Mining Icebergs in Time-Stamped Databases

Jhimli Adhikari[1], P R Rao[2], Witold Pedrycz[3]

[1]Department of Computer Science, Narayan Zantye College, Bicholim, Goa - 403 529, India
jhimli_adhikari@yahoo.co.in
[2]Department of Computer Science and Technology, Goa University, Goa - 403 206, India
pralhadrrao@gmail.com
[3]Department of Electrical and Computer Engineering, University of Alberta, Alberta T6G 2V4, Canada
pedrycz@ece.ualberta.ca

Abstract. Many organizations possess large databases collected over a long period of time. Analysis of such databases might be strategically important for further growth of the organizations. It might be interesting as well as useful to learn about interesting changes in sales over time. In this paper, we have introduced a new pattern, called notch, of an item in time-stamped databases. Based on this pattern, we have proposed two special kinds of notch, called generalized notch and iceberg notch, in time-stamped databases. Also we have identified an application of generalized notch. We have designed an algorithm for mining interesting icebergs in time-stamped databases. We have presented experimental results on both synthetic and real-world databases.

Keywords: data mining; generalized notch; iceberg notch; notch; temporal pattern; time-stamped database.

1 Introduction

Many organizations collect transactional data continuously over a long period of time. A database grown over a long period of time might contain useful as well as interesting temporal patterns. By taking into the account of time aspect, many interesting applications as well as patterns such as surprising patterns [17], discords [19], calendar based patterns [24] have been reported in the recent time. Surprising patterns, anomaly detection and discords could be considered as exceptional types of pattern occurring in a time series data. These exceptional patterns are important as well as interesting contributions to temporal data mining. The goal of this paper is to identify another type of exceptional pattern in transactional time-stamped data.

Though an analysis of time series data [8], [9], [16], [29] has been intensively studied, the analysis of time-stamped data still calls for more research. Specifically, in the context of multiple time-stamped databases, little work has been reported so far. Therefore, there arises an urgent need to study multiple time-stamped databases. In Example 1, we observe an interesting type of temporal pattern in multiple time-stamped databases that needs to be analyzed fully.

The support of an itemset [4] is defined as the fraction of transactions containing the itemset. It has been used extensively in identifying different types of patterns in a database. Some examples are association rule [4], negative association rule [31] and conditional pattern [1]. Nonetheless the support measure has a limited use in discovering some other types of patterns in a database. We illustrate this issue using the following example.

Example 1. Consider a company that maintains customers' transactions on a yearly basis. Many important problems can be studied given such yearly databases. Let item *A* be of our interest. In view of analyzing item *A* over the years, let us consider the sales series of *A* from the year 1970 to 1979.

(0.9, 1000, 1970), (0.31, 2700, 1971), (0.36, 2500, 1972), (0.29, 3450, 1973), (0.37, 1689, 1974), (0.075, 7098, 1975), (0.073, 8900, 1976), (0.111, 6429, 1977), (0.09, 9083, 1978), (0.07, 10050, 1979). The first, second, and the third component of each triple refers to the support of *A*, the number of transactions in the yearly database and the corresponding year, respectively.

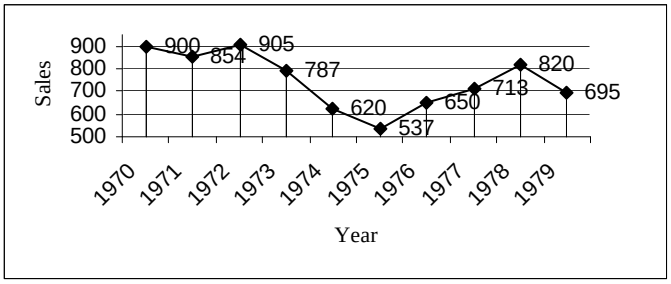

Figure 1. Sales of item *A* reported in the successive years

The sales series of item *A* is depicted in Figure 1. There is a significant downfall of sales from 1972 and rise in sales from the year 1975. Year 1975 is an important point [26] for the company. It is a significant down-to-up change in the sales series. It is not surprising to observe a significant up-to-down change in a sales series of an item. Such patterns in time-stamped series are interesting as well as important to investigate. It could reveal the sources of customers' purchase behavior and that might provide an important knowledge to an organization.

At this point, one might be interested in knowing how a time series data differs from a time-stamped data. Transactional data are time-stamped data collected over time at no particular frequency [20]. Whereas, time series data are time-stamped data collected over time at a particular frequency. For example, point of sales data could be considered as time-stamped data, but sales per month / year could be considered as time series data. One could convert transactional data into time series data for the purpose of specific data analyses. The frequency associated with time series data varies from problem to problem. For future planning of business activities, one might need to analyze the past data of customer transactions collected over a long period of time. While analyzing the past data it is useful as well as important to figure out the abrupt changes in sales of an item along with time. Existing algorithms, as mentioned

above, fail to detect these changes. Therefore, in this paper our objective is to define such exceptional change (pattern) and design an algorithm to extract such pattern from time-stamped databases.

For the purpose of studying patterns in time-stamped databases one may need to handle multiple databases over time. One could call these time variant databases as *time databases*. In this context, the choice of time granularity is an important issue as the characteristics of temporal patterns is heavily dependent on this parameter. It is quite reasonable to consider the time granularity as one year, since a season re-appears on a yearly basis and the customers' purchase behaviour might vary from season to season.

Consider an established company having data over fifty consecutive years. The company might be interested in knowing the performance of different items over the years. Such analysis might help the company in devising the future strategies. The objective of this paper is to identify abrupt changes in sales of each item over the years as depicted in Figure 1.

Rest of the paper is organized as follows. We discuss related work in Section 2. In Section 3, we introduce a new temporal pattern, called notch, of an item. Based on this pattern, we propose the concepts of generalized notch (Section 4) and iceberg notch (Section 5). We present another view of sales series in Section 6. In Section 7 we design an algorithm for mining icebergs in time-stamped databases. Experimental results are presented in Section 8.

2. Related Work

Temporal sequences appear in a vast range of domains ranging from engineering to medicine and finance, and the ability to model and extract information from them becomes crucial from a conceptual as well as applied perspective. Identifying exceptional patterns in time-stamped databases deserves much attention. In Sections 4 and 5 we shall propose two exceptional patterns in time-stamped databases.

There are mainly two broad directions of temporal data mining [27]. One concerns the discovery of casual relationships among temporally oriented events. Another one deals with the discovery of similar patterns within the same time sequence or among different time sequences. Sequences of events describe the behavior and actions of users or systems that can be collected in several domains. The proposed problem falls under the first category of problems, since we are interested in identifying exceptional patterns by comparing sales in different years.

