Mining Icebergs in Time-Stamped Databases

Jhimli Adhikari¹, P R Rao², Witold Pedrycz³

¹Department of Computer Science, Narayan Zantye College, Bicholim, Goa - 403 529, India
jhimli_adhikari@yahoo.co.in

²Department of Computer Science and Technology, Goa University, Goa - 403 206, India
pralhadrao@gmail.com

³Department 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.

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 \( i = 1, 2, ..., k \). 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

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

(1)

The change of sales series at year \( i \) is decreasing if

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

(2)

The change of sales series at year \( i \) is altering if

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

(3)

or,

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

(4)

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

\[\text{Proceedings of the Fifth Indian International Conference on Artificial Intelligence}\]
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

646
(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:

\[
\text{left-height}(A, i) = |\text{sales}(A, i_1) - \text{sales}(A, i)|, \quad \text{and} \\
\text{right-height}(A, i) = |\text{sales}(A, i) - \text{sales}(A, i_2)|
\]

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 \(\alpha\), and (ii) The width of the generalized notch is greater than or equal to \(\beta\). Both \(\alpha\) and \(\beta\) are user-defined thresholds.

An iceberg notch is a generalized notch having a larger width and a larger 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 \(\alpha\) and \(\beta\) be set to 300 and 5, respectively. Also, let the values of \(i_1\) and \(i_2\) be 1975, 1972, and 1978, respectively (with respect to Figure 1). We observe that \(\text{leftHeight}(A, 1975) = (\text{sales}(A, 1972) - \text{sales}(A, 1975)) = |905 - 537| = 368\) and \(\text{rightHeight}(A, 1975) = |\text{sales}(A, 1975) - \text{sales}(A, 1978)| = |537 - 820| = 283\), respectively. Therefore, the height of the iceberg is maximum \{368, 283\} i.e. 368 \(\geq\) \(\alpha\). Also, \(i_2 - i_1 = (1978 - 1972)\)
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:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TS(A)</td>
<td>–1</td>
<td>+1</td>
<td>–1</td>
<td>–1</td>
<td>–1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>–1</td>
<td></td>
</tr>
</tbody>
</table>

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 > δ. We put –1 in the ternary series of the item corresponding to year 1971, if x – y > δ. We put 0 in the ternary series of the item corresponding to year 1971, if |x – y| ≤ δ. 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 \((k, F, \alpha, \beta)\)

**Inputs:** \( k \), \( F \), \( \alpha \), \( \beta \)  
\( k \): number of yearly databases  
\( F \): array of frequencies of items in yearly databases  
\( \alpha \): user-defined threshold of height  
\( \beta \): user-defined threshold of width  

**Outputs:** Interesting icebergs in time databases

01: let index = 1;  
02: for \( i = 1 \) to \( n \) do  
03: \hspace{1em} let \( j = 2, left = 2, flat = f \);  
04: \hspace{2em} prevDown = \( f \), prevUp = \( f \);  
05: \hspace{1em} while not end of the \( i \)-th sales series do  

if there is downward trend and prevUp is $f$ then
find mid, leftWidth; let prevDown = $t$;
end if {05}
if there is upward trend and prevDown is $t$ then
find left, mid, right, leftWidth, rightWidth;
let prevDown = $f$; leftHeight = rightHeight;
compute rightHeight;
GN(index).type = down; go to 30;
end if {09}
if there is upward trend and prevDown is $t$ then
find left, mid, right, leftWidth, rightWidth;
let prevDown = $f$; leftHeight = rightHeight;
compute rightHeight;
GN(index).type = up; go to 30;
end if {15}
if sales of $j$-th and $(j+1)$-th year are same then
find left; let flat = $t$, prevDown = $f$, prevUp = $f$;
go to 04;
end if
if flat is $f$ then
compute height as defined in Definition 5;
if current notch satisfies criteria $\alpha$ and $\beta$ then
store it in GN(index); increase index by 1;
end if
if the current generalized notch is downward then
let prevDown = $f$, prevUp = $t$;
else if current generalized notch is upward then
let prevDown = $t$; let prevUp = $f$;
end if
end if
end if {35}
end if {30}
end while {04}
end for {02}
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
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 × *k*). Again, the time complexity of line 44 can not be
more than *O*(n × *k*), since the number of interesting icebergs is always less than n × *k*.

