International Journal of Image and Graphics

SEARCH FOR PATTERNS IN COMPRESSED TIME SERIES

KEVIN B. PRATT
Harris Corporation
1025 West NASA Boulevard, W3/9708
Melbourne, Florida 32919
kpratt01@harris.com

EUGENE FINK
Computer Science and Engineering
University of South Florida
Tampa, Florida 33620
eugene@csee.usf.edu
www.csee.usf.edu/∼eugene

We describe a technique for fast compression of time series, indexing of compressed series, and retrieval of series similar to a given pattern. The compression procedure identifies "important" points of a series and discards the other points. We use the important points not only for compression, but also for indexing a database of time series. Experiments show the effectiveness of this technique for indexing of stock prices, weather data, and electroencephalograms.

Keywords: Time Series; Compression; Indexing and Retrieval; Similarity Measures.

1. Introduction

We view a *time series* as a sequence of values measured at equal intervals. For example, the series in Figure 1 includes the values 20, 22, 25, 22, 25, and so on. A visual representation of time series helps human experts to analyze temporal data. For instance, traders use stock charts to identify price trends, and doctors analyze electroencephalograms (EEG) to diagnose brain diseases. We present a technique for compression and indexing of time series, which helps to visualize and analyze temporal data.

First, we describe a compression procedure based on extraction of certain important points from a series, which works in linear time and takes constant memory. For example, we can compress the series in Figure 1 by extracting the circled points and discarding the other points.

Second, we give a technique for indexing of compressed series, which supports retrieval of series similar to a given pattern. For example, stock analysts can use it to find instances of a standard pattern in price charts, and doctors can apply it to identify illness patterns in EEG. We have tested this technique on four data sets, summarized in Table 1, which are publicly available through the Internet.

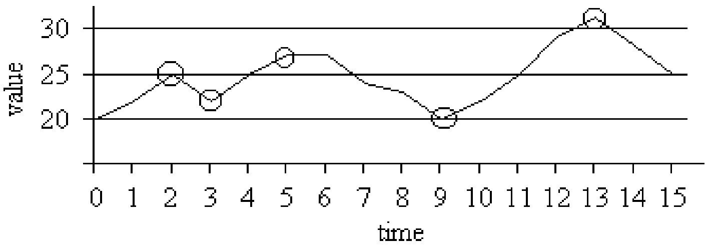


Figure 1: Example of a time series.

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	79,000
EEG	64 electrodes, 1 second	64	256	16,000

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 downloaded daily prices from America Online, discarded newly listed and de-listed stocks, and used ninety-eight stocks in the experiments.

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

Electroencephalograms. We have used EEG obtained by Henri Begleiter at the Neurodynamics Laboratory of the SUNY Health Center at Brooklyn. These data are from sixty-four electrodes at standard points on the scalp; we have copied them from the archive at the University of California at Irvine (`kdd.ics.uci.edu`).

2. Previous Work

We now review related work on the comparison and indexing of time series.

Feature sets. Researchers have considered various feature sets for compressing time series and measuring similarity between them. They have extensively studied Fourier transforms, which allow fast compression;[58,59,60,62] however, this technique has several disadvantages. In particular, it smoothes local minima and maxima, which may lead to a loss of important information, and it does not work well for erratic series.[33] Chan and his colleagues[10,11] applied Haar wavelet transforms[8,25] to time series and showed several advantages of this technique over Fourier transforms.

Guralnik and Srivastava[27] considered the problem of detecting a change in the trend of a data stream, and developed a technique for finding "change points" in a series. Last *et al.*[42] proposed a general framework for knowledge discovery in time series, which included representation of a series by its key features, such as slope and signal-to-noise ratio. They described a technique for computing these features and identifying the points of change in the feature values. We present a procedure for finding important points in a series, which are a special case of change points.

Researchers have also studied the use of small alphabets for compression of series, and applied string matching[29] to the pattern search.[2,32,34,41,49,55] For example, Guralnik *et al.*[28] compressed stock prices using a nine-letter alphabet. Singh and McAtackney[59] represented stock prices, particle dynamics, and stellar light intensity by a three-letter alphabet. Lin and Risch[45] used a two-letter alphabet to encode major spikes in a series. Das *et al.*[15] utilized an alphabet of primitive shapes for efficient compression. These techniques give a high compression rate, but their descriptive power is limited, which makes them inapplicable in many domains.

Perng *et al.*[52] investigated a compression technique based on extraction of "landmark points," which included local minima and maxima. Keogh and Pazzani[36] used the endpoints of best-fit line segments to compress a series. Keogh *et al.*[38] reviewed and compared the compression techniques based on approximation of a time series by a sequence of straight segments. We describe an alternative compression technique, based on selection of important minima and maxima, and give a fast procedure for finding important points.

Similarity measures. Several researchers have defined similarity as the distance in a feature space. For example, Caraca-Valente and Lopez-Chavarrias[9] used Euclidean distance between feature vectors containing angle of knee movement and muscle strength, and Lee *et al.*[43] applied Euclidean distance to compare feature vectors containing color, texture, and shape of video data. This technique works well when all features have the same units of scale;[24] however, it is often ineffective for combining disparate features.

An alternative definition of similarity is based on bounding rectangles; two series are similar if their bounding rectangles are similar. This measure allows fast pruning of clearly dissimilar series,[43,52] but it is less effective for selecting the most similar series among close candidates.

