LOCATING PATTERNS IN DISCRETE TIME-SERIES

by

KEVIN B. PRATT

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science
Department of Computer Science and Engineering
College of Engineering
University of South Florida

May 2001

Major Professor: Eugene Fink, Ph.D.

©Copyright by Kevin B. Pratt 2001
All rights reserved

DEDICATION

To Merl and William T. Ward, Lance Armstrong, and Terrence W. Pratt,
for their inspiration.

ACKNOWLEDGMENT

This thesis has greatly benefited from the thoughtful comments, insightful observations, and careful editorial suggestions of Eugene Fink. I am very grateful to him.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LOCATING PATTERNS IN DISCRETE TIME-SERIES

by

KEVIN B. PRATT

An Abstract

of a thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science
Department of Computer Science and Engineering
College of Engineering
University of South Florida

May 2001

Major Professor: Eugene Fink, Ph.D.

We describe a technique for fast compression of time-series, indexing of the resulting compressed series, and retrieval of series similar to a given pattern.

The compression algorithm identifies "important" points of a time-series and discards the other points. It runs in linear time, takes constant memory, and gives good results for a wide variety of time-series.

We use the important points not only for compression, but also for indexing a database of time-series, which supports efficient search for patterns and allows the user to control the trade-off between the speed and accuracy of search. The experiments show the effectiveness of the developed technique for identifying patterns in stock prices, meteorological data, and electrocardiograms.

Abstract Approved:____
Major Professor: Eugene Fink, Ph.D.
Assistant Professor,
Department of Computer Science and Engineering

Date Approved:________________________________

CHAPTER 1 – INTRODUCTION

The purpose of the described work is to develop a technique for fast search for a given pattern in a time-series. For example, suppose we believe that the pattern in the stock prices in Figure 1.1 helps predict future prices, or that the pattern in the electrocardiogram in Figure 1.2 indicates a disease. If these patterns are useful, we may need to search for similar patterns. In Figures 1.1 and 1.2, we show instances of similar patterns located by the developed technique.

The analysis of time-series, which includes compressing, indexing and searching time-series, is an active research area. We may use the similarity among time-series for the following purposes:

- classification and taxonomy,
- query and retrieval,
- grouping similar series together (clustering), and
- identification of unusual series or intervals within a series.

A similarity measure needs to be accurate and allow efficient implementation. A related problem is to develop a method for compressing time-series that preserves similarity between series.

Efficient search for patterns in time-series may find applications in many domains. For example, a pattern in stock prices may correlate with economic events. As another example, a pattern in an electrocardiogram may precede onset of a disease.

