Section 6.4.3 to detect periodicities of a pattern/event which occurs in certain time-intervals.

6.4.1 Preprocessing steps

Given a period defined with respect to a calendar schema S and a sequence Q of time-intervals (with timestamps in S) in which a particular pattern/event has occurred, the following preprocessing steps need to be done to be able to mine periodicities of the pattern/event:

(a) TS(Q) and the total number of period instances that intersect TS(Q) are determined.

(b) If there are two overlapping time-intervals in Q, then they are replaced by their union. For example, suppose the calendar schema S is (year, month, day). Now, if [2009/3/10, 2009/3/20] and [2009/3/18, 2009/3/25] are two time-intervals in Q, then they will be replaced by [2009/3/10, 2009/3/25]. *This preprocessing step is repeated as many times as required till a sequence Q' of disjoint time-intervals is obtained from Q.*

(c) If a time-interval in Q′ is not contained within a single period instance, then it is broken up and replaced by a set of disjoint time-intervals that satisfy this property. For example, suppose a monthly period is considered in the calendar schema (year, month, day). Now, if the time-interval [2012/1/21, 2012/3/14] is in Q′, then it can be replaced by [2012/1/21, 2012/1/31], [2012/2/1, 2012/2/29] and [2012/3/1, 2012/3/14]. Similarly, if a yearly period is considered in the same calendar schema, then the time-interval [2000/10/18, 2001/4/20] can be replaced

by [2000/10/18, 2000/12/31] and [2001/1/1, 2001/4/20]. *This preprocessing step is repeated as many times as required till from Q', a sequence Q'' is obtained in which every time-interval is contained within a single period instance. It may be noted that the time-intervals in Q'' are also disjoint.*

Please note: TS(Q) = TS(Q') = TS(Q'') and hence CSTD is the same for Q, Q' and Q'' Also, the set of timestamps at which the pattern/event occurs remains unchanged by the fore-mentioned preprocessing steps As such, for any $x \in$ CSTD, the value of cert(x) with respect to Q, Q' and Q'' is the same

DEFINITION 6.9 *Stripped time-interval.* Given a period defined with respect to a calendar schema S and a sequence Q of time-intervals with timestamps in S, a set stripintv(I) of stripped timestamps can be associated with a time-interval I in Q, provided I is contained within a single period instance.

$$\text{stripintv}(I) = \{\{\text{strip}(h) \mid h \text{ is a timestamp in } I\} \cap \text{CSTD}\}$$

Because I is contained in a single period instance, stripintv(I) is either empty or it is a time-interval. In the latter case, stripintv(I) is called the stripped time-interval of I. For example, if a monthly period in the calendar schema (year, month, day) and the following sequence of time-intervals – [2004/2/15, 2004/2/29], [2003/12/29, 2003/12/31] and [2005/10/20, 2005/10/31] are considered, then CSTD is [1, 28]. As such, the stripped time-interval of [2004/2/15, 2004/2/29] is [15, 28] and that of [2005/10/20, 2005/10/31] is [20, 28]. It may be noted that stripintv([2003/12/29, 2003/12/31]) = $\emptyset$ i.e. [2003/12/29, 2003/12/31] does not have a corresponding stripped time-interval.

(*d*) From Q″, the sequence Q_s of stripped time-intervals is constructed by taking stripintv(I) for every I in Q″ for which stripintv(I) ≠ ∅. In the context of the fore-mentioned example in Definition 6.9, if the given sequence of time-intervals – viz. [2004/2/15, 2004/2/29], [2003/12/29, 2003/12/31] and [2005/10/20, 2005/10/31] is considered to be Q″, then Q_s will consist of the following stripped time-intervals – viz. [15, 28] and [20, 28].

Please note: **Though the time-intervals in Q″ are disjoint, as shown in the above example, the stripped time-intervals in Q_s may not be disjoint.**

6.4.2 An alternative formula for cert(x), x ∈ CSTD

For a period defined with respect to a calendar schema S and a sequence Q of time-intervals (with timestamps in S) in which a particular pattern/event has occurred, an alternative formula for cert(x) at a given x∈CSTD is now established. In the following theorem, it is assumed that the preprocessing steps (a), b), c) and d)) described in Section 6.4.1 have been carried out. In the theorem, np denotes the total number of period instances that intersect TS(Q), Q″ denotes the sequence of time-intervals obtained after Step c) and Q_s denotes the sequence of stripped time-intervals obtained after Step d).

