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 Özsoyoglu used vantage-point trees to index series by numeric features [1]. Park, Lee, and Chu 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 $a[m] - a[j] \geq R$.

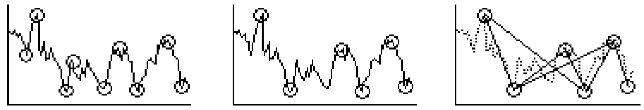
In Figure 3, we give a procedure for identifying major minima and maxima, which performs one pass through a series; it takes linear time and constant memory. We have implemented it in Visual Basic 6 and tested on a 2.4-GHz Pentium computer; for an n-point series, it takes $1.8 \cdot n$ microseconds.

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



(a) Major minima and maxima for 91% compression. (b) Major minima and maxima for 94% compression. (c) Major inclines for 94% compression.

Figure 1: Major minima and maxima.

(a) Identify the pattern incline with the greatest height. (b) Find a similar incline in a series. (c) Align the inclines and compare the segments.

Figure 2: Retrieving a segment that may match the pattern.

major downward inclines is similar. We denote the number of major minima in the series by u, 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 $a[iMax[m]]$, it identifies the next *larger* maximum and stores its number in $next[m]$. 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

[1] Tolga Bozkaya and Z. Meral Özsoyoglu. Indexing large metric spaces for similarity search queries. ACM *Transactions on Database Systems*, 24(3):361–404, 1999.

[2] Kin-Pong Chan and Ada Wai-Chee Fu. Efficient time series matching by wavelets. In *Proceedings of the Fifteenth International Conference on Data Engineering*, pages 126–133, 1999.

[3] Herbert Edelsbrunner. A note on dynamic range searching. *Bulletin of the European Association for Theoretical Computer Science*, 15:34–40, 1981.

[4] Eugene Fink and Kevin B. Pratt. Indexing of compressed time series. In Mark Last, Abraham Kandel, and Horst Bunke, editors, *Data Mining in Time Series Databases*, pages 51–78. World Scientific, Singapore, 2003.

[5] Eamonn J. Keogh, Selina Chu, David Hart, and Michael J. Pazzani. An online algorithm for segmenting time series. In *Proceedings of the* IEEE *International Conference on Data Mining*, pages 289–296, 2001.

[6] Eamonn J. Keogh and Michael J. Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *Proceedings of the Fourth* ACM *International Conference on Knowledge Discovery and Data Mining*, pages 239–243, 1998.

[7] Chung-Sheng Li, Philip S. Yu, and Vittorio Castelli. MALM: A framework for mining sequence database at multiple abstraction levels. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, pages 267–272, 1998.

[8] Sanghyun Park, Sang-Wook Kim, and Wesley W. Chu. Segment-based approach for subsequence searches in sequence databases. In *Proceedings of the Sixteenth* ACM *Symposium on Applied Computing*, pages 248–252, 2001.

[9] Chang-Shing Perng, Haixun Wang, Sylvia R. Zhang, and D. Scott Parker. Landmarks: A new model for similarity-based pattern querying in time series databases. In *Proceedings of the Sixteenth International Conference on Data Engineering*, pages 33–42, 2000.

[10] Kevin B. Pratt and Eugene Fink. Search for patterns in compressed time series. *International Journal of Image and Graphics*, 2(1):89–106, 2002.

Table 1: Data sets used in the experiments.

data set	description	num. of series	points per series	total num. of points
stock prices	98 stocks, 2.3 years	98	610	60,000
air and sea temperatures	68 buoys, 18 years, 2 sensors per buoy	136	1,800–6,600	450,000
wind speeds	12 stations, 18 years	12	6,570	80,000

COMPRESSION

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

i = FIND-FIRST
if $i < n$ and $a[i] > a[1]$ **then** i = FIND-MAX(i)
while $i < n$ **do**
  i = FIND-MIN(i)
  i = FIND-MAX(i)

FIND-FIRST

Finding the first major minimum or maximum.

$iMin = 1$; $iMax = 1$; $i = 2$
while $i < n$ and $a[i] - a[iMin] < R$ and $a[iMax] - a[i] < R$ **do**
  if $a[i] < a[iMin]$ **then** $iMin = i$
  if $a[i] > a[iMax]$ **then** $iMax = i$
  $i = i + 1$
if $i < n$ and $iMin < iMax$ **then output** $iMin$
if $i < n$ and $iMax < iMin$ **then output** $iMax$
return i

FIND-MIN(i)

Finding the first major minimum after the ith point.