The envelope-count technique is based on dividing a series into short segments, called envelopes, and defining a yes/no similarity for each envelope. Two series are similar within an envelope if their point-by-point differences are within a certain threshold. The overall similarity is measured by the number of envelopes where the series are similar.[3] This measure allows fast computation of similarity, and it can be adapted for noisy and missing data.[4,14]

Finally, we can measure a point-by-point similarity of two series and aggregate these measures, which often requires interpolation to obtain missing points. Keogh and Pazzani[36] used linear interpolation with this technique, and Perng *et al.*[52] applied cubic approximation. Keogh and Pazzani[37] also described a point-by-point similarity with modified Euclidean distance, which does not require interpolation.

Indexing and retrieval. Researchers have studied a variety of techniques for indexing of time series. They have utilized several advanced structures, such as *kd*-trees, R-trees, and grids. For example, Deng[16] applied *kd*-trees to arrange series by their significant features, Chan and Fu[10] combined wavelet transforms with R-trees, and Bozkaya and her colleagues[5,6] used vantage-point trees for indexing series by numeric features.

Park *et al.*[51] indexed series by their local extrema and by properties of the segments between consecutive extrema. Li *et al.*[44] proposed a retrieval technique based on a multi-level abstraction hierarchy of features. Aggarwal and Yu[1] considered grids, but found that their performance in a multi-dimensional space is often no better than exhaustive search. They also reviewed the use of compression with exhaustive-search retrieval, and concluded that exhaustive search among compressed series is often faster than sophisticated indexing. We describe a new technique for indexing of compressed series and show that it allows fast retrieval of series similar to a given pattern.

3. Important Points

We compress a time series by selecting some of its minima and maxima, called *important points,* and dropping the other points (see Figure 2). The intuitive idea is to discard minor fluctuations and keep major minima and maxima. We control the compression rate with a parameter R, which is always greater than one; an increase of R leads to selection of fewer points. A point a_m of a series $a_1, ..., a_n$ is an *important minimum* if there are indices i and j, where $i < m < j$, such that

- a_m is the minimum among $a_i, ..., a_j$, and
- $a_i/a_m \geq R$ and $a_j/a_m \geq R$.

Intuitively, a_m is the minimal value of some segment $a_i, ..., a_j$, and the endpoint values of this segment are much larger than a_m (see Figure 3). Similarly, a_m is an *important maximum* if there are indices i and j, where $i < m < j$, such that

- a_m is the maximum among $a_i, ..., a_j$, and
- $a_m/a_i \geq R$ and $a_m/a_j \geq R$.

In Figure 4, we give a procedure for selecting important points, which performs one pass through a series; it takes linear time and constant memory. This procedure can process new points as they arrive, without storing the original series. For example, it can compress a live electroencephalogram without waiting until the end of the data collection. Furthermore, it is effective for erratic series, such as stock charts. We have implemented it in Visual Basic 6 and tested on a 300-MHz Pentium computer. For an n-point series, the compression time is $14 \cdot n$ microseconds.

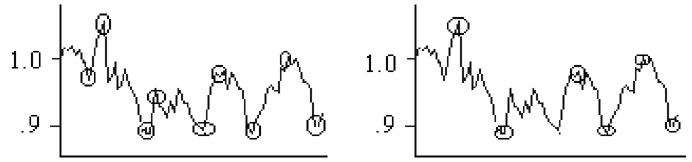


Figure 2: Important points for 91% compression (left) and 94% compression (right).

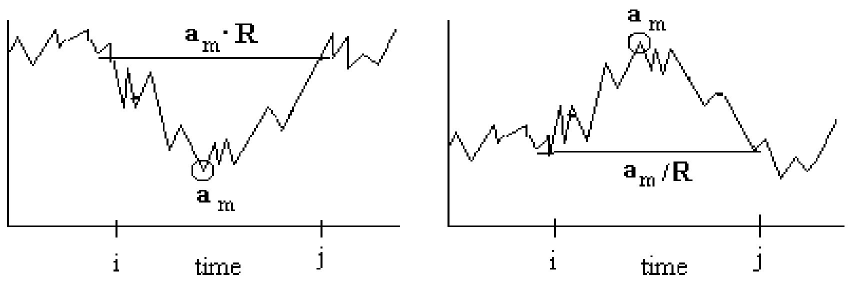


Figure 3: Examples of an important minimum (left) and important maximum (right).

We have applied the compression procedure to the data sets in Table 1, and compared it with two simple techniques: equally spaced points and randomly selected points. We have experimented with different *compression rates,* which are defined as the percentage of points removed from a series. For example, "eighty-percent compression" means that we select 20% of points and discard the other 80%.

For each compression technique, we have measured the difference between the original series and the resulting compressed series. We have considered three measures of difference between the original series, $a_1, ..., a_n$, and the series interpolated from the compressed version, $b_1, ..., b_n$.

Mean difference: $\frac{1}{n} \cdot \sum_{i=1}^{n} |a_i - b_i|$.

Maximum difference: $\max_{i \in [1..n]} |a_i - b_i|$.

Root mean square difference: $\sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (a_i - b_i)^2}$.

We summarize the results in Table 2, which shows that important points are more accurate than the two simple methods.