Theorem 6.1 For x∈CSTD,

cert(x) = (number of stripped time-intervals in Q_s which contain x) / np

Proof: Let x∈CSTD. Now, let $\{t_1, t_2, \ldots, t_m\}$ be the set of timestamps in the calendar schema S such that the pattern/event occurs at t_i for i = 1, 2,…m and $strip(t_i) = x$.

Also, let $J_1, J_2, \ldots J_p$ be all the elements in the sequence Q_s of stripped time-intervals that contain x. Because of Definition (6.8), it suffices to establish that $m = p$.

A function f is now defined from $\{1,2,3\ldots.m\}$ to $\{1,2,3,\ldots p\}$. Let $1 \leq i \leq m$. Since the pattern/event occurs at t_i and the sequence Q'' is disjoint, there is precisely one time-interval g(i) in Q'' such that $t_i \in g(i)$. Since $strip(t_i) = x$, stripintv(g(i)) contains x and hence is non-empty. Thus, stripintv(g(i)) is a stripped time-interval in Q_s that contains x. Hence, stripintv(g(i)) is $J_{f(i)}$ for some f(i) such that $1 \leq f(i) \leq p$. This defines the function f. It shall now be shown that f is a bijection and this will prove that $m = p$.

Let $1 \leq i, j \leq m$ such that $i \neq j$. Then $t_i \neq t_j$ with $strip(t_i) = strip(t_j) = x$. Since every period instance contains precisely one timestamp t with $strip(t) = x$, t_i and t_j must be in different period instances. However, every time-interval in Q'' is contained within a single period instance. As such, $g(i) \neq g(j)$ and they are different elements of the sequence Q''. Also, since $strip(t_i) = strip(t_j) = x$, stripintv(g(i)) and stripintv(g(j)) contain x and hence are non-empty. The preprocessing step d) ensures that stripintv(g(i)) and stripintv(g(j)) are different elements of Q_s that contain x. Hence, they are $J_{f(i)}$ and $J_{f(j)}$ where $f(i) \neq f(j)$. Thus, the function f is one-to-one.

Again, let $1 \leq k \leq p$. Thus, $x \in J_k = stripintv(I)$ for some I in Q''. Also, I will contain a timestamp t such that $strip(t) = x$. Since $t \in I$ which is in Q'', the pattern/event occurs at t. Hence, $t = t_i$ for some i such that $1 \leq i \leq m$. Since $t_i \in I$ (which is in Q'') and Q'' is disjoint, clearly, $I = g(i)$. Therefore, $J_k = stripintv(I) = stripintv(g(i)) = J_{f(i)}$. Hence, $k = f(i)$. This proves that f is onto. Thus the result holds.

6.4.3 Algorithms proposed

Let S be a calendar schema and Q be a sequence of time-intervals (with timestamps in S) in which a particular pattern/event has occurred. For a period defined with respect to S, the algorithms in Section 6.4.3.1 can be used to ascertain whether the pattern/event is partially or fully periodic at a given stripped timestamp x in CSTD. It may be noted that the algorithms in Section 6.4.3.1 can also be used to check if the pattern/event is almost fully periodic at x. In Section 6.4.3.2, an algorithm is proposed that can be used to find all the stripped timestamps in CSTD at which the pattern/event is either fully or almost fully periodic. The efficiency arguments of all the proposed algorithms are presented in Section 6.4.3.3.

6.4.3.1 Determining the nature of periodicity of a pattern/event at a given stripped timestamp x in CSTD

In this section, at first, an algorithm CapChange is presented to capture information about non-zero changes in the certainty of the pattern/event in CSTD. Next, the FindCert algorithm is presented, which uses this information to determine the certainty of the pattern/event at any given stripped timestamp x in CSTD. A value of cert(x) = 1 indicates a full periodicity of the pattern/event at x, whereas cert(x) < 1 shows that there is a partial periodicity of the pattern/event at x. If cert(x) < 1 but greater than or equal to a user-specified threshold, then it indicates that the pattern/event is almost fully periodic at x.

Before executing the CapChange algorithm, the preprocessing steps that are described in Section 6.4.1 need to be carried out. These steps use the input *sequence Q of time-intervals and the given period information to determine*

(i) np, viz. the total number of period instances that intersect TS(Q) and

(ii) Q_s, viz. the sequence of stripped time-intervals obtained from Q.

Q_s has to be given as input to the CapChange algorithm whereas np is used by the FindCert algorithm.

