10-2003

Indexing of Time Series by Major Minima and Maxima

Eugene Fink
University of South Florida

Kevin B. Pratt
Computer Science Innovations

Harith Suman Gandhi
University of South Florida

Follow this and additional works at: http://repository.cmu.edu/compsci

Published In
Indexing of Time Series by Major Minima and Maxima

Eugene Fink
Computer Science and Eng.
University of South Florida
Tampa, Florida 33620
eugene@csee.usf.edu

Kevin B. Pratt
Computer Science Innovations
1235 Evans Road
Melbourne, Florida 32904
kpratt@csi-inc.com

Harith Suman Gandhi
Computer Science and Eng.
University of South Florida
Tampa, Florida 33620
g_reddy@tvratings.com

Abstract – We describe a technique for fast compression of time series and indexing of compressed series. We have tested it on three data sets: stock prices, air and sea temperatures, and wind speeds.

Keywords: Compression, indexing, retrieval.

Introduction: A time series is a sequence of real values measured at equal intervals. We describe an algorithm for compressing a time series by extracting its major minima and maxima, and then present a technique for indexing a database of compressed series, which supports retrieval of series similar to a given pattern. We have tested it on three data sets, summarized in Table 1.

Stock prices: We have used stocks from the Standard and Poor’s 100 listing for the period from January 1998 to April 2000. We have discarded the newly listed and de-listed stocks, and used prices of ninety-eight stocks.

Air and sea temperatures: We have experimented with daily temperature readings by sixty-eight buoys in the Pacific Ocean, from 1980 to 1998, downloaded from an archive at the University of California at Irvine (kdd.ics.uci.edu).

Wind speeds: We have used daily wind speeds at twelve sites in Ireland, from 1961 to 1978, obtained from an archive at Carnegie Mellon University (lib.stat.cmu.edu/datasets).

Previous work: Researchers have considered a variety of techniques for compressing time series. In particular, Perng and his colleagues investigated a compression procedure based on extraction of “landmark points,” which included minima and maxima [9]. Keogh and Pazzani used the endpoints of best-fit line segments to compress a series [6]. Keogh and his colleagues reviewed and compared the compression techniques based on approximation of a series by a sequence of line segments [5].

Researchers have also studied various methods for indexing of time series. In particular, Chan and Fu combined wavelet transforms with R-trees [2]. Bozkaya and Özyooylu used vantage-point trees to index series by numeric features [1]. Park, Lee, and Chi indexed series by their minima and maxima [8]. Li, Yu, and Castelli proposed a retrieval technique based on a multi-level abstraction hierarchy of features [7]. We have developed a new technique for indexing of compressed series by their minima and maxima [4, 10].

Compression: We compress a time series by selecting its major minima and maxima, and dropping the other points, as shown in Figures 1(a,b). We use a positive parameter $R$ to control the compression rate; an increase of $R$ leads to selection of fewer points.

A point $a[m]$ of a series $a[1..n]$ is a major minimum if there are indices $i$ and $j$, where $i < m < j$, such that

- $a[m]$ is a minimum among $a[i..j]$, and
- $a[i] - a[m] \geq R$ and $a[j] - a[m] \geq R$.

Intuitively, $a[m]$ is a minimal value of some segment $a[i..j]$, and the endpoint values of this segment are much larger than $a[m]$. Similarly, $a[m]$ is a major maximum if there are indices $i$ and $j$, where $i < m < j$, such that

- $a[m]$ is a maximum among $a[i..j]$, and
- $a[m] - a[i] \geq R$ and $a[m] - a[j] \geq R$.

In Figure 3, we give a procedure for identifying major minima and maxima, which performs one pass through a series; it takes linear time and constant memory. We have implemented it in Visual Basic 6 and tested on a 2.4-GHz Pentium computer; for an $n$-point series, it takes $1.8 \cdot n$ microseconds.

Indexing: The indexing of a time-series database is based on the notion of major inclines, illustrated in Figure 1(c). A segment $a[i..j]$ is a major upward incline if

- $a[i]$ is a major minimum;
- $a[j]$ is a major maximum;
- for every $m \in [i..j]$, $a[i] \leq a[m] \leq a[j]$.