**Theorem 1.** Correctness 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-WebView-
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

<table>
<thead>
<tr>
<th>Database</th>
<th>NT</th>
<th>ALT</th>
<th>AFI</th>
<th>NI</th>
</tr>
</thead>
<tbody>
<tr>
<td>mushroom (M)</td>
<td>8124</td>
<td>24.000</td>
<td>1624.800</td>
<td>120</td>
</tr>
<tr>
<td>retail (R)</td>
<td>88,162</td>
<td>11.306</td>
<td>99.674</td>
<td>10,000</td>
</tr>
<tr>
<td>BMS-WebView-1 (B)</td>
<td>1,49,639</td>
<td>2.000</td>
<td>44.575</td>
<td>6,714</td>
</tr>
<tr>
<td>T10I4D100K (T)</td>
<td>1,00,000</td>
<td>11.102</td>
<td>1276.124</td>
<td>870</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>D</th>
<th>NT</th>
<th>ALT</th>
<th>AFI</th>
<th>NI</th>
<th>D</th>
<th>NT</th>
<th>ALT</th>
<th>AFI</th>
<th>NI</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_0</td>
<td>812</td>
<td>24.000</td>
<td>295.273</td>
<td>66</td>
<td>M_3</td>
<td>812</td>
<td>24.000</td>
<td>221.454</td>
<td>88</td>
</tr>
<tr>
<td>M_1</td>
<td>812</td>
<td>24.000</td>
<td>286.588</td>
<td>68</td>
<td>M_6</td>
<td>812</td>
<td>24.000</td>
<td>216.533</td>
<td>90</td>
</tr>
<tr>
<td>M_2</td>
<td>812</td>
<td>24.000</td>
<td>249.846</td>
<td>78</td>
<td>M_7</td>
<td>812</td>
<td>24.000</td>
<td>191.059</td>
<td>102</td>
</tr>
<tr>
<td>M_3</td>
<td>812</td>
<td>24.000</td>
<td>282.435</td>
<td>69</td>
<td>M_8</td>
<td>812</td>
<td>24.000</td>
<td>229.271</td>
<td>85</td>
</tr>
<tr>
<td>M_4</td>
<td>812</td>
<td>24.000</td>
<td>259.840</td>
<td>75</td>
<td>M_9</td>
<td>816</td>
<td>24.000</td>
<td>227.721</td>
<td>86</td>
</tr>
</tbody>
</table>