The CapChange algorithm uses the following data-structure to capture information about a non-zero change in the certainty of the pattern/event in CSTD:

chrec{tmp: stripped timestamp

iv, div: integer}

Here, *tmp* is a stripped timestamp in CSTD at which there is a non-zero change in the certainty of the pattern/event; *iv* is the number of stripped time-intervals in Q_s that contain the stripped timestamp tmp; and *div* is the change that occurred in the value of iv at tmp.

The pseudo-code of the CapChange and FindCert algorithms are presented below:

Global variables defined and set by the preprocessing steps:

np: integer /* *number of period instances that intersect TS(Q).* */

Global variables defined and set by the CapChange algorithm:

ch[]: array of chrec records

nch: integer /* *number of non-zero changes in the certainty of the pattern/event in CSTD* */

Algorithm **CapChange(Q_s, m)**

Input:

Q_s: sequence of stripped time-intervals /* *obtained after performing the preprocessing steps described in Section 6.4.1* */

m: integer /* *number of stripped time-intervals in* Q_s */

Output:

None

Global variables updated:

ch[] and nch

Variables in the scope of the **CapChange** *algorithm:*

imax: integer

```
populate(): void
{
    i: integer
    i = 0
    for every stripped interval [L, R] in Qs
        i = i + 1
        ch[i].tmp = L
        ch[i].div = 1
        if (R + 1 is in CSTD) then
            i = i + 1
```

```
            ch[i].tmp = R + 1
            ch[i].div = -1
        endif
    end for
    imax = i
}

collapse(): void
{
    i, j, netdiv: integer
    nch = 0
    i = 1
    while (i ≤ imax)
        j = i
        netdiv = 0
        while ( j ≤ imax and ch[i].tmp is equal to ch[j].tmp)
            if (ch[j].div is equal to 1) then
                netdiv = netdiv + 1
            else
                netdiv = netdiv - 1
            endif
            j = j +1
        end while
```

```
            if (netdiv is equal to 0) then
                i = j
                continue
            endif
            nch = nch + 1
            ch[nch].tmp = ch[i].tmp
            ch[nch].div = netdiv
            i = j
        end while
}

computeiv(): void
{
        i: integer
        for i = 1 to nch
            if (i is equal to 1) then
                ch[i].iv = ch[i].div
            else
                ch[i].iv = ch[i-1].iv + ch[i].div
            endif
        end for
}
```

/ Given below are the steps of the algorithm* **CapChange** */

Step 1. populate()

Step 2. Sort the cherec records in the ch[] array in non-decreasing order of the tmp field

Step 3. collapse()

Step 4. computeiv()

Algorithm **FindCert(x)**

Input:

x: stripped timestamp in CSTD

Output:

cert(x): The certainty value of the pattern/event at x

Global variables accessed:

np, ch[], nch

Variables in the scope of the **FindCert** *algorithm:*

i: integer

Step 1. if (x < ch[1].tmp or x > ch[nch].tmp) then

return 0;

endif

Step 2. Do binary search for x in ch[1....nch].tmp

Step 3. if (x is equal to ch[i].tmp) then

```
            return ch[i].iv/np
        endif
Step 4. if (ch[i].tmp < x < ch[i+1].tmp) then
            return ch[i].iv/np
        endif
```

Every stripped time-interval [L, R] in Q_s contributes to

- an increase by one in the number of stripped time-intervals in Q_s containing the stripped timestamp L (which is in CSTD)
- a decrease by one in the number of stripped time-intervals in Q_s containing the stripped timestamp $R + 1$ (provided $R + 1$ is in CSTD)

As such, by Theorem 6.1, a non-zero change in the certainty of the pattern may occur at these stripped timestamps in CSTD. At first, all these stripped timestamps are captured by the populate() routine of the CapChange algorithm in the following manner – for every stripped time-interval [L, R] in Q_s, the populate() routine inserts two chrec records into the array ch[], one with $tmp = L$ and $div = 1$ and the other with $tmp = R + 1$ and $div = -1$. It may be noted that the record for $tmp = R + 1$ is not inserted if $R + 1$ is not in CSTD. Next, all the chrec records that are inserted into the ch[] array by the populate() routine are sorted in non-decreasing order of the tmp field. Using the collapse() routine, the div field values of the chrec records that have the same tmp field value in the sorted ch[] array are now added up – a non-zero net div field value indicates a non-zero change in the number of stripped time-intervals in Q_s that contain this tmp field value. By Theorem 6.1, this indicates that there is a non-zero change in the certainty of the pattern/event at this tmp field value and so, all the chrec records having this tmp field value are collapsed into (i.e. replaced by) a single chrec record in the sorted ch[] array. The value of the div field of this chrec record is set to the net div field value.