4. Pattern Retrieval

We begin by defining a measure of similarity between time series, which underlies the retrieval procedure. We measure similarity on a zero-to-one scale; zero means no likeness and one means perfect likeness. First, we define similarity between two numbers, a and b:

$$\mathrm{sim}(a, b) = 1 - \frac{|a - b|}{\max(|a|, |b|)}.$$

IMPORTANT-POINTS — Top-level function for finding important points.
The input is a time series $a_1, ..., a_n$; the output is the
values and indices of the selected important points.

```
i = FIND-FIRST-TWO
if i < n and a_i > a_1 then i = FIND-MINIMUM(i)
while i < n do
    i = FIND-MAXIMUM(i)
    i = FIND-MINIMUM(i)
```

FIND-FIRST-TWO — Find the first and second important points.

```
iMin = 1; iMax = 1
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 iMin < iMax
    then output (a_iMin, iMin); output (a_iMax, iMax)
    else output (a_iMax, iMax); output (a_iMin, iMin)
return i
```

FIND-MINIMUM(i) — Find the first important 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
output (a_iMin, iMin)
return i
```

FIND-MAXIMUM(i) — Find the first important 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
output (a_iMax, iMax)
return i
```

Figure 4: Compression procedure. We process a series $a_1, ..., a_n$ and use a global variable n to represent its size. The procedure outputs the values and indices of important points.

Table 2: Accuracy of three compression techniques. We give the average difference between an original series and its compressed version, using three difference measures; smaller differences correspond to more accurate compression.

	mean difference			**maximum difference**			**root mean square diff.**		
	important points	fixed points	random points	important points	fixed points	random points	important points	fixed points	random points
				eighty percent compression					
stock prices	0.02	0.03	0.04	0.70	1.70	1.60	0.05	0.14	0.14
air temp.	0.01	0.03	0.03	0.33	0.77	0.72	0.03	0.10	0.10
sea temp.	0.01	0.03	0.03	0.35	0.81	0.75	0.03	0.10	0.10
wind speeds	0.02	0.03	0.03	0.04	1.09	1.01	0.04	0.05	0.05
EEG	0.03	0.06	0.07	0.68	1.08	1.00	0.10	0.18	0.17
				ninety percent compression					
stock prices	0.03	0.06	0.07	1.10	1.70	1.70	0.08	0.21	0.21
air temp.	0.02	0.05	0.05	0.64	0.80	0.78	0.08	0.16	0.14
sea temp.	0.01	0.04	0.05	0.60	0.83	0.82	0.07	0.16	0.14
wind speeds	0.03	0.04	0.04	0.06	1.09	1.03	0.05	0.06	0.06
EEG	0.08	0.13	0.12	0.82	1.10	1.09	0.17	0.27	0.24
				ninety-five percent compression					
stock prices	0.05	0.10	0.12	1.30	1.80	1.80	0.11	0.32	0.30
air temp.	0.03	0.09	0.08	0.74	0.83	0.83	0.12	0.23	0.21
sea temp.	0.03	0.08	0.08	0.78	0.85	0.85	0.12	0.23	0.21
wind speeds	0.05	0.04	0.04	0.08	1.09	1.10	0.07	0.08	0.08
EEG	0.13	0.17	0.16	0.90	1.10	1.10	0.24	0.31	0.28

The similarity between two series, $a_1, ..., a_n$ and $b_1, ..., b_n$, is the mean of their point-by-point similarities:

$$\frac{1}{n} \cdot \sum_{i=1}^{n} \mathrm{sim}(a_i, b_i).$$

The main advantage of this similarity measure for the reported work is its effectiveness for compressed series. In Table 3, we show the correlation of the similarity measure with the three difference measures. We have also checked how well the similarity of original series correlates with the similarity of their compressed versions. We give the results in Figure 5, which shows a good linear correlation.

We now give a procedure that inputs a pattern series and retrieves similar series from a database. It includes three main steps: identifying a "prominent feature" of the pattern, finding similar features in the database, and comparing the pattern with each series that has a similar feature.

We begin by defining a *leg* of a time series, which is the segment between two consecutive important points. For each leg, we store the values listed in Figure 6, which are denoted *vl*, *vr*, *il*, *ir*, *ratio*, and *length*; we give an example of these values in Figure 7. The *prominent leg* of a pattern series is the leg with the greatest endpoint ratio.

The retrieval procedure inputs a pattern and searches for similar segments in a database (see Figure 8). First, it finds the pattern leg with the greatest endpoint ratio, denoted $ratio_p$, and identifies all legs in the database that have a similar ratio.

Table 3: Correlation between the similarity and three difference measures. The correlation is negative because greater similarity corresponds to smaller differences.

data set	**correlation** mean diff.	max diff.	root mean square diff.
stock prices	-0.76	-0.65	-0.79
air temperatures	-0.86	-0.93	-0.92
sea temperatures	-0.66	-0.82	-0.73
wind speeds	-0.84	-0.94	-0.89
EEG	-0.93	-0.93	-0.96

A ratio is considered similar if it is between $ratio_p/C$ and $ratio_p \cdot C$, where C is a parameter for controlling the matching process. We index all legs in the database by their ratio, using a red-black tree. If the total number of legs is n, and the number of legs with ratio between $ratio_p/C$ and $ratio_p \cdot C$ is k, then the retrieval time is $O(k + \lg n)$.

After identifying these legs, the procedure discards the legs whose length is not similar to the pattern's prominent leg. The length is considered similar if it is between $length_p/D$ and $length_p \cdot D$, where $length_p$ is the length of the pattern leg, and D is a control parameter. Finally, the procedure identifies the segments that contain the selected legs (see Figure 9), and then computes their similarity to the pattern. If the similarity is above a given threshold T, the procedure outputs the segment as a match. In Figure 10, we give an example of a stock-price pattern and similar segments retrieved from the stock database.