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

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

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

652
Table 2c. Characteristics of the time databases generated from BMS-WebView-1

<table>
<thead>
<tr>
<th>D</th>
<th>NT</th>
<th>ALT</th>
<th>AFI</th>
<th>NI</th>
<th>D</th>
<th>NT</th>
<th>ALT</th>
<th>AFI</th>
<th>NI</th>
</tr>
</thead>
<tbody>
<tr>
<td>R₀</td>
<td>9000</td>
<td>11.244</td>
<td>12.070</td>
<td>8384</td>
<td>R₅</td>
<td>9000</td>
<td>10.856</td>
<td>16.710</td>
<td>5847</td>
</tr>
<tr>
<td>R₁</td>
<td>9000</td>
<td>11.209</td>
<td>12.265</td>
<td>8225</td>
<td>R₆</td>
<td>9000</td>
<td>11.200</td>
<td>17.416</td>
<td>5788</td>
</tr>
<tr>
<td>R₂</td>
<td>9000</td>
<td>11.337</td>
<td>14.597</td>
<td>6990</td>
<td>R₇</td>
<td>9000</td>
<td>11.155</td>
<td>17.346</td>
<td>5788</td>
</tr>
<tr>
<td>R₃</td>
<td>9000</td>
<td>11.490</td>
<td>16.663</td>
<td>6206</td>
<td>R₈</td>
<td>9000</td>
<td>11.997</td>
<td>18.690</td>
<td>5777</td>
</tr>
<tr>
<td>R₄</td>
<td>9000</td>
<td>10.957</td>
<td>16.039</td>
<td>6148</td>
<td>R₀</td>
<td>7482</td>
<td>2.000</td>
<td>3.847</td>
<td>3003</td>
</tr>
<tr>
<td>B₀</td>
<td>7482</td>
<td>2.000</td>
<td>5.016</td>
<td>2983</td>
<td>B₅</td>
<td>7482</td>
<td>2.000</td>
<td>4.573</td>
<td>3272</td>
</tr>
<tr>
<td>B₁</td>
<td>7482</td>
<td>2.000</td>
<td>5.782</td>
<td>2588</td>
<td>B₁₀</td>
<td>7482</td>
<td>2.000</td>
<td>4.636</td>
<td>3228</td>
</tr>
<tr>
<td>B₂</td>
<td>7482</td>
<td>2.000</td>
<td>4.359</td>
<td>3433</td>
<td>B₁₁</td>
<td>7482</td>
<td>2.000</td>
<td>4.805</td>
<td>3114</td>
</tr>
<tr>
<td>B₃</td>
<td>7482</td>
<td>2.000</td>
<td>3.786</td>
<td>3952</td>
<td>B₁₂</td>
<td>7482</td>
<td>2.000</td>
<td>4.192</td>
<td>3570</td>
</tr>
<tr>
<td>B₄</td>
<td>7482</td>
<td>2.000</td>
<td>4.194</td>
<td>3568</td>
<td>B₁₃</td>
<td>7482</td>
<td>2.000</td>
<td>4.565</td>
<td>3214</td>
</tr>
<tr>
<td>B₅</td>
<td>7482</td>
<td>2.000</td>
<td>3.786</td>
<td>3952</td>
<td>B₁₄</td>
<td>7482</td>
<td>2.000</td>
<td>5.379</td>
<td>2782</td>
</tr>
<tr>
<td>B₆</td>
<td>7482</td>
<td>2.000</td>
<td>4.168</td>
<td>3590</td>
<td>B₁₅</td>
<td>7482</td>
<td>2.000</td>
<td>4.654</td>
<td>3215</td>
</tr>
<tr>
<td>B₇</td>
<td>7482</td>
<td>2.000</td>
<td>4.365</td>
<td>3428</td>
<td>B₁₆</td>
<td>7482</td>
<td>2.000</td>
<td>4.953</td>
<td>3021</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>D</th>
<th>NT</th>
<th>ALT</th>
<th>AFI</th>
<th>NI</th>
<th>D</th>
<th>NT</th>
<th>ALT</th>
<th>AFI</th>
<th>NI</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₀</td>
<td>5000</td>
<td>11.123</td>
<td>64.968</td>
<td>856</td>
<td>T₁₀</td>
<td>5000</td>
<td>11.113</td>
<td>64.913</td>
<td>856</td>
</tr>
<tr>
<td>T₁</td>
<td>5000</td>
<td>10.987</td>
<td>63.880</td>
<td>860</td>
<td>T₁₁</td>
<td>5000</td>
<td>11.165</td>
<td>64.988</td>
<td>859</td>
</tr>
<tr>
<td>T₂</td>
<td>5000</td>
<td>11.189</td>
<td>65.128</td>
<td>859</td>
<td>T₁₂</td>
<td>5000</td>
<td>11.127</td>
<td>64.617</td>
<td>861</td>
</tr>
<tr>
<td>T₃</td>
<td>5000</td>
<td>11.078</td>
<td>64.330</td>
<td>861</td>
<td>T₁₃</td>
<td>5000</td>
<td>11.089</td>
<td>64.694</td>
<td>857</td>
</tr>
<tr>
<td>T₄</td>
<td>5000</td>
<td>11.003</td>
<td>63.895</td>
<td>861</td>
<td>T₁₄</td>
<td>5000</td>
<td>11.169</td>
<td>65.088</td>
<td>858</td>
</tr>
<tr>
<td>T₅</td>
<td>5000</td>
<td>11.131</td>
<td>64.867</td>
<td>858</td>
<td>T₁₅</td>
<td>5000</td>
<td>11.028</td>
<td>64.338</td>
<td>857</td>
</tr>
<tr>
<td>T₆</td>
<td>5000</td>
<td>11.171</td>
<td>64.645</td>
<td>864</td>
<td>T₁₆</td>
<td>5000</td>
<td>11.132</td>
<td>64.795</td>
<td>859</td>
</tr>
<tr>
<td>T₇</td>
<td>5000</td>
<td>11.075</td>
<td>64.764</td>
<td>855</td>
<td>T₁₇</td>
<td>5000</td>
<td>11.031</td>
<td>64.661</td>
<td>853</td>
</tr>
<tr>
<td>T₈</td>
<td>5000</td>
<td>11.123</td>
<td>65.121</td>
<td>854</td>
<td>T₁₈</td>
<td>5000</td>
<td>11.072</td>
<td>64.374</td>
<td>860</td>
</tr>
<tr>
<td>T₉</td>
<td>5000</td>
<td>11.151</td>
<td>64.755</td>
<td>861</td>
<td>T₁₉</td>
<td>5000</td>
<td>11.090</td>
<td>64.856</td>
<td>855</td>
</tr>
</tbody>
</table>