At the end of the collapse() routine, for every non-zero change in the certainty of the pattern/event in CSTD, there is therefore one chrec record ch[i] in the ch[] array, with $1 \leq i \leq nch$. Conversely, for every chrec record ch[i] with $1 \leq i \leq nch$ in the ch[] array, there is a non-zero change in the certainty of the pattern/event at the stripped timestamp ch[i].tmp in CSTD. The computeiv() routine next computes ch[i].iv, one by one, for the records in the ch[] array, as i goes from 1 to nch, using ch[i].iv = ch[i-1].iv + ch[i].div. For i =1, ch[i-1].iv is taken to be zero. In this manner, in the ch[] array, the CapChange algorithm correctly captures information about the non-zero changes in the certainty of the pattern/event in CSTD.

The information gathered by the CapChange algorithm is used by the FindCert algorithm to determine the certainty of the pattern/event at any given stripped timestamp x in CSTD. Since the ch[] array is sorted on the tmp field of the chrec records, given a stripped timestamp x in CSTD, the FindCert algorithm simply uses a binary search to determine cert(x). If cert(x) = 1, then the pattern/event is fully periodic at the stripped timestamp x in CSTD. If on the other hand, cert(x) < 1, then the pattern/event is partially periodic at x. If cert(x) < 1 but greater than or equal to a user-specified threshold, then the pattern/event is almost fully periodic at x.

6.4.3.2 Determining all the stripped timestamps in CSTD at which the pattern/event is either fully or almost fully periodic

As mentioned earlier in Definition 6.8, for a stripped timestamp x in CSTD, $cert(x) = 1$ indicates a full periodicity of the pattern/event at x. Also, if $cert(x) < 1$ but greater than or equal to a user-defined threshold, then the pattern/event is considered to be almost fully periodic at x. Hence, to identify all the stripped timestamps in CSTD at which the pattern/event is either fully or almost fully periodic, it suffices to find the local maxima of the certainty function in CSTD. In this section, an algorithm LocMax is presented to detect local maxima of the certainty function in CSTD. For each local maximum of the certainty function in CSTD, the LocMax algorithm identifies four stripped timestamps – *lstart*, *peakstart*, *peakend*, *lend* where $lstart \leq peakstart \leq peakend \leq lend$. The certainty function has the value *startvalue* at lstart. Between lstart and peakstart, the certainty function value increases and reaches a maximum value *peakvalue* between peakstart and peakend. Between peakend and lend, the certainty function value decreases to reach a value *endvalue* at lend. If the peakvalue field is equal to 1, then it indicates that the pattern/event is fully periodic between peakstart and peakend. If the peakvalue field is less than 1 but greater than or equal to a user-specified threshold value, the pattern/event is considered to be *almost fully* periodic between peakstart and peakend.