Agrawal et al. [6] introduced the *shape definition language* (SDL), which used limited vocabulary such as {*Up, up, stable, zero, down, Down*} to describe different gradients in the series. The similarity of two time series is proportional to the length of the longest common sequence of words in their *SDL* representation. Such coarse information might not be always helpful. We define two exceptional patterns viz. a generalized notch and an iceberg notch.

Perng et al. [25] proposed the landmark model where perceptually important points of a time series are used in its representation. The perceptual importance depends on the specific type of the time series. In general, sound choices for landmarks are local

maxima and minima as well as inflection points. The advantage of using the landmark-based method is that this time representation is invariant to amplitude scaling and time warping. Some of the local maxima and minima might lead to higher level of exceptionality. Here we are concerned with defining such exceptionalities in time-stamped databases.

There has been a significant amount of work on discovering temporal patterns of interest in sequence databases and time series databases. Temporal data mining is concerned with the analyses of data with an intention of finding patterns and regularities from a set of temporal data. In this context sequential association rule [6], periodical association rule [22], calendar association rule [21], calendar-based periodic pattern [24] and up-to-date pattern [15] are some of the interesting temporal patterns reported in the recent time.

As noted in Section 1, support history of an item provides important information of an item over time. We have proposed an algorithm for clustering items in multiple databases based on their support history [2]. We have introduced the notion of stability of an item based on its support history.

Lomet et al. [23] integrated a temporal indexing technique, the TSB-tree, into Immortal DB to serve as the core access method. The TSB-tree provides high performance access and update for both current and historical data.

Keogh et al. [18] proposed an algorithm for finding unusual time series where the notion of time discords is introduced. A time discord is a subsequence of a longer time series that is maximally different from all other subsequences of the series. Discords can be used to detect anomalies in an efficient way.

Many algorithms are designed incrementally to support time-dependent analysis of data. We have proposed algorithms incrementally to study overall influence of a set of items on another set of items in time databases [3]. Castellana et al. [10] proposed a new approach to performing change detection analyses based on a combination of supervised and unsupervised techniques is presented. Experimental results are based on image data. Wang et al. [30] examined an unsupervised search method to discover motifs from multivariate time series data. The algorithm first scans the entire series to construct a list of candidate motifs in linear time, the list is then used to populate a sparse self-similarity matrix for further processing to generate the final selections. But our algorithm is based on time-stamped data. 3 Problem definition

We are given k time databases as described in Section 2. Let I be the set of all items in time databases. Each item in I is associated with an amount of sales in time database DT_i, for $i = 1, 2,\ldots, k$. Thus, there exists a series containing k sales amount for each item in I. There may exist interesting notches in different years for a sales series of an item. In this paper, we are interested in mining notches in sales series of different items in time databases.

3. Notches in Sales Series

The change in sales of an item could be defined by the change of its support over time. The support of an item results in a *support history* [7] of the item in time databases. An analysis of a support history could be important in understanding

customers' behavior [2]. While dealing with the support history, the size of a database is an important issue. Support 0.129 from a database containing 1,000 transactions might be less important than the support 0.091 from a database containing 1,00,000 transactions. Thus, a mere analysis of the support history over time might not be effective in an application. One needs to analyze the supports along with the sizes in time databases. In Example 1, we observe that the support of *A* has been decreased from year 1970 to 1971. But, the sales of *A* have been increased from the year 1970 to 1971. Hence, a negative change in support of an item might imply a positive change in frequency of the item. Thus, one needs to be careful in dealing with support history of an item in different databases.

Let us consider a company that has been operating for the last *k* years. For the purpose of studying temporal patterns, yearly databases could be constructed based on a time-stamp. Each of these databases corresponds to a specific period of time. Let *D* be the collection of customer transactions over *k* years. For the purpose of defining a temporal pattern we divide *D* into *k* yearly databases. Let DT_i be the database corresponding to the *i*-th year, $i = 1, 2, \ldots, k$. Each of these time databases is mined using a traditional data mining technique [5], [13]. Mining time-stamped databases could help business leaders make better decisions by listening to their suppliers and / or customers via their transactions collected over time [20].

Over the years, an item may exhibit many consecutive data points having similar sales. As opposed to similar data patterns considering each data point, a limited yet meaningful number of points, may play a dominant role in many decision making problems. These meaningful data points could be defined in various ways, like average, peak, or slope of lines [26]. In the context of the proposed problem, such compression of data points seems to be irrelevant. Given the sales series of an item, one might be interested in identifying abrupt changes in the sales series. The goal of this paper is to define a new type of pattern based on abrupt variation of sales of an item over the years and to design an algorithm to mine such patterns in time databases.

Over the years, there may exist many ups and downs in sales of an item. One might be interested in identifying abrupt changes of sales in different years. It might be helpful to figure out the causes behind it and to take actions accordingly. Let $s_i(A)$ be the sales of item *A* for the *i*-th year, $i = 1, 2, \ldots, k$. We define the change in sales series of item *A* at year *i* as follows [28].

The change in sales series at year *i* is *increasing* if

$$s_{i-1}(A) < s_i(A) < s_{i+1}(A),\ i = 2, 3, \ldots, k\text{-}1. \quad (1)$$

The change of sales series at year *i* is *decreasing* if

$$s_{i-1}(A) > s_i(A) > s_{i+1}(A),\ i = 2, 3, \ldots, k\text{-}1. \quad (2)$$

The change of sales series at year *i* is *altering* if

$$s_{i-1}(A) < s_i(A) \text{ and } s_i(A) > s_{i+1}(A), \quad (3)$$

or, $$s_{i-1}(A) > s_i(A) \text{ and } s_i(A) < s_{i+1}(A),\ i = 2, 3, \ldots, k\text{-}1 \quad (4)$$

The notion of *strict extrema* [11] at a year corresponding to an item is defined as follows.

Let $s_i(A)$ be the amount of sales of an item A at year i, $i = 1, 2,\ldots, k$. There exists a *strict extreme* at year i for the item A if the change of support history of A at year i is altering. Based on the concept of strict extrema, we define a notch as follows.
Definition 1. There exists a *notch* at year i for the item A if there is a strict extreme at year i in the sales series of item A.

Let $\Delta s_i(A)$ be the difference in sales of item A between the years i and i-1. Now we propose a few definitions as follows.

Definition 2. Let there exists a notch at year i for item A. The notch at year i for item A is *downward* if $\Delta s_i(A) < 0$ and $\Delta s_{i+1}(A) > 0$.