$iMin = i$
while $i < n$ and $a[i] - a[iMin] < R$ **do**
  if $a[i] < a[iMin]$ **then** $iMin = i$
  $i = i + 1$
if $i < n$ or $a[iMin] < a[i]$ **then output** $iMin$
return i

FIND-MAX(i)

Finding the first major maximum after the ith point.

$iMax = i$
while $i < n$ and $a[iMax] - a[i] < R$ **do**
  if $a[i] > a[iMax]$ **then** $iMax = i$
  $i = i + 1$
if $i < n$ or $a[iMax] > a[i]$ **then output** $iMax$
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 indicies of the major maxima, $iMax[1..v]$; it outputs the major upward inclines.

initialize an empty stack S
PUSH(S, 1)
for $m = 2$ **to** v **do**
  while S is not empty and $a[iMax[\text{TOP}(S)]] \leq a[iMax[m]]$ **do**
    $next[\text{TOP}(S)] = m$
    POP(S)
  PUSH(S, m)
while S is not empty **do**
  $next[\text{TOP}(S)] = v + 1$
  POP(S)

$k = 1$; $m = 1$
while $k \leq u$ and $m \leq v$ **do**
  while $m \leq v$ and $iMax[m] < iMin[k]$ **do**
    $m = m + 1$
  $w = m$
  while $w \neq v + 1$ **do**
    output incline $(iMin[k], iMax[w])$
    $w = next[w]$
  $k = k + 1$

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.

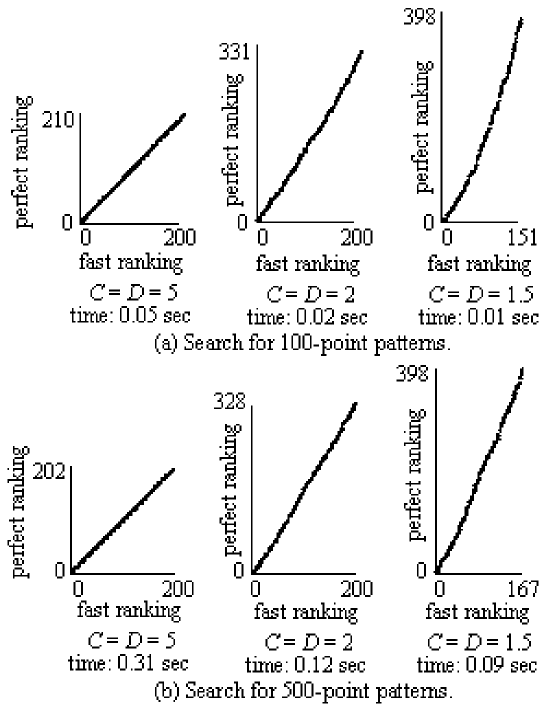


Figure 6: Retrieval of stock charts. The horizontal axes show the ranks of matches retrieved by the fast procedure, and the vertical axes are the ranks assigned to the same matches by a slow exhaustive search. We also give the retrieval times.

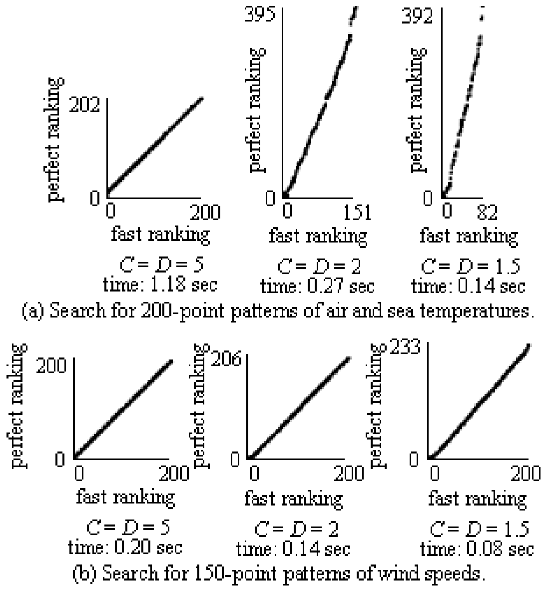


Figure 7: Retrieval of weather patterns. The horizontal axes show the ranks of matches assigned by the fast procedure, and the vertical axes are the exaustive-search ranks. In addition, we show the retrieval times.