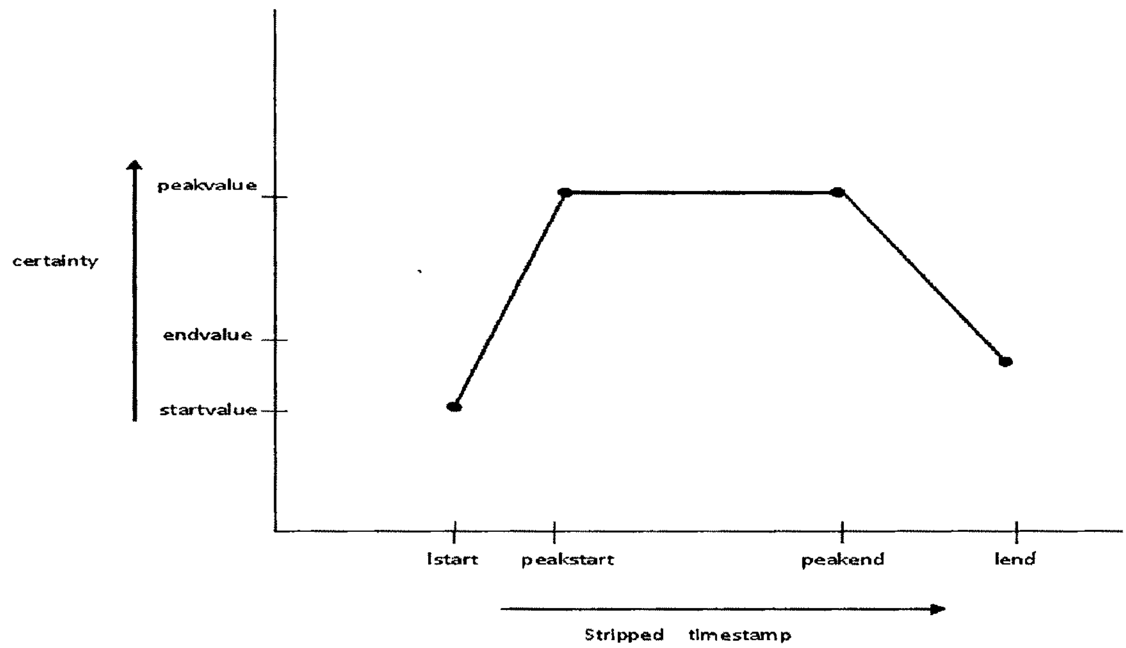

Figure 6.1 Characteristics of a local maximum of the certainty function in CSTD

The LocMax algorithm uses the following data-structure to record information about a local maximum of the certainty function in CSTD:

lmaxnode {lstart, lend, peakstart, peakend: stripped timestamp

startvalue, peakvalue, endvalue: real number}

Please note: The fields used in this structure are as described above

The CapChange algorithm presented in the preceding section has to be executed prior to the LocMax algorithm. The pseudo-code of the LocMax algorithm is given below. The code is self explanatory (see Figures 6.2 and 6.3).

Global variables defined and set by the preprocessing steps:

np: integer /* *number of period instances that intersect TS(Q)* */

Global variables defined and set by the CapChange algorithm:

nch: integer /* *number of non-zero changes in the certainty of the pattern/event in CSTD* */

ch[]: array of chrec records /* *ch[] is the array of chrec records that is created by the CapChange algorithm. For each non-zero change in the certainty of the pattern/event in CSTD, there is one record in ch[i], with 1 ≤ i ≤ nch. Conversely, for each chrec record ch[i] with 1 ≤ i ≤ nch in the ch[] array, there is a non-zero change in the certainty of the pattern/event at the stripped timestamp ch[i].tmp in CSTD* */

Global variables defined and set by the LocMax algorithm:

nlmax: integer /* *number of local maxima of the certainty function in CSTD* */

lmax[]: array of lmaxnode

Algorithm **LocMax()**

Input:

None

Output:

None

Global variables accessed (not updated):

ch[], nch, np

Global variables updated:

nlmax, lmax[]

Variables in the scope of the **LocMax** *algorithm:*

i: integer

nondecreasing: boolean

```
Step 1.  nondecreasing = false,  nlmax = 0

Step 2.  for i = 1 to nch

         Step 2(a)  if (nondecreasing is false) and (ch[i].div > 0) then

                        if (nlmax > 0) then

                            lmax[nlmax].lend = ch[i].tmp - 1

                            lmax[nlmax].endvalue = ch[i-1].iv/ np

                        endif

                        nlmax = nlmax + 1

                        lmax[nlmax].lstart = ch[i].tmp

                        lmax[nlmax].startvalue = ch[i].iv/np

                        nondecreasing = true

                    endif

         Step 2(b)  if (nondecreasing is true and ch[i].div < 0) then

                        lmax[nlmax].peakstart = ch[i-1].tmp

                        lmax[nlmax].peakvalue = ch[i-1].iv / np

                        lmax[nlmax].peakend = ch[i].tmp - 1
```