Definition 3. Let there exists a notch at year i for item A. The notch at year i for item A is *upward* if $\Delta s_i(A) > 0$ and $\Delta s_{i+1}(A) < 0$.

Itemset [4] could be considered of as a basic type of pattern present in a transactional database. Many important as well as interesting patterns are based on itemset patterns. Similarly an upward / a downward notch could be considered as a basic type of pattern in time databases. In Section 4, we illustrate how the notion of notch could help analyzing a special type of trend in time databases. Thus, it is important to mine notches in time databases.

Notches in sales series of an item could be considered as basic building blocks to construct temporal patterns. Next we introduce the notions of two interesting temporal patterns viz., generalized notch and iceberg notch in time databases. One could simply scan the sales series of an item to identify its interesting notches. Let n and k be the number of items in time databases and the number of (yearly) time-stamped databases, respectively. Then the time complexity of identifying notches is $O(n \times k)$.

4. Generalized Notch

Based on the concept of notch, we present here the notion of a generalized notch in time databases. Let us refer to Figure 1. There are two downward notches in the years 1971 and 1975 having sales 854 and 537, respectively. The concept of notch can be generalized based on strict extrema as mentioned in Section 3. One could notice in Figure 1 that the downward notch in the year 1975 is wider than that of 1971. The width of a downward generalized notch is based on the two consecutive local maxima within which the downward generalized notch is enclosed. The width of the downward generalized notch in 1975 is 1978 – 1972 = 6. Similarly, the width of an upward generalized notch is based on the two consecutive local minimums within which the upward generalized notch is enclosed. The width of the upward generalized notch in 1971 is 1975 – 1971 = 4. Based on the above discussion, we define width of a generalized notch as follows.

Definition 4. Let there be a generalized downward (upward) notch in the year i. Also, let the generalized downward (upward) notch be enclosed within the local maximums

(minimums) in the years i_1 and i_2 ($> i_1$). The width of the generalized notch in the year i is equal to $i_2 - i_1$.

The width of a generalized notch could be divided into left width and right width. The left width and right width are equal to $(i - i_1)$ and $(i_2 - i)$, respectively. In case of downward generalized notch the sales value gradually decreases, and then attains the minimum value, and then it gradually increases. Thus, the change of sales value between two consecutive years seems to be an important characteristic of a generalized notch. In this regard, one might be interested in the change of sales for a year as compared to its previous year. Also, the sales at year i as compare to sales of year i_1 and i_2 are important characteristics of a generalized notch. Accordingly, one could define left-height and right-height of the generalized notch in the year i for an item A as follow:

$$\textit{left-height}\ (A, i) = |\textit{sales}(A, i_1) - \textit{sales}(A, i)|, \text{ and}$$

$$\textit{right-height}\ (A, i) = |\textit{sales}(A, i) - \textit{sales}(A, i_2)|$$

We define the height of a generalized notch as follows.

Definition 5. Let there exists a generalized downward (upward) notch in the year i. Also, let the generalized downward (upward) notch be enclosed with the local maximums (minimums) i_1 and i_2 ($> i_1$). The height of the generalized notch in the year i is equal to *maximum* {*left-height* (A, i), *right-height* (A, i)}.
Based on the concept of generalized notch, we focus on the notion of iceberg.

5. Iceberg Notch

An analysis of sales series of items is an important issue. In view of performing this task, one could analyze the sales series for each item. In analyzing a sales series in-depth, it is evident that an existence of a notch might be an indication of a bigger notch. This represents an exceptionality of sales of an item. Based on such an exception, we define iceberg in time databases as follows.

Definition 6. An *iceberg* notch is a generalized notch that satisfies the following conditions: (i) The height of the generalized notch is greater than or equal to α, and (ii) The width of the generalized notch is greater than or equal to β. Both α and β are user-defined thresholds.
An iceberg notch is a generalized notch having a larger width and a lager height. The concept of iceberg in data management is not new. For example, iceberg queries [14] are commonly used in data mining, particularly in market basket analysis.

Let us illustrate the concept of an iceberg using an example. Let the value of α and β be set to 300 and 5, respectively. Also, let the values of i, i_1 and i_2 be 1975, 1972, and 1978, respectively (with respect to Figure 1). We observe that *leftHeight*$(A, 1975)$ $= |\textit{sales}(A, 1972) - \textit{sales}(A, 1975)| = |905 - 537| = 368$ and *rightHeight*$(A, 1975) = |\textit{sales}(A, 1975) - \textit{sales}(A, 1978)| = |537 - 820| = 283$, respectively. Therefore, the height of the iceberg is *maximum* {368, 283} i.e. $368 \geq \alpha$. Also, $i_2 - i_1 = (1978 -$

1972) = 6 ≥ β. So, there exists an interesting downward iceberg notch in the year 1975. We also observe an upward notch in the year 1972 with height 368 ≥ α and width (1975 – 1971) = 4 < β. So, the upward notch in the year 1972 is not an iceberg. We state the problem as follows.

Let there are k time databases. Let I be the set of all items in time databases. There exists a sales series of each item in I, consisting of k sales values, one for each time database. Mine iceberg notches in sales series for each item in I.

6. Sales Series

A sales series of an item might provide an interesting information about the item. It is basically the same as the support history of the item. As noted above, in many problems, it is preferable to analyze sales series rather than looking at the support history of an item. Many temporal patterns might originate by analyzing such types of temporal series.

Each data in a sales series can be mapped into a member in the set {–1, 0, 1} by comparing with the previous data in the same series. Thus, a time-stamped series could be mapped into a ternary series. It provides a simplified view of the original time-stamped series data. Such simplified view might provide some useful information. The procedure for mapping a time-stamped series into a ternary series is illustrated in the following example.

Example 2. Consider the sales data given in Example 1. The sales of item *A* in 1971 decreased from the sales in 1970. We note this behavior by putting –1 in the ternary series of item *A* corresponding to year 1971. The sales of item *A* in 1972 increased over the sales in 1971. We note this behaviour by putting +1 in the ternary series of item *A* corresponding to year 1972. If the sales of item *A* in any year remains same as that of previous year then we note this behaviour by putting 0 in the ternary series. Thus, we obtain the ternary series (*TS*) of item *A* in the following form:

Year	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979
TS(*A*)		–1	+1	–1	–1	–1	+1	+1	+1	–1

In the above series, one can observe the existence of a generalized notch. The width of a downward generalized notch can be obtained from a run of –1's and the subsequent run of +1's. The width is obtained by adding the number of –1's in the first run and the number of +1's in the second run. A similar procedure can be followed to finding width of an upward generalized notch. In *TS*(*A*) we observe a downward generalized notch in 1975. The width of this notch is equal to 3 + 3 = 6. Also, there exists an upward generalized notch in 1978 having width of 4.