5. Extended Legs

The described procedure can miss a matching segment that does not have a leg corresponding to the pattern's prominent leg. We illustrate this problem in Figure 11, where the prominent leg of the pattern has no equivalent in the matching series. To avoid this problem, we introduce the notion of *extended legs.* Intuitively, a segment is an extended leg if it would be a leg under a higher compression rate (see Figure 11c). Formally, points a_i and a_j of a series $a_1, ..., a_n$ form an extended upward leg if

- a_i is a local minimum, and a_j is a local maximum, and
- for every $m \in [i..j]$, we have $a_i < a_m < a_j$.

The definition of an extended downward leg is similar.

We identify all extended legs of all series in the database and index them in the same way as normal legs. The advantage of this approach is more accurate retrieval, and the disadvantage is a larger indexing structure. In the worst case, an n-leg series can have $n^2/2$ extended legs; however, if series in the database do not have an upward or downward trend, the average number of legs in $O(n \cdot \lg n)$.

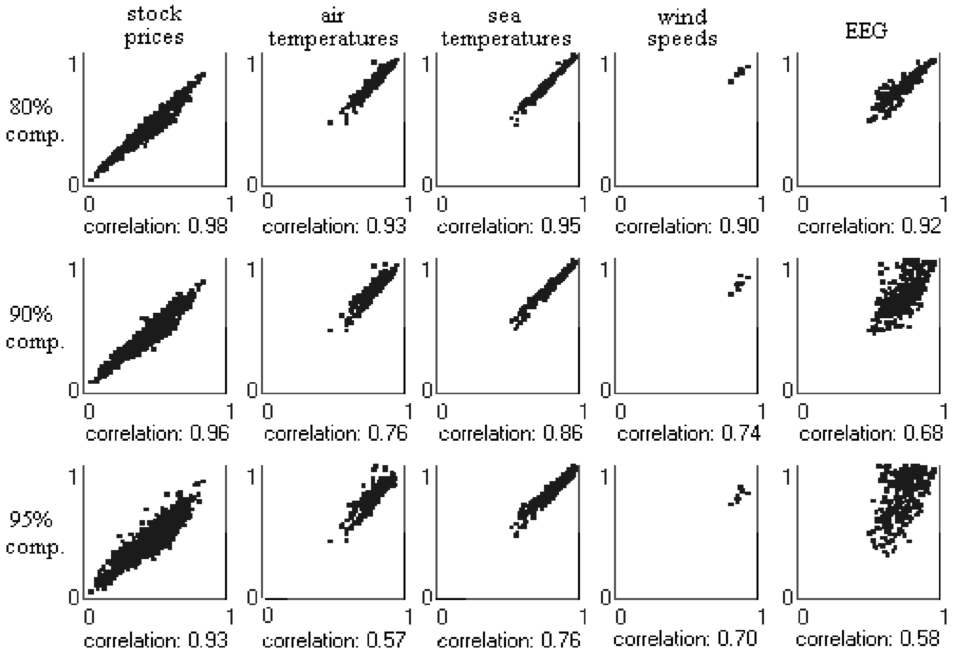


Figure 5: Correlation between the similarity of original series and the similarity of their compressed versions. We consider three compression rates: 80%, 90%, and 95%.

vl	value of the left important point of the leg
vr	value of the right important point of the leg
il	index of the left important point
ir	index of the right important point
ratio	ratio of the endpoints, defined as vr/vl
length	length of the leg, defined as $ir - il$

Figure 6: Basic data for a leg.

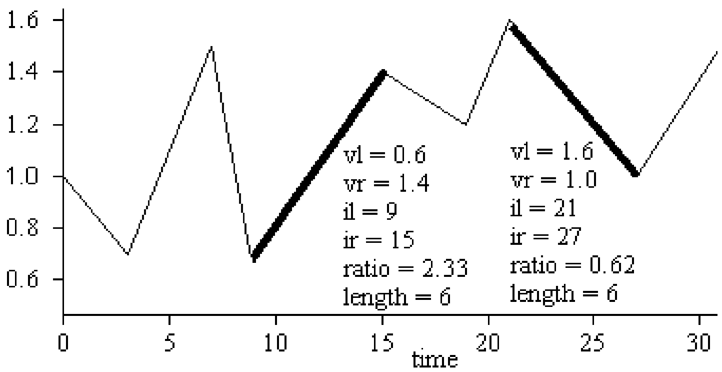


Figure 7: Example legs. We show the basic data for the two legs marked by thick lines.

PATTERN-RETRIEVAL

The procedure inputs a pattern series and searches a time-series database;
its output is a list of segments from the database that match the pattern.

Identify the pattern leg p with the greatest endpoint ratio, denoted $ratio_p$.
Find all legs in the database with endpoint ratio between $ratio_p/C$ and $ratio_p \cdot C$.
For each leg l in the set of selected legs:
    If $length_l < length_p/D$ or $length_l > length_p \cdot D$, then discard l from the set.
For each leg l in the set of remaining legs:
    Identify the segment corresponding to the pattern (see Figure 9).
    Compute the similarity between the segment and the pattern.
    If the similarity is above the threshold T, then output the segment.