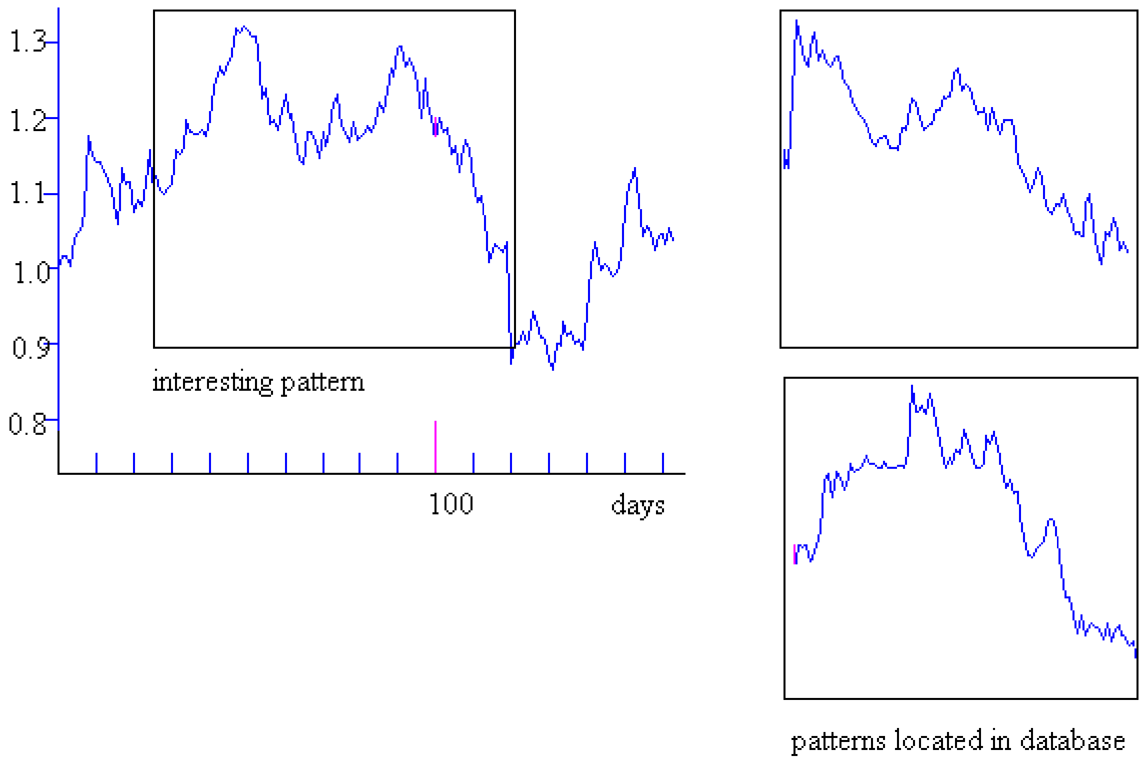


Figure 1.1: Example of a pattern in a stock chart and similar patterns located in a stock database.

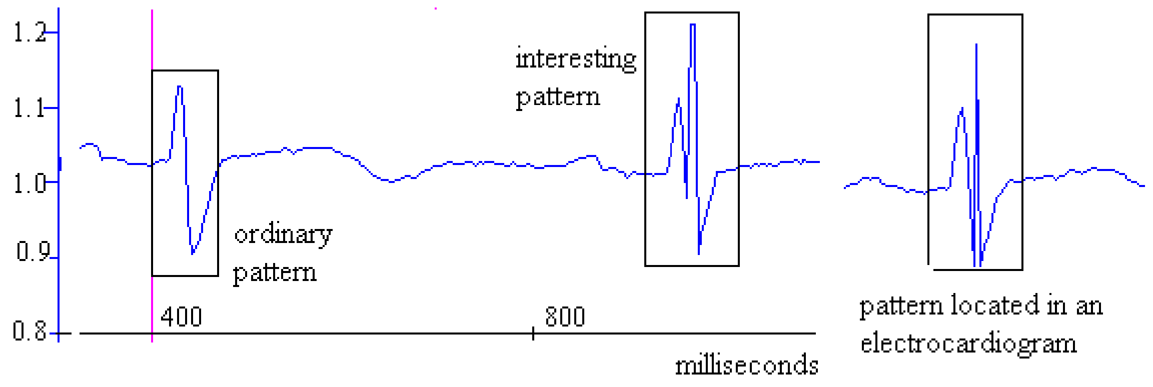


Figure 1.2: Example of two patterns in an electrocardiogram and a located similar pattern.

1.1 Contributions

The main contributions of the presented work include a technique for compressing time-series and an algorithm for fast retrieval of time-series that closely match a given pattern.

Formally, a *time-series* is a sequence of values, measured at equal time intervals; we assume that all values in the series are *positive*. For example, the lower time-series in Figure 1.3 has the values 20, 22, 25, 22, 25, 27, 27, and so on. We may use a subsequence as a *compressed representation* of a full sequence. For example, we can select every tenth value, or all local maxima and minima. When using maxima and minima for compression, we often call them "important points" of the sequence. We circled these important points in the lower curve in Figure 1.3.

We give a compression technique based on extraction of certain important points from a time-series, which works in linear time and takes constant memory. It requires one pass through the time-series, with no pre-processing. The technique gives good results for a variety of time-series, including erratic time-series, such as shown in Figure 1.1.