nondecreasing = false

endif

Step 2(c) if (i is equal to nch) then

lmax[nlmax].lend = ch[i].tmp

lmax[nlmax].endvalue = ch[i].iv / np

endif

end for

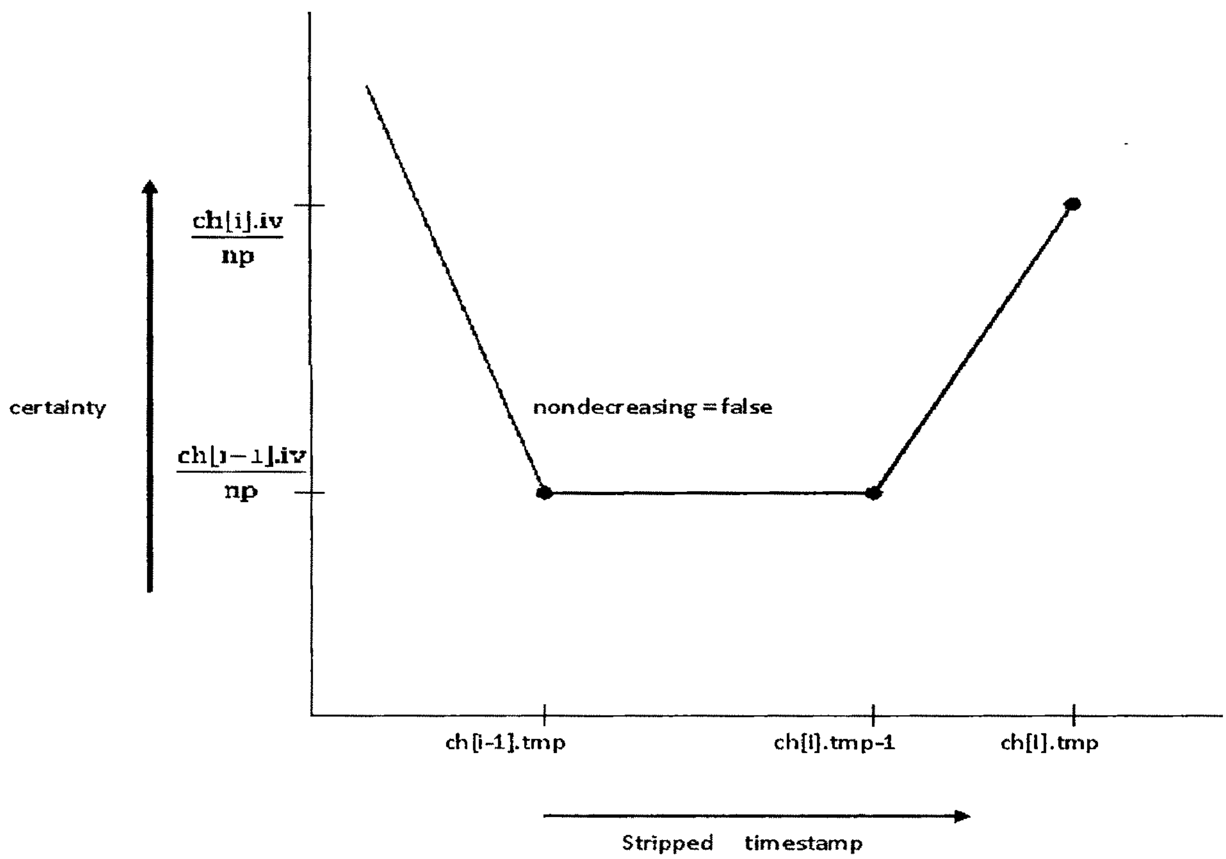

Figure 6.2 Detection of the start of a local maximum (Step 2(a) of the LocMax algorithm)