Figure 8: Search for segments similar to a given pattern. We use three control parameters: maximal ratio deviation C, maximal length deviation D, and similarity threshold T.

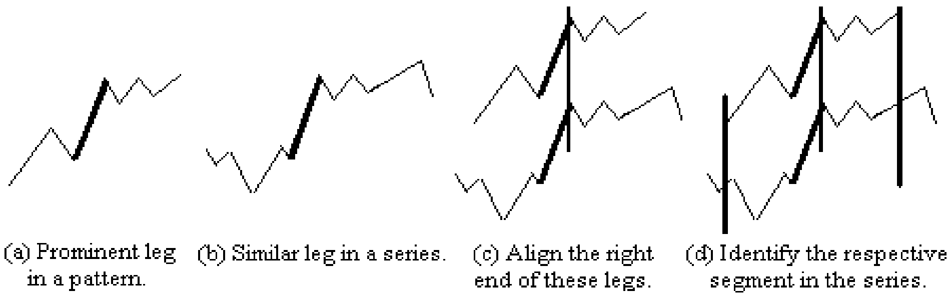


Figure 9: Identifying a segment that may match the pattern.

Figure 10: Example of retrieved stock charts.

(a) Prominent leg of a pattern. (b) Dissimilar legs. (c) Extended legs.

Figure 11: Example of extended legs. The pattern (a) matches the series (b), but the pattern's prominent leg has no equivalent in the series. If we identify the extended legs (c), the prominent leg matches one of them.

EXTENDED-LEGS
The input is the list of legs in a series;
the output is a list of all extended legs.

initialize an empty stack S of leg indices
PUSH$(S, 1)$
for $k = 2$ **to** n **do**
  while S is not empty and $ir_{\text{TOP}(S)} < ir_k$ **do**
    $next[\text{TOP}(S)] = k$; POP(S)
  PUSH(k)
while S is not empty **do**
  $next[\text{TOP}(S)] =$ NIL; POP(S)

initialize an empty list of extended legs
for $k = 1$ **to** $n - 1$ **do**
  $m = next[k]$
  while m is not NIL **do**
    add (il_k, ir_m) to the list of extended legs
    $m = next[m]$

Figure 12: Identifying extended legs. We assume that normal legs are numbered 1 to n.

In Figure 12, we give a procedure for identifying upward extended legs in a series; the procedure for downward legs is similar. We assume that normal upward legs in the input series are numbered from 1 to n. First, the procedure processes important maxima; for each maximum ir_k, it identifies the *next larger* maximum and stores its index in $next[k]$. Second, it uses this information to identify extended legs. The running time of the first part is linear in the number of normal legs, and the time of the second part is linear in the number of extended legs.

To evaluate the retrieval accuracy, we have compared the retrieval results with the results of identifying matches by a slow exhaustive search. We have ranked the matches found by the retrieval procedure from most to least similar. In Figures 13 and 14, we plot the ranks of matches found by the fast procedure versus the ranks of exhaustive-search matches. For instance, if the fast procedure has found only three among seven closest matches, the graph includes the point $(3, 7)$.

The retrieval time grows linearly with the pattern length and with the number of candidate segments identified at the first two steps of the retrieval procedure. If we increase C and D, the procedure finds more candidates and misses fewer matches, but the retrieval takes more time.

We have measured the speed of a Visual Basic implementation on a 300-MHz PC. If the pattern has m legs and the procedure identifies k candidate matches, the retrieval time is $70 \cdot m \cdot k$ microseconds. In Figures 13 and 14, we give time measurements for different retrieval accuracies. For the stock database with 60,000 points, the retrieval takes from 0.1 to 2.5 seconds. For the database of air and sea temperatures, which includes 450,000 points, the time is between 1 and 10 seconds.

6. Concluding Remarks

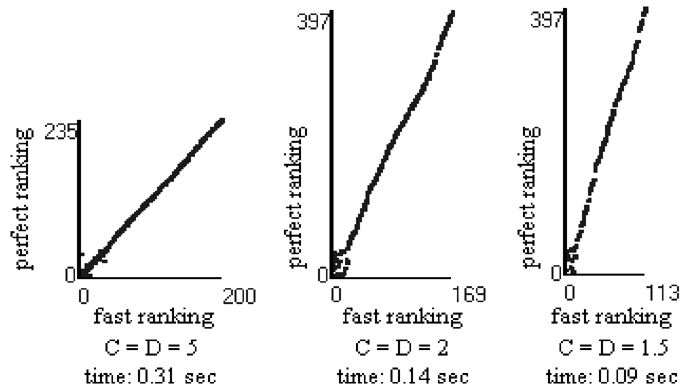
The contributions of the described work include a procedure for compressing time series and its use for indexing and retrieval. The key idea is to index series by their prominent features and retrieve the series whose compressed representation is similar to the compressed pattern. The experiments have shown the effectiveness of this technique for indexing of stock prices, weather data, and electroencephalograms.