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 $\alpha$ and $\beta$ 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 $\alpha$ and $\beta$. 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 $\alpha$ 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)
<table>
<thead>
<tr>
<th>Item</th>
<th>Year (Sales)</th>
<th>Type</th>
<th>Height</th>
<th>Item</th>
<th>Year (Sales)</th>
<th>Type</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>116</td>
<td>4(1489)</td>
<td>u</td>
<td>1344</td>
<td>41</td>
<td>4(2617)</td>
<td>u</td>
<td>2617</td>
</tr>
<tr>
<td>116</td>
<td>7(353)</td>
<td>d</td>
<td>1136</td>
<td>41</td>
<td>9(2355)</td>
<td>u</td>
<td>2355</td>
</tr>
<tr>
<td>114</td>
<td>2(1314)</td>
<td>u</td>
<td>1062</td>
<td>0</td>
<td>2(2331)</td>
<td>d</td>
<td>1315</td>
</tr>
<tr>
<td>114</td>
<td>4(69)</td>
<td>d</td>
<td>1062</td>
<td>48</td>
<td>9(4544)</td>
<td>u</td>
<td>932</td>
</tr>
<tr>
<td>56</td>
<td>5(810)</td>
<td>u</td>
<td>796</td>
<td>48</td>
<td>4(4704)</td>
<td>u</td>
<td>711</td>
</tr>
<tr>
<td>56</td>
<td>9(14)</td>
<td>d</td>
<td>796</td>
<td>0</td>
<td>3(2403)</td>
<td>d</td>
<td>609</td>
</tr>
<tr>
<td>67</td>
<td>6(103)</td>
<td>d</td>
<td>704</td>
<td>0</td>
<td>4(1794)</td>
<td>d</td>
<td>609</td>
</tr>
<tr>
<td>94</td>
<td>5(2)</td>
<td>d</td>
<td>650</td>
<td>0</td>
<td>7(1619)</td>
<td>u</td>
<td>571</td>
</tr>
<tr>
<td>94</td>
<td>9(652)</td>
<td>u</td>
<td>650</td>
<td>0</td>
<td>2(556)</td>
<td>u</td>
<td>556</td>
</tr>
<tr>
<td>1</td>
<td>3(69)</td>
<td>d</td>
<td>644</td>
<td>0</td>
<td>5(2089)</td>
<td>u</td>
<td>512</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Year (Sales)</th>
<th>Type</th>
<th>Height</th>
<th>Item</th>
<th>Year (Sales)</th>
<th>Type</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>333469</td>
<td>9(582)</td>
<td>u</td>
<td>506</td>
<td>966</td>
<td>4(297)</td>
<td>d</td>
<td>122</td>
</tr>
<tr>
<td>333469</td>
<td>12(76)</td>
<td>d</td>
<td>506</td>
<td>966</td>
<td>6(419)</td>
<td>u</td>
<td>122</td>
</tr>
<tr>
<td>333469</td>
<td>4(151)</td>
<td>d</td>
<td>388</td>
<td>998</td>
<td>2(346)</td>
<td>u</td>
<td>103</td>
</tr>
<tr>
<td>333469</td>
<td>6(539)</td>
<td>u</td>
<td>388</td>
<td>998</td>
<td>6(243)</td>
<td>d</td>
<td>103</td>
</tr>
<tr>
<td>333449</td>
<td>9(474)</td>
<td>u</td>
<td>379</td>
<td>966</td>
<td>8(395)</td>
<td>u</td>
<td>93</td>
</tr>
<tr>
<td>333449</td>
<td>11(95)</td>
<td>d</td>
<td>379</td>
<td>966</td>
<td>10(302)</td>
<td>d</td>
<td>93</td>
</tr>
<tr>
<td>333449</td>
<td>4(148)</td>
<td>d</td>
<td>372</td>
<td>829</td>
<td>16(431)</td>
<td>u</td>
<td>90</td>
</tr>
<tr>
<td>333449</td>
<td>6(520)</td>
<td>u</td>
<td>372</td>
<td>966</td>
<td>11(389)</td>
<td>u</td>
<td>87</td>
</tr>
<tr>
<td>110877</td>
<td>5(353)</td>
<td>u</td>
<td>344</td>
<td>419</td>
<td>4(179)</td>
<td>d</td>
<td>85</td>
</tr>
<tr>
<td>110877</td>
<td>10(9)</td>
<td>d</td>
<td>344</td>
<td>419</td>
<td>6(264)</td>
<td>u</td>
<td>85</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Item</th>
<th>Year (Sales)</th>
<th>Type</th>
<th>Width</th>
<th>Item</th>
<th>Year (Sales)</th>
<th>Type</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>56</td>
<td>5(810)</td>
<td>u</td>
<td>8</td>
<td>9823</td>