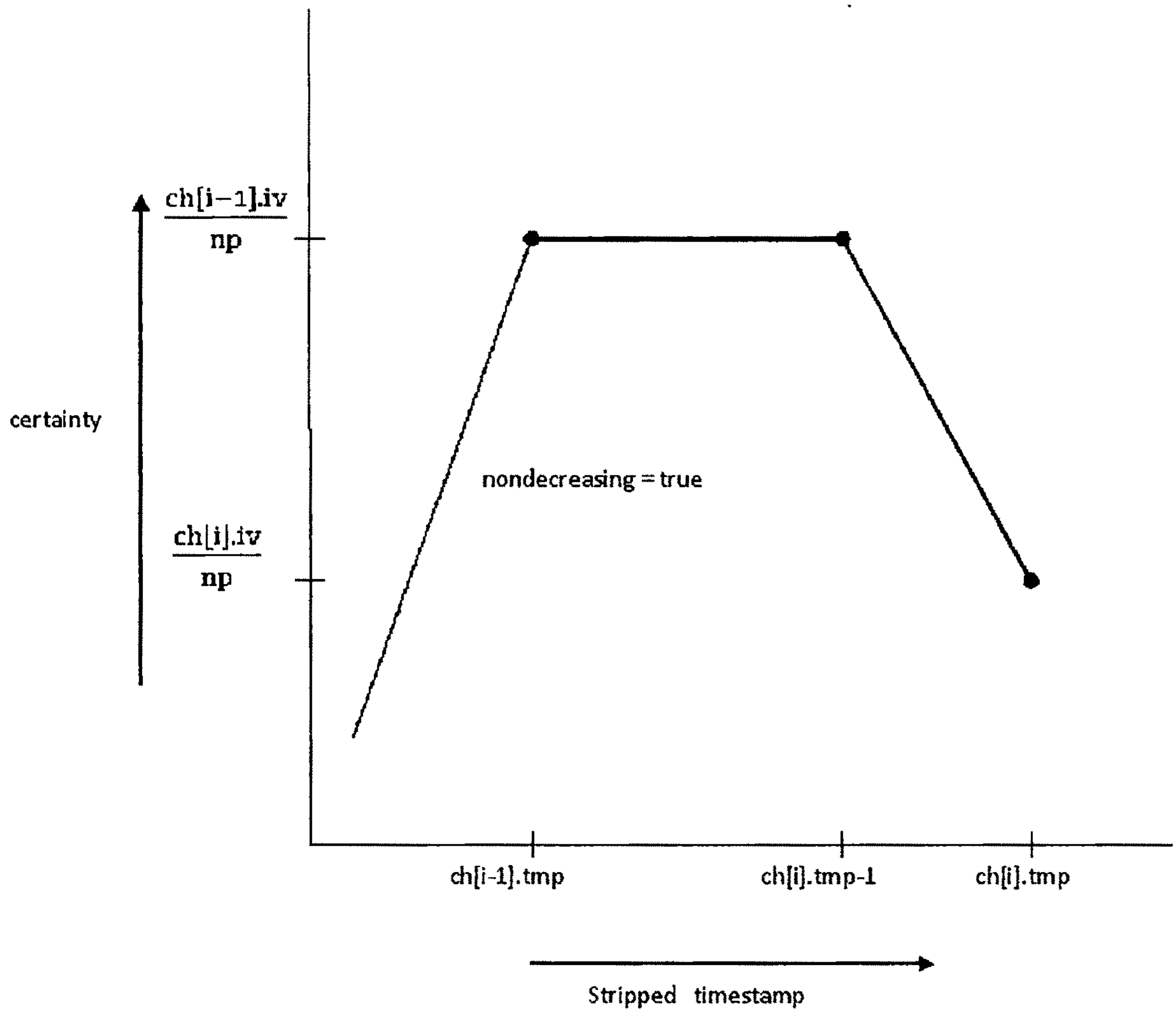

Figure 6.3 Detection of the end of the peak of a local maximum (Step 2(b) of the LocMax algorithm)

6.4.3.3 Efficiency arguments

Worst-case time complexity of the preprocessing steps: Among the sequences of time-intervals Q, Q′, Q″ and Q_s, the sequence Q″ will have the highest number n of time-intervals. In terms of n, the time required by the different preprocessing steps is shown below –

Step a): O(n)

Step b): Since an initial sorting needs to be done on the left endpoint of the time-intervals, this step is O(n log n)

Step c): O(n)

Step d): O(n)

Thus, the overall time-complexity is O(n + n log n + n + n), which is O(n log n).

Worst-case time complexity of the CapChange algorithm: In terms of the value of n as defined above, the time required by the different steps of the algorithm is shown below –

Step 1: O(n)

Step 2: O(n log n)

Step 3: O(n)

Step 4: O(n)

Thus, the overall worst-case time-complexity of the CapChange algorithm is O(n + n log n + n + n), which is O(n log n).

Worst-case time complexity of the FindCert algorithm: In terms of the same value of n as defined above, the time required by the different steps of the algorithm is shown below –

Step 1: O(1)

Step 2: O(log n)

Step 3: O(1)

Step 4: O(1)

Thus, the overall worst-case time-complexity of the FindCert algorithm is O(1 + log n + 1 + 1), which is O(log n).

To determine the certainty of the pattern/event under study at α different stripped timestamps in CSTD, the CapChange algorithm has to be executed just once. After this, the FindCert algorithm has to be executed once for each of the α stripped timestamps. Thus, the overall worst-case time-complexity of this task is $O((n + \alpha) \log n)$.

Worst-case time complexity of the LocMax algorithm: In terms of the same value of n as defined above, the time required by the different steps of the algorithm is shown below –

Step 1: O(1)

Step 2: This step is O(1) and it is executed O(n) times. Hence, overall time spent in this step is O(n) /* *since* $nch \leq n$ */

Steps 2(a), 2(b), 2(c): Each of these steps is O(1) and each of them is executed O(n) times. Hence, overall time spent in these steps is O(n)

Thus, the overall worst-case time-complexity of the LocMax algorithm is O(1 + n + n + n+ n), which is O(n).

To determine all the stripped timestamps in CSTD at which the pattern/event under study is fully or almost fully periodic, the CapChange algorithm has to be executed just once. After this, the LocMax algorithm has to be executed once. Thus, the overall worst-case time-complexity of this task is O(n log n + n), which is O(n log n)

6.5 EXPERIMENTAL RESULTS

The algorithms CapChange and LocMax are tested on three real-life datasets.