Acknowledgements

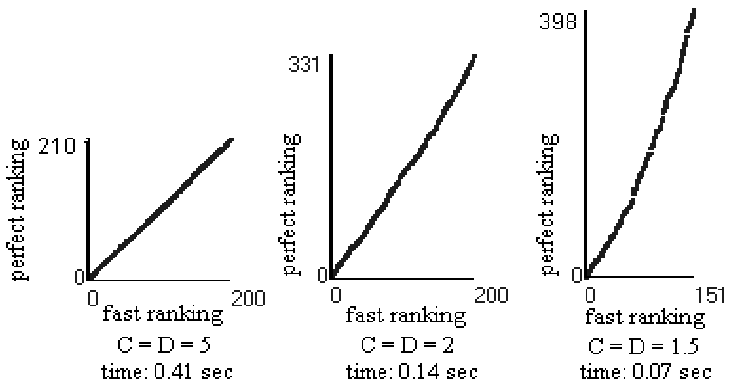
We are grateful to Dmitry Goldgof, Rafael Perez, Mark Last, and Eamonn Keogh for their valuable comments and suggestions.

References

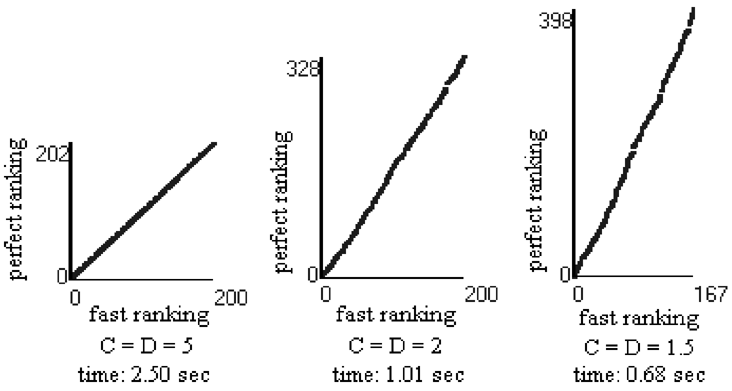
1. Charu C. Aggarwal and Philip S. Yu. The IGrid index: Reversing the dimensionality curse for similarity indexing in high-dimensional space. In *Proceedings of the Sixth ACM International Conference on Knowledge Discovery and Data Mining*, pages 119–129, 2000.
2. Rakesh Agrawal, Giuseppe Psaila, Edward L. Wimmers, and Mohamed Zait. Querying shapes of histories. In *Proceedings of the Twenty-First International Conference on Very Large Data Bases*, pages 502–514, 1995.
3. Rakesh Agrawal, Manish Mehta, John C. Shafer, Ramakrishnan Srikant, Andreas Arning, and Toni Bollinger. The Quest data mining system. In *Proceedings of the Second ACM International Conference on Knowledge Discovery and Data Mining*, pages 244–249, 1996.
4. Bela Bollobas, Gautam Das, Dimitrios Gunopulos, and Heikki Mannila. Time-series similarity problems and well-separated geometric sets. In *Proceedings of the Thirteen Annual Symposium on Computational Geometry*, pages 454–456, 1997.
5. Tolga Bozkaya and Z. Meral Özsoyoglu. Indexing large metric spaces for similarity search queries. *ACM Transactions on Database Systems*, 24(3):361–404, 1999.
6. Tolga Bozkaya, Nasser Yazdani, and Z. Meral Özsoyoglu. Matching and indexing sequences of different lengths. In *Proceedings of the Sixth International Conference on Information and Knowledge Management*, pages 128–135, 1997.
7. Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting*. Springer-Verlag, Berlin, Germany, 1996.



(a) Search for four-leg patterns.

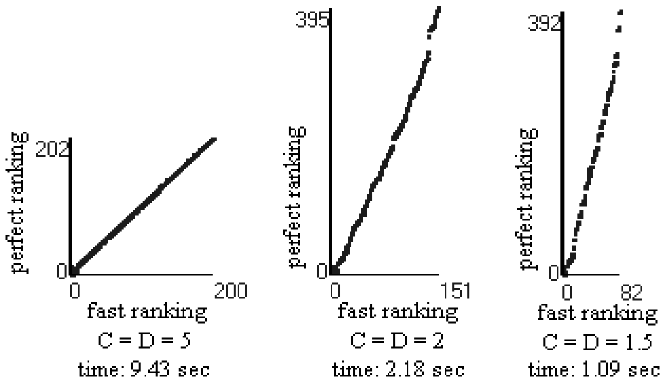


(b) Search for six-leg patterns.

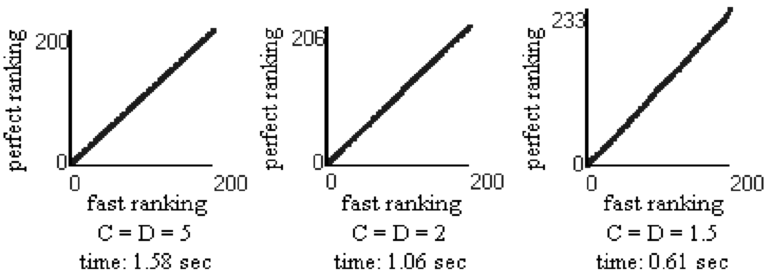


(c) Search for thirty-leg patterns.

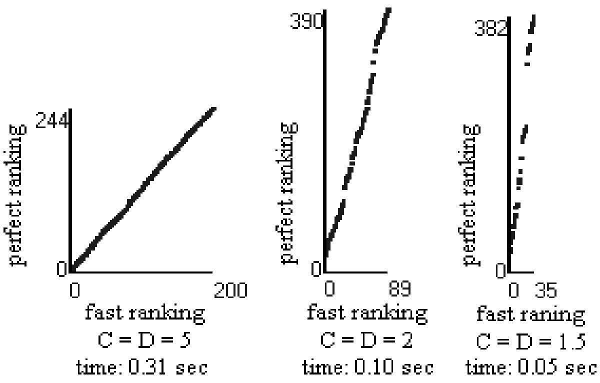
Figure 13: Retrieval of stock charts. The horizontal axes show the ranks of matches retrieved by the fast procedure. The vertical axes are the ranks assigned to the same matches by the exhaustive search. If the fast procedure has found all matches, the graph is a forty-five degree line; otherwise, it is steeper. We also give the retrieval times.