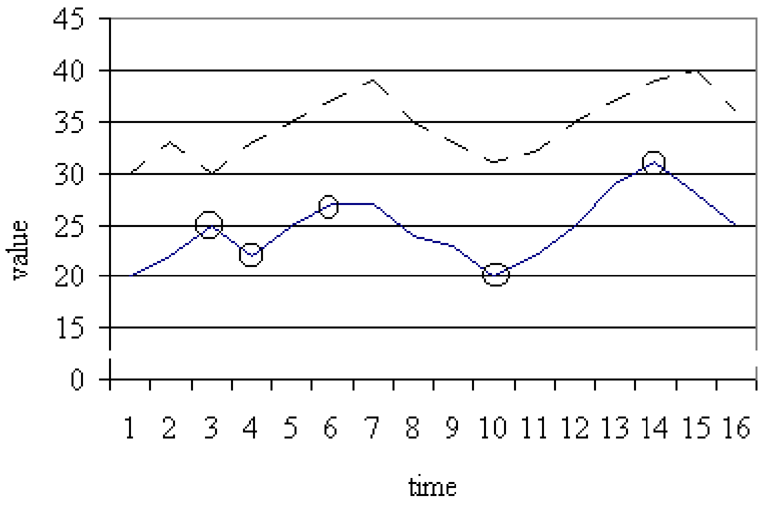


Figure 1.3: Example of two time-series which are similar, but not identical.

Then, we propose a metric for measuring the similarity of two time-series, compare it with alternative metrics, and show that it works well with compressed data. We measure similarity on a scale from zero to one, where zero means no likeness and one means perfectly alike.

Finally, we present a technique for indexing time-series, and show its utility for fast retrieval of similar patterns, using the described similarity metric.

Historically, most time-series research has been done with data that is already fully collected, such as last year's stock prices; however, we often have to analyze data in the process of collection. For example, we may need to process a continuous electrocardiogram of a patient in an emergency room. We refer to it as *streaming* data. The developed techniques work for streaming data, as well as for traditional "static" data.

1.2 Data sets

We tested the developed techniques on large time-series from different domains, summarized in Table 1.1. All these data are publicly available.

- Standard and Poor's 100 stock prices

We used stocks from the Standard and Poor's 100 listing of large companies, as well as the Dow Jones Industrials, for the period from January 1, 1998 to April 20, 2000. The ending date was the last available date when we downloaded the data; it had no special significance. We downloaded daily split-corrected prices from America Online and repaired missing values by duplicating the prior-day prices. We discarded newly listed and de-listed stocks, and used ninety-eight stocks in experiments.

- Air and sea temperatures

We used daily temperature readings from sixty-eight buoys in the Pacific Ocean from1980 to 1998, downloaded from the Knowledge Discovery and Data Mining database at the University of California at Irvine, at

Table 1.1: Test data sets.

Data set	Number of series	Description	Total number of points	Measure-ment intervals
Stock prices	98	98 stocks, 2.3 years, 252 values per year for each stock	60,000	1 day
Air and sea temperatures	68	68 buoys, 2 sensors per buoy, 18 years, 365 values per year for each sensor	445,000	1 day
Wind speeds	12	12 stations, 18 years, 365 values per year for each station	79,000	1 day
Electroen-cephalogram	61	61 electrodes, 256 values per electrode	17,000	0.004 second
Electro-cardiogram	1	1 electrode, 2200 values	2,200	0.006 second

`kdd.ics.uci.edu/databases`. These buoys do not have fixed locations, and often drift. To repair the data, we developed a routine to position the buoys on a map and interpolate missing values.

Air and sea temperatures are related. For example, we may see two patterns in the air temperature at zero latitude in Figure 1.4. The first pattern is very similar to the pattern in the sea temperature at the same location. It is roughly similar to the air and sea temperatures two degrees further south. The second pattern in the air temperature at zero latitude is close to the sea temperature at the same location. It is absent two degrees further south.

♦ Wind speed

We used daily wind speeds from twelve sites in the Republic of Ireland from 1961 to 1978, obtained from the Statistical Database at Carnegie Mellon University, at `www.stat.cmu.edu/datasets/wind.desc`.