The first real-life dataset contains the time-intervals during which tropical storms occurred in the eastern-pacific region between the years 1949 to 2008. The source of this data is http://weather.unisys.com/hurricane/index.html. The timestamps in this dataset are in the calendar schema (year/month/day). The CapChange and LocMax algorithms are used to look for yearly periodicities of tropical storms in the eastern-pacific region. In Table 6.1, all the stripped time-intervals in which a tropical storm is likely to occur with a certainty of at least 20% in the eastern-pacific region are shown. The maximum certainty value reached across each of these stripped time-intervals is also stated.

Table 6.1 Yearly periodicities of tropical storms across the eastern-pacific region

Stripped time-intervals	Maximum certainty value reached in this span (in %)
6th July to 8th July	25
16th July to 24th July	34
27th July to 30th July	30
21st August to 6th September	32
17th September to 4th October	32

The second real-life dataset is a time-series containing daily average temperatures (in degrees centigrade) of Hveravellir (in Iceland) from 1st January, 1972 to 31st December, 1974. The source of this data is http://robjhyndman.com/TSDL/meteorology/. The timestamps in this dataset are in the calendar schema (year/month/day). At first, the dynamic time-warping (DTW) technique (described in Appendix IV) is implemented to extract from this time-series, all the time-intervals (spanning across 4 to 20 days) in which a 3°C temperature rise is detected. The CapChange and LocMax algorithms are then used to look for monthly periodicities of these 3°C temperature rises across the time-series.

Table 6.2 shows the days that are likely to be involved in a 3°C temperature rise in Hveravellir with a certainty of at least 30 %.

Table 6.2 Monthly periodicities of a 3°C temperature rise in Hveravellir (in Iceland)

Stripped time-intervals	Maximum certainty value reached in this span (in %)
1st to 11th	47

The third real-life dataset is a time-series containing IBM closing stock prices (in USD) from 1st January, 1980 to 8th October, 1992. The source of this data is http://robjhyndman.com/TSDL/finance/ . The timestamps in this dataset are also in the calendar schema (year/month/day). Again, the DTW technique is used to extract from this time-series, all the time-intervals (spanning across 5 to 20 days) in which there is a 2$ rise followed by a 2$ fall in the closing stock prices. The CapChange and LocMax algorithms are then used to look for yearly periodicities of this pattern across the time-series.

Table 6.3 shows the stripped time-intervals that are likely to be involved in such a pattern with a certainty of at least 30%. The maximum certainty value reached across each of these stripped time-intervals is also stated.

Table 6.3 Yearly periodicities of a 2$ rise followed by a 2$ fall in the closing stock prices of IBM

Stripped time-intervals	Maximum certainty value reached in this span (in %)
3rd January to 10th January	38

13^{th} January to 15^{th} January	30
2^{nd} February to 3^{rd} February	30
4^{th} March to 11^{th} March	61
16^{th} March to 17^{th} March	61
21^{st} April to 24^{th} April	46
4^{th} May to 12^{th} May	46
23^{rd} May to 25^{th} May	30
8^{th} June to 29^{th} June	76
17^{th} July to 28^{th} July	61
26^{th} August to 5^{th} September	61
31^{st} October to 6^{th} November	38
6^{th} December to 10^{th} December	54
23^{rd} December to 31^{st} December	76

6.6 SUMMARY OF CONTRIBUTIONS

In this chapter, several notions related to patterns/events that occur in certain time-intervals were introduced. In particular, the notions of a stripped timestamp, a common stripped timestamp domain (CSTD) and the certainty of a pattern/event at a stripped timestamp in CSTD were precisely defined. At first, an algorithm CapChange was proposed to capture non-zero changes in the certainty of the pattern/event in CSTD. Next, an algorithm FindCert was presented, which used this information to determine the certainty of the pattern/event at a given stripped timestamp x in CSTD. For a pattern/event which occurs during certain time-intervals, the certainty value at a stripped timestamp x indicates the nature of periodicity of the pattern/event at x. Finally, an algorithm LocMax was proposed to detect local maxima of the certainty function in CSTD. This helps to identify all the stripped timestamps in CSTD at which the pattern/event is fully or almost fully periodic. The correctness of the proposed algorithms were justified and their efficiency arguments were given. Experimental results obtained by testing the algorithms on real-life datasets were reported.

The work reported in this chapter concludes the contributions made by the present work to the area of interval data mining.