A slightly different procedure could also be followed to obtaining a ternary series corresponding to a sales series of an item. Let x and y be the sales for the year 1970 and 1971, respectively. Let δ be the level of significance of difference in sales. We put +1 in the ternary series of the item corresponding to year 1971, if $y - x > \delta$. We put –1 in the ternary series of the item corresponding to year 1971, if $x - y > \delta$. We put 0 in the ternary series of the item corresponding to year 1971, if $|x - y| \leq \delta$. The

method of obtaining a ternary series using this procedure might be useful in many situations, since a small change in sales value might be insignificant in many situations. This procedure seems to be more realistic than the previous one.

7. Mining Icebergs in Time-Stamped Databases

Let there are n items in time databases. For each item in time databases there exists a time-stamped series containing k data. In this section we are interested in identifying icebergs in each time-stamped series.

For mining icebergs in time databases, we make use of an existing frequent itemset mining algorithm [5], [13]. For the requirement of proposed problem one needs to mine the frequencies of each item in the time databases. Based on the discussion held in previous sections, we design an algorithm for mining icebergs in time databases. Let n and k be the number of items in time databases and the number of time databases, respectively. In this algorithm, we use a two-dimensional array F for storing frequencies of all items in time databases. F consists of n rows and $k + 1$ columns. The first column contains the items in time databases. For example, $F(i)(1)$ contains the i-th item in time databases, $i = 1, 2, \ldots, n$. The i-th row of F contains i-th item and its frequencies in k time databases. For example, $F(i)(j)$ contains the frequency of i-th item in $(j - 1)$-th database, $j = 2, 3, \ldots, k+1$. Therefore, we need to check the existence of a generalized notch using the values in the columns from 2 to $k + 1$.

For the purpose of computing interestingness of a generalized notch, we determine the change of sales of a local minimum (maximum) with respect to its previous and next local maximum (minimum). During the process of mining icebergs, the generalized notches are kept in array *GN*. A generalized notch can be described by the following attributes: left year (*leftYear*), right year (*rightYear*), item (*item*), year of occurrence (*year*), type of generalized notch (*type*), change of sales at the year of occurrence with respect to the previous local extremum (*leftHeight*), change of sales at the year of occurrence with respect to the next local extremum (*rightHeight*), width of generalized notch (*width*) and the sales at the year of occurrence (*sales*). The goal of the proposed algorithm is to find all the interesting icebergs for each item in time databases. The algorithm is given as follows.

Algorithm 1. Mine icebergs in time-stamped databases.

```
procedure MiningIcebergs (k, F, α, β)
Inputs: k, F, α, β
k: number of yearly databases
F: array of frequencies of items in yearly databases
α: user-defined threshold of height
β: user-defined threshold of width
Outputs: Interesting icebergs in time databases
01: let index = 1;
02: for i = 1 to n do
03:  let j = 2,left = 2,flat = f;
     prevDown = f,prevUp = f;
04:  while not end of the i-th sales series do
```

```
05:   if there is downward trend and prevUp is f then
06:    find mid, leftWidth; let prevDown = t;
07:    compute leftHeight; go to 04;
08:   end if {05}
09:   if there is upward trend and prevDown is t then
10:    find left, mid, right, leftWidth, rightWidth;
11:    let prevDown = f; leftHeight = rightHeight;
12:    compute rightHeight;
13:    GN(index).type = down; go to 30;
14:   end if {09}
15:   if there is upward trend and prevDown is f then
16:    let prevUp = t; find mid, leftWidth;
17:    compute leftHeight; go to 04;
18:   end if {15}
19:   if there is downward trend and prevUp is true then
20:    find left, mid, right, leftWidth, rightWidth;
21:    let prevUp = f; let prevDown = t;
22:    let leftHeight  = rightHeight;
23:    compute rightHeight;
24:    GN(index).type = up; go to 30;
25:   end if {19}
26:   if sales of j-th and (j+1)-th year are same then
27:    find left; let flat = t, prevDown = f, prevUp = f;
28:    go to 04;
29:   end if
30:   if flat is f then
31:    compute height as defined in Definition 5;
32:    if current notch satisfies criteria α and β then
33:     store it in GN(index); increase index by 1;
34:    end if
35:   if the current generalized notch is downward then
36:    let prevDown = f, prevUp = t;
37:   else if current generalized notch is upward then
38:    let prevDown = t; let prevUp = f;
39:   end if
40:   end if {35}
41:  end if {30}
42: end while {04}
43: end for {02}
44: display icebergs from GN;
end procedure
```

The lines 2-43 are repeated for each item in time databases. In each repetition, the interesting icebergs corresponding to an item are identified. The variable *index* is used to index array *GN*. The variable *j* is used to keep track of current sales of the item under consideration. The starting value of *j* is 2, since the sales data for the first year of an item is kept staring from column number 2 of array *F*. We use three Boolean variables viz., *flat*, *prevUp* and *prevDown*. While identifying downward (or, upward) generalized notches, we first go through its left leg of a generalized notch. After reaching its minimum / maximum value, if the next point also attains the same value then *flat* becomes true. Truth values *true* and *false* are represented by *t* and *f* respectively. The width of a generalized notch is determined by the following years:

left year (*left*), middle year (*mid*), and right year (*right*). Accordingly, the width of a generalized notch has two components viz, left width (*leftWidth*) and right width (*rightWidth*). After identifying the left leg of a downward (or, upward) generalized notch, *prevDown* (or, *prevUp*) becomes true. After storing the details of the current generalized notch *index* gets increased by 1 (line 33). We identify generalized notches for each item in time databases. For this purpose, we introduce a *for loop* at line 02 that ends at line 43. Some lines such as lines 40, 41, 42, 43 are ended with a number enclosed in curly brackets to mark the end of composite statement starting with the line number kept in curly bracket.

Lines 2 and 4 repeat for n and k times respectively. In other words, the sales series corresponding to each item is processed for identifying icebergs. Thus, the time complexity of lines 1-43 is $O(n \times k)$. Again, the time complexity of line 44 can not be more than $O(n \times k)$, since the number of interesting icebergs is always less than $n \times k$.

Theorem 1. Corectness of the *MiningIcebergs* algorithm.
Proof: Consider that there are n items in k time-stamped databases. Each sales series is processed using lines 2-43. For the purpose of mining interesting icebergs, each sales series is checked completely by applying a while-loop shown in lines 4-42. A sales series can start with one of the following three ways: (a) showing a downward trend, (b) showing an upward trend, and (c) remained at a constant level. The algorithm handles each of the cases separately.