<td>7(30)</td>
<td>u</td>
<td>9</td>
</tr>
<tr>
<td>11</td>
<td>4(427)</td>
<td>u</td>
<td>7</td>
<td>2046</td>
<td>4(133)</td>
<td>u</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>6(39)</td>
<td>d</td>
<td>7</td>
<td>3321</td>
<td>7(24)</td>
<td>u</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>5(361)</td>
<td>u</td>
<td>7</td>
<td>411</td>
<td>4(32)</td>
<td>u</td>
<td>8</td>
</tr>
<tr>
<td>67</td>
<td>6(103)</td>
<td>d</td>
<td>7</td>
<td>3121</td>
<td>4(0)</td>
<td>d</td>
<td>8</td>
</tr>
<tr>
<td>94</td>
<td>5(2)</td>
<td>d</td>
<td>7</td>
<td>2919</td>
<td>6(32)</td>
<td>u</td>
<td>8</td>
</tr>
<tr>
<td>98</td>
<td>2(404)</td>
<td>u</td>
<td>7</td>
<td>103</td>
<td>7(267)</td>
<td>u</td>
<td>7</td>
</tr>
<tr>
<td>11</td>
<td>8(32)</td>
<td>d</td>
<td>6</td>
<td>855</td>
<td>3(118)</td>
<td>u</td>
<td>7</td>
</tr>
<tr>
<td>52</td>
<td>6(703)</td>
<td>u</td>
<td>6</td>
<td>1659</td>
<td>3(116)</td>
<td>u</td>
<td>7</td>
</tr>
<tr>
<td>53</td>
<td>6(109)</td>
<td>d</td>
<td>6</td>
<td>976</td>
<td>6(55)</td>
<td>d</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 6. Top 10 generalized notches (according to width) at a given α
<table>
<thead>
<tr>
<th>item</th>
<th>year (sales)</th>
<th>type</th>
<th>width</th>
<th>item</th>
<th>year (sales)</th>
<th>type</th>
<th>width</th>
</tr>
</thead>
<tbody>
<tr>
<td>112551</td>
<td>9(6)</td>
<td>d</td>
<td>10</td>
<td>673</td>
<td>8(71)</td>
<td>d</td>
<td>8</td>
</tr>
<tr>
<td>335213</td>
<td>10(4)</td>
<td>d</td>
<td>10</td>
<td>651</td>
<td>11(82)</td>
<td>u</td>
<td>8</td>
</tr>
<tr>
<td>112339</td>
<td>5(224)</td>
<td>u</td>
<td>9</td>
<td>524</td>
<td>4(29)</td>
<td>u</td>
<td>8</td>
</tr>
<tr>
<td>335185</td>
<td>4(130)</td>
<td>u</td>
<td>9</td>
<td>283</td>
<td>15(229)</td>
<td>u</td>
<td>7</td>
</tr>
<tr>
<td>112407</td>
<td>5(107)</td>
<td>u</td>
<td>9</td>
<td>487</td>
<td>15(139)</td>
<td>d</td>
<td>7</td>
</tr>
<tr>
<td>335181</td>
<td>5(54)</td>
<td>u</td>
<td>9</td>
<td>336</td>
<td>13(42)</td>
<td>d</td>
<td>7</td>
</tr>
<tr>
<td>335213</td>
<td>5(54)</td>
<td>u</td>
<td>9</td>
<td>523</td>
<td>11(117)</td>
<td>u</td>
<td>7</td>
</tr>
<tr>
<td>335177</td>
<td>5(51)</td>
<td>u</td>
<td>9</td>
<td>658</td>
<td>14(88)</td>
<td>d</td>
<td>7</td>
</tr>
<tr>
<td>110877</td>
<td>5(353)</td>
<td>u</td>
<td>8</td>
<td>733</td>
<td>16(63) u</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>110315</td>
<td>11(310)</td>
<td>u</td>
<td>8</td>
<td>807</td>
<td>10(29)</td>
<td>u</td>
<td>7</td>
</tr>
</tbody>
</table>

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 $\alpha=30$, $\beta=4$)

Figure 6. Number of interesting icebergs vs. $\alpha$ for mushroom ($\beta=3$)

Figure 7. Number of interesting icebergs vs. $\beta$ for mushroom ($\alpha=50$)

Figure 8. Number of interesting icebergs vs. $\alpha$ for retail ($\beta=2$)
Figure 9. Number of interesting icebergs vs. $\beta$ for retail ($\alpha = 20$)

Figure 10. Number of interesting icebergs vs. $\alpha$ for BMS-WebView-1 ($\beta = 4$)

Figure 11. Number of interesting icebergs vs. $\beta$ for BMS-WebView-1 ($\alpha = 30$)

Figure 12. Number of interesting icebergs vs. $\alpha$ for T10I4D100K ($\beta = 5$)
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