The definition of a major downward incline is symmetric. The length of a major incline $a[i..j]$ is $j - i$, and its height is $a[j] - a[i]$; note that the height of an upward incline is positive, whereas the height of a downward incline is negative. We identify all major inclines of all series in the database, and index these inclines by their length and height using a range tree, which is a standard structure for indexing points by two co-ordinates [3].

In Figure 4, we give a procedure for identifying the major upward inclines of a series; the procedure for the
major downward inclines is similar. We denote the number of major minima in the series by $a$, and the number of major maxima by $v$. The procedure inputs a list of major minima and a list of major maxima, and identifies the major upward inclines. First, for every major maximum $a[i][\text{Max}[m]]$, it identifies the next larger maximum and stores its number in $\text{next}[m]$. Second, it uses this information to identify the major inclines. Its running time is linear in the number of major inclines.

**Retrieval:** The retrieval procedure inputs a pattern series and searches for similar segments in a database (Figure 5). First, it finds the pattern’s major incline with the greatest absolute value of a height; we denote the length of this incline by $l$, and its height by $h$. Second, it retrieves all major inclines in the database that have a similar length and height; an incline is considered similar if its length is between $l/C$ and $l \cdot C$, and its height is between $h/D$ and $h \cdot D$, where $C$ and $D$ are parameters for controlling the retrieval process. Third, it identifies the segments that contain the selected inclines (Figure 2), applies a given similarity test to compare them with the pattern, and outputs the segments that pass this test.

**Experiments:** To evaluate the retrieval accuracy, we have compared the retrieval results with the matches identified by a slow exhaustive search. We have ranked the matches found by the retrieval procedure from the most to the least similar. In Figures 6 and 7, we plot the ranks of matches found by the fast procedure versus the ranks of the exhaustive-search matches. For instance, if the fast procedure has found only three among the seven closest matches, the graph includes the point (3, 7). If the fast procedure has found all matches, the graph is a forty-five degree line; otherwise, it is steeper.

We have measured the retrieval speed of a Visual-Basic implementation on a 2.4-GHz Pentium computer. Note that, if we increase $C$ and $D$, the procedure misses fewer matches, but the retrieval is slower. In Figures 6 and 7, we give time measurements for different retrieval accuracies. For the stock database with 60,000 points, the retrieval takes from 0.01 to 0.31 seconds. For the database of air and sea temperatures, which includes 450,000 points, the time is between 0.1 and 1.2 seconds.

For the wind-speed database with 80,000 points, the time varies from 0.08 to 0.20 seconds.

**Acknowledgments:** We are grateful to Dmitry Goldgof, Rafael Perez, Mark Last, and Eamonn Keogh for their comments and suggestions.

**References**


Table 1: Data sets used in the experiments.

<table>
<thead>
<tr>
<th>data set</th>
<th>description</th>
<th>num. of series</th>
<th>points per series</th>
<th>total num. of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>stock prices</td>
<td>98 stocks, 2.3 years</td>
<td>98</td>
<td>610</td>
<td>60,000</td>
</tr>
<tr>
<td>air and sea</td>
<td>68 buoys, 18 years, 2 sensors per buoy</td>
<td>136</td>
<td>1,800-6,600</td>
<td>450,000</td>
</tr>
<tr>
<td>wind speeds</td>
<td>12 stations, 18 years</td>
<td>12</td>
<td>6,570</td>
<td>80,000</td>
</tr>
</tbody>
</table>

**COMPRESSION**

The procedure inputs a time series \( a[1..n] \), and outputs the indices of major minima and maxima.

\[
i = \text{FIND-FIRST} \\
\text{if } i < n \text{ and } a[i] > a[1] \text{ then } i = \text{FIND-MAX}(i) \\
\text{while } i < n \text{ do} \\
\quad i = \text{FIND-MIN}(i) \\
\quad i = \text{FIND-MAX}(i)
\]

**FIND-FIRST**

Finding the first major minimum or maximum.