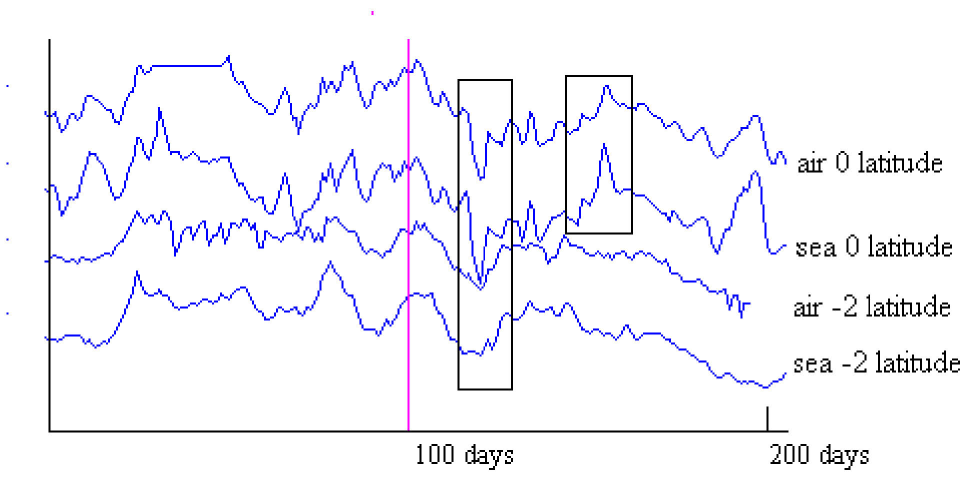


Figure 1.4: Example of patterns in air and sea temperatures.

The first pattern appears in all time-series, whereas the second is only in the first two.

- Electroencephalograms

These data show electrical changes in the scalp, and help to analyze brain activity. In Figure 1.5, we illustrate electroencephalograms for a scalp location designated "CZ," and four immediately surrounding locations. We used electroencephalograms of a human subject obtained by Henri Begleiter at the Neurodynamics Laboratory of the State University of New York Health Center at Brooklyn. These data are from sixty-four electrode sensors located at standard sites on the scalp, and were downloaded from the Knowledge Discovery and Data Mining database at the University of California at Irvine, at `kdd.ics.uci.edu/databases.`

- Electrocardiograms

Electrocardiograms track electrical activity of the heart. The healthy activity has several standard patterns, and deviations from them may indicate pathology (see Figure 1.2). We applied the developed technique to search for abnormal patterns in an electrocardiogram, downloaded from the University of Washington at `www.ms.washington.edu/~s530/data.html`.

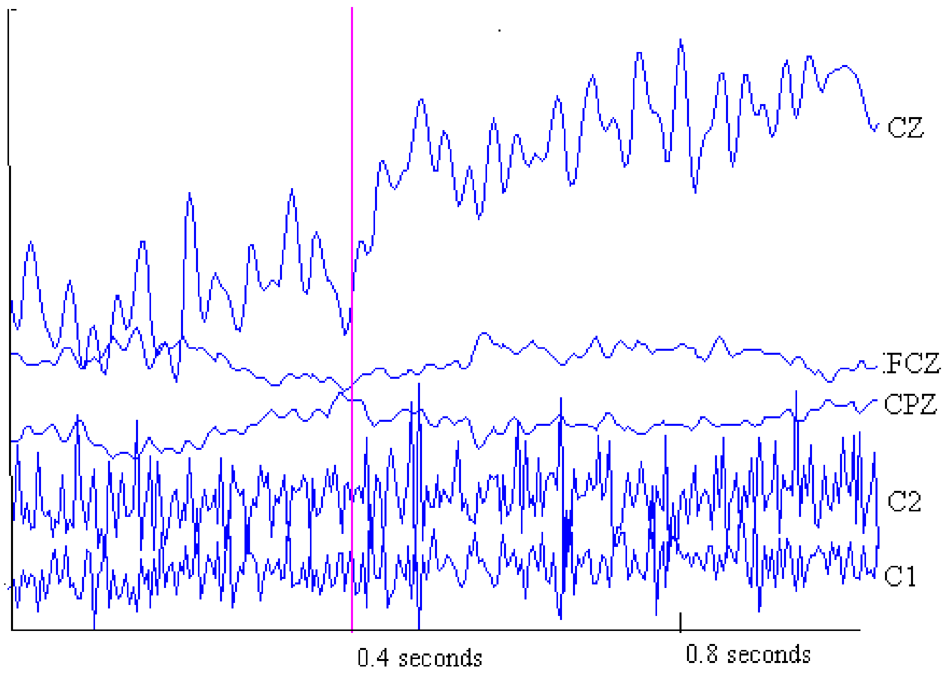


Figure 1.5: **Electroencephalogram data at five electrodes.**

CHAPTER 2 - PREVIOUS WORK

We now review previous work on the comparison of time-series and search for patterns. Note that this work does not directly relate to the prediction of future values, which has been another active direction in time-series analysis [Franses, 1998; Spiegel, 1996; Plane, 1996].

2.1 Feature sets

Researchers have investigated the use of various feature sets for compressing time-series and measuring similarity between series.