Case (a) has been checked at the line 5. Once a downward trend changes we again go back to while-loop at line 4 for finding one of the following two possibilities viz, an upward trend and a constant sale.

Case (b) has been checked at the line 15. Once the upward trend changes we again go back to while-loop at line 4 for finding one of the following two possibilities viz, a downward trend and a constant sale.

Case (c) has been checked at the line 26. Once the flatness changes we again go back to while-loop at line 4 for finding one of the following two possibilities viz, a downward trend and an upward trend.

Once the left leg of a downward generalized notch is detected in lines 5-8, its right leg is detected in lines 9-14. When the left leg of an upward generalized notch is detected in lines 15-18, its right leg is detected in lines 19-25. After detecting a generalized notch at lines 13 and 24, we go to line 30 for detecting its interestingness and re-initializing required Boolean variables.

8. Experimental studies

We have carried out several experiments for mining generalized notches in different databases. All the experiments have been implemented on a 1.6 GHz Pentium IV with 256 MB of memory using visual C++ (version 6.0) software. We present experimental results using three real and one synthetic databases. Databases *mushroom* [12], *retail* [12] and *BMS-WebView-1* are real. Database *BMS-Web-Wiew-1* can be found from KDD CUP 2000 [12]. The dataset *T10I4D100K* [12] was

generated synthetically using a generator developed by IBM Almaden Quest research group. The characteristics of these databases are given in Table 1.

Table 1. Database characteristics

Database	*NT*	*ALT*	*AFI*	*NI*
mushroom (*M*)	8124	24.000	1624.800	120
retail (*R*)	88,162	11.306	99.674	10,000
BMS-WebView-1 (*B*)	1,49,639	2.000	44.575	6,714
T10I4D100K (*T*)	1,00,000	11.102	1276.124	870

The symbols used in Tables 1, 2a, 2b, 2c, and 2d come with the following meaning: *D*, *NT*, *ALT*, *AFI*, and *NI* denote database, the number of transactions, average length of a transaction, average frequency of an item, and number of items, respectively. The databases *mushroom* and *retail* have been divided into 10 sub-databases, called yearly databases, for the purpose of conducting experiments. The databases *BMS-WebView*-1 and *T10I4D100K* have been divided into 20 sub-databases. The databases obtained from *mushroom* and *retail* are named as M_i and R_i respectively, i = 0, 1, …, 9. The databases obtained from *BMS-WebView*-1 and *T10I4D100K* are named as B_i and T_i respectively, i = 0, 1, …, 19. We present some characteristics of the input databases in Tables 2a, 2b, 2c, and 2d.

Table 2a. Characteristics of the time databases generated from *mushroom*

D	*NT*	*ALT*	*AFI*	*NI*	*D*	*NT*	*ALT*	*AFI*	*NI*
M_0	812	24.000	295.273	66	M_5	812	24.000	221.454	88
M_1	812	24.000	286.588	68	M_6	812	24.000	216.533	90
M_2	812	24.000	249.846	78	M_7	812	24.000	191.059	102
M_3	812	24.000	282.435	69	M_8	812	24.000	229.271	85
M_4	812	24.000	259.840	75	M_9	816	24.000	227.721	86

Table 2b. Characteristics of the time databases generated from *retail*

D	*NT*	*ALT*	*AFI*	*NI*	*D*	*NT*	*ALT*	*AFI*	*NI*
R_0	9000	11.244	12.070	8384	R_5	9000	10.856	16.710	5847
R_1	9000	11.209	12.265	8225	R_6	9000	11.200	17.416	5788
R_2	9000	11.337	14.597	6990	R_7	9000	11.155	17.346	5788
R_3	9000	11.490	16.663	6206	R_8	9000	11.997	18.690	5777
R_4	9000	10.957	16.039	6148	R_9	7162	11.692	15.348	5456

Table 2c. Characteristics of the time databases generated from *BMS-WebView-1*

D	*NT*	*ALT*	*AFI*	*NI*	*D*	*NT*	*ALT*	*AFI*	*NI*
B_0	7482	2.000	5.016	2983	B_{10}	7482	2.000	4.573	3272
B_1	7482	2.000	4.494	3330	B_{11}	7482	2.000	4.895	3057
B_2	7482	2.000	5.782	2588	B_{12}	7482	2.000	4.636	3228
B_3	7482	2.000	4.359	3433	B_{13}	7482	2.000	4.805	3114
B_4	7482	2.000	4.228	3539	B_{14}	7482	2.000	4.192	3570
B_5	7482	2.000	4.194	3568	B_{15}	7482	2.000	4.656	3214
B_6	7482	2.000	3.786	3952	B_{16}	7482	2.000	5.379	2782
B_7	7482	2.000	3.477	4304	B_{17}	7482	2.000	4.863	3077
B_8	7482	2.000	4.168	3590	B_{18}	7482	2.000	4.654	3215
B_9	7482	2.000	4.365	3428	B_{19}	7481	2.000	4.953	3021

In Tables 3-6 we have presented upward generalized notches using '*u*' and downward generalized notches using '*d*'. In *mushroom*, there are many items having high frequency as it is relatively dense. Also, we get many generalized notches having relatively large height as shown in Table 3. On the other hand, the items in *retail* is somewhat skewed in the sense that some generalized notches for few items have large height. In *mushroom* and *retail* the generalized notches are relatively wider. The variation of sales over the years for an item in *mushroom* and *retail* is higher.