\[
iMin = 1; \ iMax = 1; \ i = 2 \\
\text{while } i < n \text{ and } a[i] - a[iMin] < R \text{ and } a[iMax] - a[i] < R \text{ do} \\
\quad \text{if } a[i] < a[iMin] \text{ then } iMin = i \\
\quad \text{if } a[i] > a[iMax] \text{ then } iMax = i \\
\quad i = i + 1 \\
\text{if } i < n \text{ and } iMin < iMax \text{ then output } iMin \\
\text{if } i < n \text{ and } iMax < iMin \text{ then output } iMax \\
\text{return } i
\]

**FIND-MIN(i)**

Finding the first major minimum after the \( i \)th point.

\[
iMin = i \\
\text{while } i < n \text{ and } a[i] - a[iMin] < R \text{ do} \\
\quad \text{if } a[i] < a[iMin] \text{ then } iMin = i \\
\quad i = i + 1 \\
\text{if } i < n \text{ or } a[iMin] < a[i] \text{ then output } iMin \\
\text{return } i
\]

**FIND-MAX(i)**

Finding the first major maximum after the \( i \)th point.

\[
iMax = i \\
\text{while } i < n \text{ and } a[iMax] - a[i] < R \text{ do} \\
\quad \text{if } a[i] > a[iMax] \text{ then } iMax = i \\
\quad i = i + 1 \\
\text{if } i < n \text{ or } a[iMax] > a[i] \text{ then output } iMax \\
\text{return } i
\]

Figure 3: Compression procedure. It processes a time series \( a[1..n] \) and identifies the major minima and maxima.

**UPWARD-INCLINES**

The procedure inputs a series \( a[1..n] \), the indices of the major minima in this series, \( iMin[1..u] \), and the indices of the major maxima, \( iMax[1..v] \); it outputs the major upward inclines.

initialize an empty stack \( S \) 
\( \text{PUSH}(S, 1) \) 
\( \text{for } m = 2 \text{ to } v \text{ do} \)
\( \quad \text{while } S \text{ is not empty and } a[\text{MAX}(\text{TOP}(S))] \leq a[\text{MAX}(m)] \text{ do} \)
\( \quad \quad \text{next}(\text{TOP}(S)) = m \)
\( \quad \quad \text{POP}(S) \)
\( \quad \text{PUSH}(S, m) \)
\( \quad \text{while } S \text{ is not empty do} \)
\( \quad \quad \text{next}(\text{TOP}(S)) = v + 1 \)
\( \quad \quad \text{POP}(S) \)
\( k = 1; \ m = 1 \)
\( \text{while } k \leq u \text{ and } m \leq v \text{ do} \)
\( \quad \text{while } m \leq v \text{ and } \text{MAX}(m) < \text{MIN}(k) \text{ do} \)
\( \quad \quad m = m + 1 \)
\( \quad \quad w = m \)
\( \quad \text{while } w \neq v + 1 \text{ do} \)
\( \quad \quad \text{output incline } (\text{MIN}(k), \text{MAX}(w)) \)
\( \quad \quad w = \text{next}(w) \)
\( k = k + 1 \)

Figure 4: Identification of the major upward inclines. We assume that the major minima in a series are numbered from 1 to \( u \), and the major maxima are numbered from 1 to \( v \).

**RETRIEVAL**

The procedure inputs a pattern series, and identifies the segments in a time-series database that match the pattern.

Identify the pattern’s major incline with the greatest absolute value of a height.

Determine the length \( l \) and height \( h \) of this incline.

Retrieve all major inclines in the database such that
- their lengths are between \( l/C \) and \( l \cdot C \), and
- their heights are between \( h/D \) and \( h \cdot D \).

For every retrieved incline:
- Identify the corresponding segment (Figure 2).
- Compare this segment with the pattern, using a given similarity test.

Figure 5: Search for segments similar to a given pattern. The procedure includes two control parameters, \( C \) and \( D \).
Figure 6: Retrieval of stock charts. The horizontal axes show the ranks of matches retrieved by the fast procedure, and the vertical axes are the ranks assigned to the same matches by a slow exhaustive search. We also give the retrieval times.

Figure 7: Retrieval of weather patterns. The horizontal axes show the ranks of matches assigned by the fast procedure, and the vertical axes are the exhaustive-search ranks. In addition, we show the retrieval times.