(a) Search for twelve-leg patterns of air and sea temperatures.



(b) Search for nine-leg patterns of wind speeds.



(c) Search for electroencephalogram patterns with twenty legs.

Figure 14: Retrieval of weather and electroencephalogram patterns. The horizontal axes show the similarity ranks assigned by the fast procedure, and the vertical axes are the exaustive-search ranks. In addition, we show the retrieval times.

8. C. Sidney Burrus, Ramesh A. Gopinath, and Haitao Guo. *Introduction to Wavelets and Wavelet Transforms: A Primer*. Prentice Hall, Englewood Cliffs, NJ, 1997.
9. Juan Pedro Caraca-Valente and Ignacio Lopez-Chavarrias. Discovering similar patterns in time series. In *Proceedings of the Sixth ACM International Conference on Knowledge Discovery and Data Mining*, pages 497–505, 2000.
10. 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.
11. Kin-Pong Chan, Ada Wai-Chee Fu, and C. Yu. Haar wavelets for efficient similarity search of time-series: With and without time warping. *IEEE Transactions on Knowledge and Data Engineering*, to appear.
12. Ed Huai-Hsin Chi, Phillip Barry, Elizabeth Shoop, John V. Carlis, Ernest Retzel, and John Riedl. Visualization of biological sequence similarity search results. In *Proceedings of the IEEE Conference on Visualization*, pages 44–51, 1995.
13. Corinna Cortes, Kathleen Fisher, Daryl Pregibon, and Anne Rogers. Hancock: A language for extracting signatures from data streams. In *Proceedings of the Sixth ACM International Conference on Knowledge Discovery and Data Mining*, pages 9–17, 2000.
14. Gautam Das, Dimitrios Gunopulos, and Heikki Mannila. Finding similar time series. In *Proceedings of the First European Conference on Principles of Data Mining and Knowledge Discovery*, pages 88–100, 1997.
15. Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule discovery from time series. In *Proceedings of the Fourth ACM International Conference on Knowledge Discovery and Data Mining*, pages 16–22, 1998.
16. Kan Deng. OMEGA*: On-Line Memory-Based General Purpose System Classifier*. PhD thesis, Robotics Institute, Carnegie Mellon University, 1998. Technical Report CMU-RI-TR-98-33.
17. Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM International Conference on Knowledge Discovery and Data Mining*, pages 71–80, 2000.
18. Herbert Edelsbrunner. A note on dynamic range searching. *Bulletin of the European Association for Theoretical Computer Science*, 15:34–40, 1981.
19. Tony Fountain, Thomas G. Dietterich, and Bill Sudyka. Mining IC test data to optimize VLSI testing. In *Proceedings of the Sixth ACM International Conference on Knowledge Discovery and Data Mining*, pages 18–25, 2000.
20. Philip Hans Franses. *Time Series Models for Business and Economic Forecasting*. Cambridge University Press, Cambridge, United Kingdom, 1998.
21. Andreas Galka. *Topics in Nonlinear Time Series Analysis with Implications for EEG Analysis*. World Scientific, Singapore, 2000.
22. Martin Gavrilov, Dragomir Anguelov, Piotr Indyk, and Rajeev Motwani. Mining the stock market: Which measure is best. In *Proceedings of the Sixth ACM International Conference on Knowledge Discovery and Data Mining*, pages 487–496, 2000.
23. Amir B. Geva. Hierarchical-fuzzy clustering of temporal patterns and its application for time-series prediction. *Pattern Recognition Letters*, 20(14):1519–1532, 1999.
24. Dina Q. Goldin and Paris C. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In *Proceedings of the First International Conference on Principles and Practice of Constraint Programming*, pages 137–153, 1995.
25. Amara L. Graps. An introduction to wavelets. *IEEE Computational Science and Engineering*, 2(2):50–61, 1995.