Table 2d. Characteristics of the time databases generated from *T10I4D100K*

D	*NT*	*ALT*	*AFI*	*NI*	*D*	*NT*	*ALT*	*AFI*	*NI*
T_0	5000	11.123	64.968	856	T_{10}	5000	11.113	64.913	856
T_1	5000	10.987	63.880	860	T_{11}	5000	11.165	64.988	859
T_2	5000	11.189	65.128	859	T_{12}	5000	11.127	64.617	861
T_3	5000	11.078	64.330	861	T_{13}	5000	11.089	64.694	857
T_4	5000	11.003	63.895	861	T_{14}	5000	11.169	65.088	858
T_5	5000	11.131	64.867	858	T_{15}	5000	11.028	64.338	857
T_6	5000	11.171	64.645	864	T_{16}	5000	11.132	64.795	859
T_7	5000	11.075	64.764	855	T_{17}	5000	11.031	64.661	853
T_8	5000	11.123	65.121	854	T_{18}	5000	11.072	64.374	860
T_9	5000	11.151	64.755	861	T_{19}	5000	11.090	64.856	855

Also, we observe that many upward generalized notch is followed by a downward generalized notch and vice versa. This is because of the fact that two consecutive

different types (a ‘u’ type followed by a ‘d’ type or a ‘d’ type followed by an ‘u’ type) generalized notches share a common leg. For example, a ‘u’ type generalized notch at year 4 is followed by a ‘d’ type generalized notch at year 7, for item 116 in *mushroom*. Also, we observe that some items have both long height and width. For example, item 56 has height 796 and width 8. These values are significantly high as compare to other items in the time databases. Also, this is true for item 67. In *BMS-WebView*-1 database items 333469 and 110877 have maximum variation. Therefore, only these two items are appearing among top ten generalized notches and their heights vary from 344 to 506. Similarly, items 966, 998 and 419 in *T10I4D100K* share common legs and they have more variations. From Table 6 one could conclude that generalized notches in *BMS-WebView*-1 are more sharpened than that of *T10I4D100K*.

We have also reported an execution time with respect to the number of data sources. We observe in Figures 2-5 that the execution time increases as the number of databases increases. We observe that the execution time of *retail* is significantly larger than that of other databases, since each of the time databases is relatively larger. In Figures 4 and 5 we have considered same α and β for *BMS-WebView-1* and *T10I4D100K* databases, respectively. But execution time of *BMS-WebView-1* is significantly larger than that of *T10I4D100K*.

In Figures 6-13 we have presented how the number of interesting icebergs decreases with respect to the increase of α and β. In *mushroom*, *ALT* and *AFI* are higher as compared to the parameters of other databases. We illustrate the graphs using two representative figures viz, Figure 6 and 7. We start changing α values from 100 (Figure 6). Initially, the number of icebergs decreases significantly. Afterwards, the decrease is not so significant. In Figure 7, the number of icebergs decreases significantly when the width remains lower. Afterwards, the number of icebergs decreases slowly.

9. Conclusions

The study of temporal patterns in time-stamped databases has not been attended well. In this paper, we have proposed new patterns in time-stamped databases. First, we have introduced the notion of notch in a sales series of an item. It represents a basic type of patterns in time-stamped databases. Based on this pattern, we have introduced two more patterns viz., generalized notch and iceberg notch, in sales series of an item. Iceberg notch represents a special sales pattern of an item over time. It could be considered as an exceptional pattern in time-stamped databases. The investigations of such patterns could be important to understand the purchasing behaviour of customers or identifying an external influence. The presence of an iceberg helps identifying the reasons for such changes in sales series. We have designed an algorithm to extract icebergs in time-stamped databases and presented experimental results for real-world and synthetic time-stamped databases.

Table 3. Top 10 generalized notches in *M*, and *R* databases (according to height)

M (α) at β = 4				R (α) at β = 2			
item	*year*(sales)	*type*	*height*	*item*	*year*(sales)	*type*	*height*
116	4(1489)	*u*	1344	41	4(2617)	*u*	2617
116	7(353)	*d*	1136	41	9(2355)	*u*	2355
114	2(1131)	*u*	1062	0	2(2331)	*d*	1315
114	4(69)	*d*	1062	48	9(4544)	*u*	932
56	5(810)	*u*	796	48	4(4704)	*u*	711
56	9(14)	*d*	796	0	3(2403)	*u*	609
67	6(103)	*d*	704	0	4(1794)	*d*	609
94	5(2)	*d*	650	0	7(1619)	*u*	571
94	9(652)	*u*	650	8978	2(556)	*u*	556
1	3(69)	*d*	644	0	5(2089)	*u*	512

Table 4. Top 10 generalized notches in *B* and *T* databases (according to height)

B (α) at β = 3				T (α) at β = 2			
item	*year* (sales)	*type*	*height*	*item*	*year* (sales)	*type*	*height*
333469	9(582)	*u*	506	966	4(297)	*d*	122
333469	12(76)	*d*	506	966	6(419)	*u*	122
333469	4(151)	*d*	388	998	2(346)	*u*	103
333469	6(539)	*u*	388	998	6(243)	*d*	103
333449	9(474)	*u*	379	966	8(395)	*u*	93
333449	11(95)	*d*	379	966	10(302)	*d*	93
333449	4(148)	*d*	372	829	16(431)	*u*	90
333449	6(520)	*u*	372	966	11(389)	*u*	87
110877	5(353)	*u*	344	419	4(179)	*d*	85
110877	10(9)	*d*	344	419	6(264)	*u*	85

Table 5. Top 10 generalized notches (according to width) at a given α

M (β) at α = 300				R (β) at α = 20			
item	*year*(sales)	*type*	*width*	*item*	*year*(sales)	*type*	*width*
56	5(810)	*u*	8	9823	7(30)	*u*	9
11	4(427)	*u*	7	2046	4(133)	*u*	8
13	6(39)	*d*	7	3321	7(24)	*u*	8
16	5(361)	*u*	7	411	4(32)	*u*	8
67	6(103)	*d*	7	3121	4(0)	*d*	8
94	5(2)	*d*	7	2919	6(32)	*u*	8
98	2(404)	*u*	7	103	7(267)	*u*	7
11	8(32)	*d*	6	855	3(118)	*u*	7
52	6(703)	*u*	6	1659	3(116)	*u*	7
53	6(109)	*d*	6	976	6(55)	*d*	7

Table 6. Top 10 generalized notches (according to width) at a given α

$B(\beta)$ at $\alpha = 50$				$T(\beta)$ at $\alpha = 5$			
item	*year* (sales)	*type*	*width*	*item*	*year* (sales)	*type*	*width*
112551	9(6)	*d*	10	673	8(71)	*d*	8
335213	10(4)	*d*	10	651	11(82)	*u*	8
112339	5(224)	*u*	9	524	4(29)	*u*	8
335185	4(130)	*u*	9	283	15(229)	*u*	7
112407	5(107)	*u*	9	487	15(139)	*d*	7
335181	5(54)	*u*	9	336	13(42)	*d*	7
335213	5(54)	*u*	9	523	11(117)	*u*	7
335177	5(51)	*u*	9	658	14(88)	*d*	7
110877	5(353)	*u*	8	733	16(63)	*u*	7
110315	11(310)	*u*	8	807	10(29)	*u*	7