In particular, they have extensively studied discrete Fourier Transforms, which convert a series into a set of coefficients [Singh, 1998; Sheikholeslami, 1998; Stoffer, 1999; Yi, 2000]. These transforms allow fast, accurate compression of a time-series; however, they have several disadvantages. In particular, the transforms smooth local extrema, which may lead to a loss of important information in some domains, such as stock charting. Also they do not work well for erratic time-series [Ikeda, 1999]. Finally, there is no way to select a segment as a pattern [Han, 1998], without reconstituting the original series and obtaining new coefficients for the segment.

Recently, researchers have studied the use of small, descriptive alphabets for compressing time-series. For example, Guralnik [1997] compressed stock prices using a nine-letter alphabet to describe three features, each of which had three values. Sing [1998] represented stock prices, particle dynamics, and stellar light intensity with small words defined over a three-letter alphabet. Chi [1995] used an alphabet of simple DNA molecule combinations for genome sequences. Lin [1998] used a two-letter alphabet to encode major spikes in a series. The prime advantage of this technique is high compression rate; however, its descriptive power is limited, which makes it unusable in many domains.

Han [1998] used small categories of discretized values, similar to alphabets. The categories are fully ordered, whereas letters in an alphabet may not be ordered. While offering substantial compression, the discretized values suffer the traditional problem that values near a category boundary can be misclassified.

Das [1998] studied another variety of a limited alphabet, based on primitive shapes, and used it to develop efficient compression algorithms. He has not developed a universal set of primitive shapes, and the technique requires the user to hand-code appropriate basic shapes for each domain.

Several researchers used statistics for multiple intervals of a series to summarize the properties of a time-series; however, this technique gives poor results for erratic time-series [Policker, 2000; Geva, 1999; Stoffer 1999; Popivanov, 1998]. These statistics usually require equal length intervals, and do not allow comparison of patterns of different length.

Perng [2000] investigated a compression technique based on extracting "landmark points" from a series, and discarding other points; his choice of landmark points included local maxima and minima of the series. Keogh [1997; 1998] used the end-points of best-fit line segments to compress the series. In Chapter 3, we offer an alternative compression technique, based on selecting local maxima and minima, and show that it is more accurate than other compressed representations. We give a linear-time algorithm for finding important points, which is more efficient than Perng's iterative algorithm.

2.2 Similarity measures

The choice of feature sets affects techniques for measuring similarity of time-series. Researchers have studied a number of similarity measures, which include the computation of similarity through weighted feature differences, use of qualitative categories, and various clustering techniques.

♦ Euclidean distance

Some researchers defined similarity as the distance between vectors in an *n*-dimensional feature space. For example, Caraca-Valente [2000] used Euclidean distance to compute similarity of the feature vectors containing angle of knee movement and muscle strength. Lee [2000] applied Euclidean distance to compare feature vectors containing color, texture, and shape of sequential video pictures. This metric works well when different features have the same units and scale [Goldin, 1995]; however, it causes errors when combining disparate features, such as time and dollars [Gunopulos, 2000].

♦ Bounding rectangles

An alternative definition of similarity is based on the notion of *bounding rectangles*, illustrated in Figure 2.1. Two series are similar if their bounding rectangles are similar. The use of bounding rectangles allows fast pruning of clearly dissimilar curves [Perng, 2000; Lee, 2000]; however, it is less effective for selecting the most similar curve among close candidates. This technique requires intelligent selection of segments for bounding rectangles, which usually involves human assistance.

♦ Envelope count

We may divide the time axis into short segments, called *envelopes*, and define a yes/no similarity for each envelope. Specifically, 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 largest number of consecutive envelopes where the series are similar [Agrawal, 1996]. This measure allows fast computation of similarity; furthermore, we can readily adapt it for handling noisy and missing data [Das, 1997; Bollobas, 1997].