26. Ulf Grenander. *Elements of Pattern Theory.* Johns Hopkins University Press, Baltimore, MD, 1996.
27. Valery Guralnik and Jaideep Srivastava. Event detection from time series data. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 33–42, 1999.
28. Valery Guralnik, Duminda Wijesekera, and Jaideep Srivastava. Pattern-directed mining of sequence data. In *Proceedings of the Fourth ACM International Conference on Knoweldge Discovery and Data Mining*, pages 51–57, 1998.
29. Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, Cambridge, United Kingdom, 1997.
30. Jiawei Han, Wan Gong, and Yiwen Yin. Mining segment-wise periodic patterns in time-related databases. In *Proceedings of the Fourth ACM International Conference on Knowledge Discovery and Data Mining*, pages 214–218, 1998.
31. John Haslett and Adrian E. Raftery. Space-time modeling with long-memory dependence: Assessing Ireland's wind power resource. *Applied Statistics*, 38:1–50, 1989.
32. Yun-Wu Huang and Philip S. Yu. Adaptive query processing for time-series data. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 282–286, 1999.
33. Keita Ikeda, Bradley V. Vaughn, and Stephen R. Quint. Wavelet decomposition of heart period data. In *Proceedings of the IEEE First Joint BMES/EMBS Conference*, pages 3–11, 1999.
34. Henrik André-Jönsson and Dushan Z. Badal. Using signature files for querying time-series data. In *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 211–220, 1997.
35. Eamonn J. Keogh. Fast similarity search in the presence of longitudinal scaling in time series databases. In *Proceedings of the Ninth IEEE International Conference on Tools with Artificial Intelligence*, pages 578–584, 1997.
36. 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.
37. Eamonn J. Keogh and Michael J. Pazzani. Scaling up dynamic time warping for data mining applications. In *Proceedings of the Sixth ACM International Conference on Knowledge Discovery and Data Mining*, pages 285–289, 2000.
38. 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*, 2001.
39. Eamonn J. Keogh, Kaushik Chakrabarti, Sharad Mehrotra, and Michael J. Pazzani. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proceedings of the ACM SIGMOD Conference*, 2001.
40. Sang-Wook Kim, Sanghyun Park, and Wesley W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *Proceedings of the Seventeenth International Conference on Data Engineering*, pages 607–614, 2001.
41. Sze Kin Lam and Man Hon Wong. A fast projection algorithm for sequence data searching. *Data and Knowledge Engineering*, 28(3):321–339, 1998.
42. Mark Last, Yaron Klein, and Abraham Kandel. Knowledge discovery in time series databases. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 31(1):160–169, 2001.

43. Seok-Lyonh Lee, Seok-Ju Chun, Deok-Hwan Kim, Ju-Hong Lee, and Chin-Wan Chung. Similarity search for multidimensional data sequences. In *Proceedings of the Sixteenth International Conference on Data Engineering*, pages 599–608, 2000.
44. 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.
45. Ling Lin and Tore Risch. Querying continuous time sequences. In *Proceedings of the Twenty-Fourth International Conference on Very Large Data Bases*, pages 170–181, 1998.
46. Hongjun Lu, Jiawei Han, and Ling Feng. Stock movement prediction and N-dimensional inter-transaction association rules. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 12:1–7, 1998.
47. Konrad Maurer and Thomas Dierks. *Atlas of Brain Mapping: Topographic Mapping of EEG and Evoked Potentials*. Springer-Verlag, Berlin, Germany, 1991.
48. Ernst Niedermeyer and Fernando Lopes DaSilva. *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*. Lippincott, Williams and Wilkins, Baltimore, MD, fourth edition, 1999.
49. Sanghyun Park, Dongwon Lee, and Wesley W. Chu. Fast retrieval of similar subsequences in long sequence databases. In *Proceedings of the Third IEEE Knowledge and Data Engineering Exchange Workshop*, 1999.
50. Sanghyun Park, Wesley W. Chu, Jeehee Yoon, and Chihcheng Hsu. Efficient searches for similar subsequences of different lengths in sequence databases. In *Proceedings of the Sixteenth International Conference on Data Engineering*, pages 23–32, 2000.
51. 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.
52. 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.
53. Shai Policker and Amir B. Geva. Non-stationary time series analysis by temporal clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 30(2):339–343, 2000.
54. Kevin B. Pratt. Locating patterns in discrete time series. Master's thesis, Computer Science and Engineering, University of South Florida, 2001.
55. Yunyao Qu, Changzhou Wang, and Xiaoyang Sean Wang. Supporting fast search in time series for movement patterns in multiple scales. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, pages 251–258, 1998.
56. Prasanna K. Sahoo, S. Soltani, A. K. C. Wong, and Y. C. Chen. A survey of thresholding techniques. *Computer Vision, Graphics and Image Processing*, 41:233–260, 1988.
57. Hanan Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990.
58. Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of the Twenty-Fourth International Conference on Very Large Data Bases*, pages 428–439, 1998.
59. Sameer Singh and Paul McAtackney. Dynamic time-series forecasting using local ap-

proximation. In *Proceedings of the Tenth IEEE International Conference on Tools with Artificial Intelligence*, pages 392–399, 1998.

60. David S. Stoffer. Detecting common signals in multiple time series using the spectral envelope. *Journal of the American Statistical Association*, 94:1341–1356, 1999.
61. Robert A. Yaffee and Monnie McGee. *Introduction to Time Series Analysis and Forecasting*. Academic Press, San Diego, CA, 2000.
62. Byoung-Kee Yi, Nikolaos D. Sidiropoulos, Theodore Johnson, H. V. Jagadish, Christos Faloutsos, and Alexadros Biliris. Online data mining for co-evolving time series. In *Proceedings of the Sixteenth International Conference on Data Engineering*, pages 13–22, 2000.

Kevin B. Pratt received his B.A. in Mathematics (1970) and Juris Doctorate (1975) from the University of Texas at Austin, and M.S. in Computer Science (2001) from the University of South Florida. He is now a software engineer at Harris Corporation. His interests include artificial intelligence, image processing, and data mining. He conducted the reported research during his graduate studies at the University of South Florida.

Eugene Fink received his B.S. degree from Mount Allison University (Canada) in 1991, M.S. from the University of Waterloo (Canada) in 1992, and Ph.D. from Carnegie Mellon University in 1999. He is currently an assistant professor in the Computer Science and Engineering Department at the University of South Florida. His research interests are in various aspects of artificial intelligence, including machine learning, planning, and theoretical foundations of AI. His interests also include e-commerce and computational geometry.