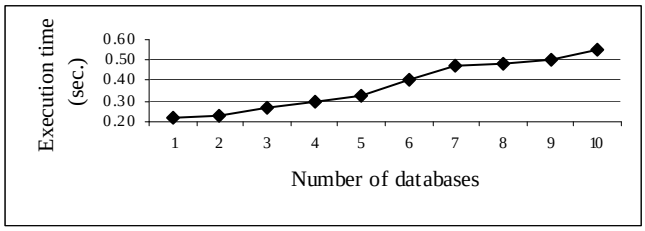

Figure 2. Execution time vs. no. of databases (*mushroom* at $\alpha = 50$, $\beta = 3$)

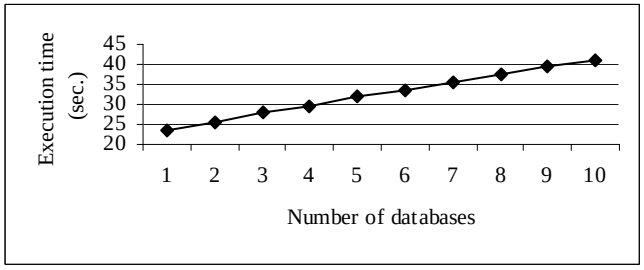

Figure 3. Execution time vs. no. of databases (*retail* at $\alpha = 20$, $\beta = 2$)

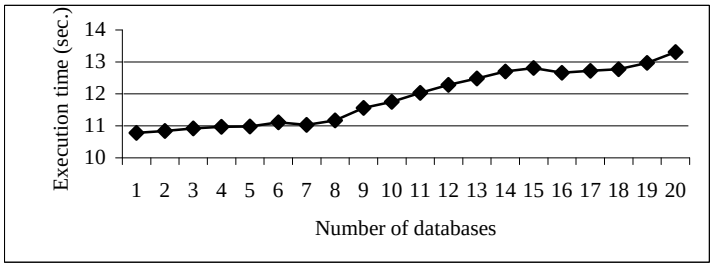

Figure 4. Execution time vs. no. of databases (*BMS-WebView-1* at $\alpha = 30$, $\beta = 4$)

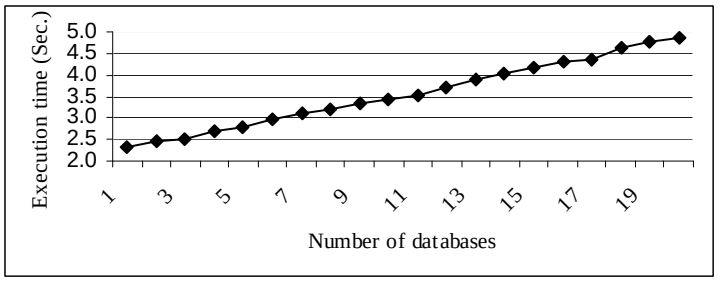

Figure 5. Execution time vs. number of databases (*T10I4D100K* at α =30, β = 4)

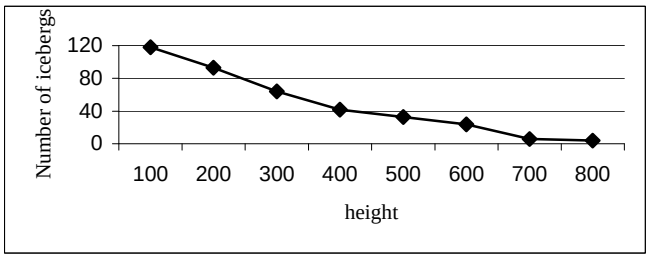

Figure 6. Number of interesting icebergs vs. α for *mushroom* (β = 3)

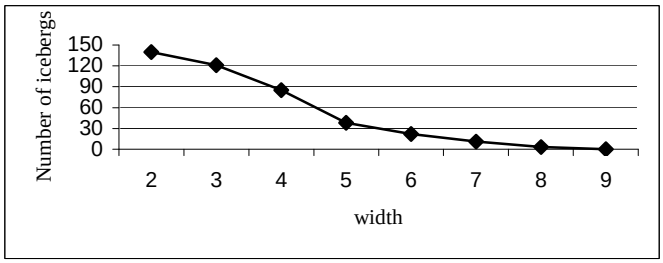

Figure 7. Number of interesting icebergs vs. β for *mushroom* (α = 50)

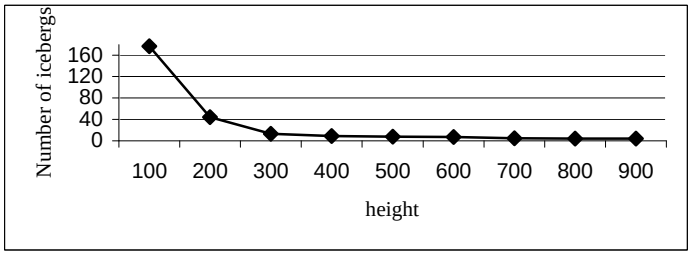

Figure 8. Number of interesting icebergs vs. α for *retail* (β = 2)

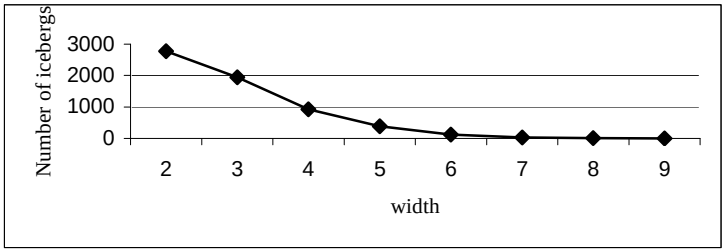

Figure 9. Number of interesting icebergs vs. β for *retail* (α = 20)

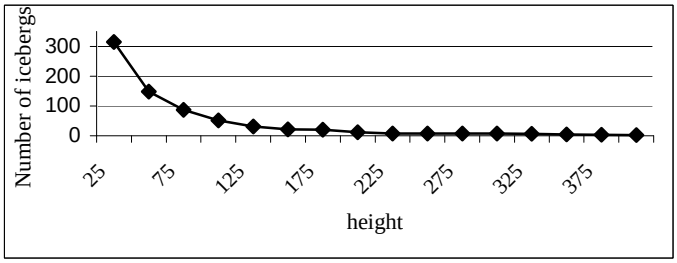

Figure 10. Number of interesting icebergs vs. α for *BMS-WebView-1* (β = 4)

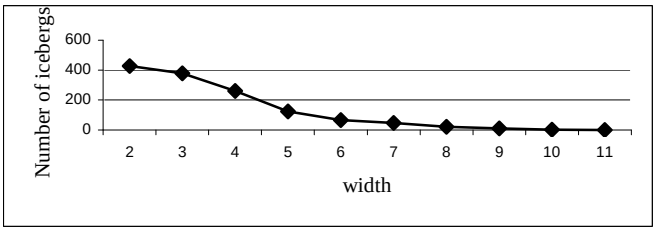

Figure 11. Number of interesting icebergs vs. β for *BMS-WebView-1* (α = 30)

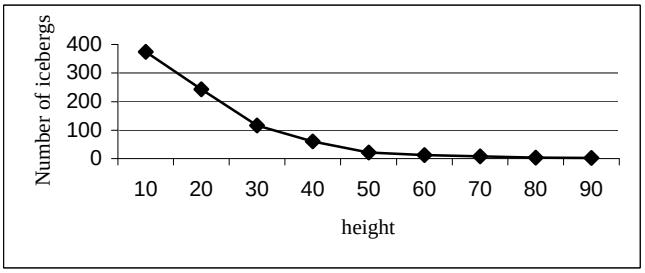

Figure 12. Number of interesting icebergs vs.α for *T10I4D100K* (β = 5)

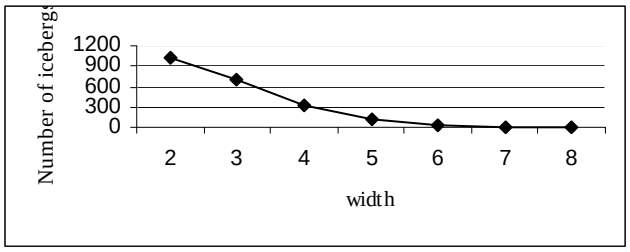

Figure 13. Number of interesting icebergs vs. β for *T10I4D100K* (α = 30)

Acknowledgement

The first author would like to thank the University Grants Commission (UGC), Government of India, for having sponsored her on faculty improvement programme with leave to take up full time research.

References

1. Adhikari, A., Rao, P.R.: Mining conditional patterns in a database, Pattern Recognition Letters, 1515--1523 (2008)
2. Adhikari, J., Rao, P.R., Adhikari, A.: Clustering items in different data sources induced by stability, The International Arab Journal of Information Technology, pp. 394--402 (2009)
3. Adhikari, J., Rao, P.R.: Measuring influence of an item in a database over time. Pattern Recognition Letters 31, pp. 179--187 (2010)
4. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of ACM SIGMOD Conference On Management of Data, pp. 207--216 (1993)
5. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of 20th Very Large databases (VLDB) Conference, pp. 487--499 (1994)
6. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of International Conference On Data Engineering, pp. 3--14 (1995)
7. Böttcher, M., Hoppner, F., Spiliopoulou,: On Exploiting the Power of Time in Data Mining, SIGKDD Explorations 10, pp. 3--11(2008)
8. Box. G., Jenkins, G., Reinsel, G.: Time series analysis, 3rd ed., Pearson Edu. (2003)
9. Brockwell, J., Richard, D.: Introduction to time series and forecasting. Springer (2002)
10. Castellana, L., D'Addabbo, A., Pasquariello, G.: A composed supervised / unsupervised approach to improve change detection from remote sensing. Pattern Recognition Lett. 28, pp. 405--413 (2007)
11. Fink, E., Gandhi, H.S.: Important extrema of time series. SMC, pp. 366--372 (2007)
12. Frequent itemset mining dataset repository, http://fimi.cs.helsinki.fi/data
13. Han, J., Pei, J., Yiwen, Y.: Mining frequent patterns without candidate generation. In: Proceedings of ACM SIGMOD Conference On Management of Data 1--12 (2000)

14. Han, J., Kamber, M., Data mining: Concepts and techniques. Morgan Kauffmann (2001)
15. Hong, T. P., Wu, Y. Y., Wang, S. L.: An effective mining approach for up-to-date patterns. Expert Systems with Applications, 36, 9747-9752, (2009)
16. Keogh, E.: A fast and robust method for pattern matching in time series databases. Proceedings of 9th Internat. Conf. on tools with AI (ICTAI). pp. 578--584 (1997)
17. Keogh, E., Lonardi, S., Chiu, B.: Finding surprising patterns in a time series database in linear time and space, SIGKDD. Canada, pp. 23--26 (2002)
18. Keogh, E.,, Lin J., Ada Wai-Chee Fu: HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. ICDM, 226—233 (2005)
19. Keogh, E., Lin, J., Lee, S. H., Herle, H. V.: Finding the most unusual time series subsequence: algorithms and applications. Knowledge and Inform. Systems 11, pp.1--27 (2006)
20. Leonard, M., Wolfe, B.: Mining transactional and time series data. In: Proceedings of the SUGI 30, pp. 18--24 (2005)
21. Li, Y., P. Ning, X.S. Wang, S. Jajodia. Discovering calendar-based temporal association rules. *Data and Knowledge Engineering*, 44(2): 193-218, (2003)
22. Li, D., Deogun, J. S.,: Discovering partial periodic sequential association rules with time lag in multiple sequences for predicton. Lecture notes in Computer Sciences, 3488, 332--341 (2005)
23. Lomet, D. B., Hong, M., Nehme, R. V., Zhang, R.: Transaction time indexing with version compression, PVLDB 1, pp. 870--881 (2008)
24. Mahanta, A.K., Mazarbhuiya, F.A., Baruah, H.K.: Finding calendar-based periodic patterns. Pattern Recognition Letters 29, pp. 1274--1284 (2008)
25. Perng, C.S., et al., Landmarks: A new model for similarity-based pattern querying in time series databases, Proceedings of the 16th International Conference on Data Engineering, pp. 33--42, (2000)
26. Pratt, K., Fink, E.: Search for patterns in compressed time series, Internat. J. of Image and Graphics 2, pp. 89--106 (2002)
27. Roddick, J.F., Spillopoulou, M.: A Bibliography of temporal, spatial and spatio-temporal data mining Research. ACM SIGKDD 1(1), pp. 34--38 (1999)
28. Singh, S., Stuart. E.: A pattern matching tool for time series forecasting. Proceedings of 14th Internat. Conf. on Pattern Recognition, Brisbane, pp. 103-105 (1998)
29. Tsay, R.S.: Identifying multivariate time series models. J. Time Series and Analysis (10) 357--372 (1989)
30. Wang, L., Chng, E.S., Li, H.: A tree-construction search approach for multivariate time series motifs discovery, Pattern Recognition Lett., 31, pp. 869-875 (2010)
31. Wu, X., Zhang , C., Zhang, S.: Efficient mining of both positive and negative association rules. ACM Transactions on Information Systems 22(3) 381--405 (2004)