♦ Aggregate similarity

We can measure point-by-point similarities of two series, and then aggregate these measures. This technique often involves interpolation to obtain values for missing points. For example, Keogh [1997; 1998] used linear interpolation with this technique,

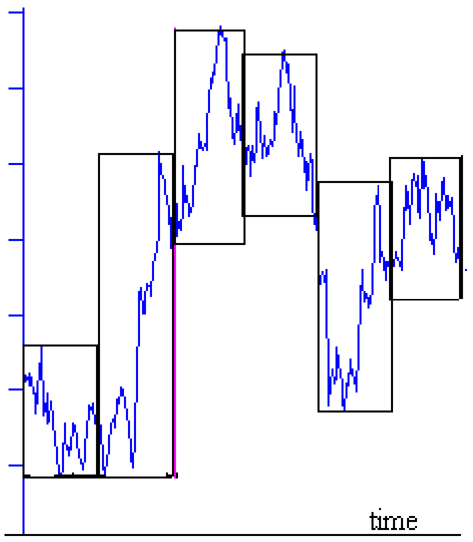


Figure 2.1: Example of bounding rectangles.
A bounding rectangle of series on a given interval is the minimal rectangle that includes all points of the series. We may consider two curves similar if they have identical bounding rectangles.

and Perng [2000] applied cubic approximation. Keogh [2000] also described the use of point-by-point similarity with modified Euclidean distance, which does not require interpolation.

We use a similar approach in the reported work; specifically, we define similarity between individual points of compressed time-series and use a weighted aggregation of these similarities.

2.3 Indexing and retrieval

Researchers have studied a variety of techniques for indexing and retrieval of time-series. They utilized several advanced techniques from algorithm theory, including tree structures and grids.

In particular, they used B-tree indexing, which is an extension of red-black trees where a node may have more than two children; for example, see the textbook by Cormen *et al.*[1990]. They have also used R-trees, which extend B-trees for indexing points in multi-dimensional space [Kamel, 1993]. Although the traditional use of R-trees was indexing spatial data, this structure also allows indexing of time-series by their position in feature space [Gunopulos, 2000].

The *kd*-tree technique is an extension of binary search trees, which uses different features of an object at different levels of the tree, and allows the both numeric and qualitative features [Gunopulos, 2000]. Deng [1998] applied this structure to index sequences by their significant features.

Bozkaya [1997; 1999] used vantage-point trees for indexing time-series by their numerical features. Aggarwal [2000] considered the use of grid structures for a similar problem, but found that generally their performance in high-dimensional space is no better than exhaustive linear search.

Gunopulos [2000] and Aggarwal [2000] reviewed the use of compression with linear-search retrieval, and concluded that exhaustive search in the database of compressed sequences is often faster than sophisticated indexing techniques.

We also use compression for efficient retrieval, and combine it with a simple indexing technique and heuristics for identifying "prominent" features of a time-series. In Chapter 5, we describe this approach, which allows fast retrieval of similar time-series and enables the user to control the trade-off between speed and accuracy of retrieval.

The developed technique meets most of the criteria suggested by Gunopulos [2000], who pointed out that a retrieval algorithm should:

- work for erratic time-series,
- accept any prototype pattern,
- find inexact matches,
- evaluate the accuracy of matches,
- work when some points are missing, and
- work on streaming data.

CHAPTER 3 - IMPORTANT POINTS

We compress a time-series by selecting some of its local maxima and minima, called *important points*, and dropping the other points (see Figure 3.1). We can control the number of selected points, which determines the compression rate. This compression technique is *lossy*, that is, we cannot restore the original series from the compressed version. In other words, the compressed curve is an approximation of the initial curve.

3.1 Choice of important points

The intuitive idea is to discard minor fluctuations in a series, and keep major maxima and minima. We control the compression rate with a knob parameter, called R, which is always greater than one. Increasing R leads to selecting fewer points.

A point a_m is an *important minimum* if there are indices i and j, where $i \leq m \leq j$, such that

- ♦ a_m is the minimum among $a_i, \ldots, a_j$, and
- ♦ $a_i/a_m \leq R$ and $a_j/a_m \leq R$.

Intuitively, a_m is an important minimum if it is the minimal value of some segment $a_i, \ldots, a_m, \ldots, a_j$ of the series, and the end-point values of this segment are much larger than a_m. For example, the point a_m in Figure 3.2(a) is an important minimum, since it is the minimum of the segment $a_i, \ldots, a_j$, and the end-point values of this segment are greater than $a_m \bullet R$. On the other hand, the local minimum a_k is *not* an important point.

The definition of an important maximum is symmetric. That is, a point a_m is an *important maximum* if there are indices i and j, where $i \leq m \leq j$, such that

- ♦ a_m is the maximum among $a_i, \ldots, a_j$, and
- ♦ $a_m/a_i \leq R$ and $a_m/a_j \leq R$.

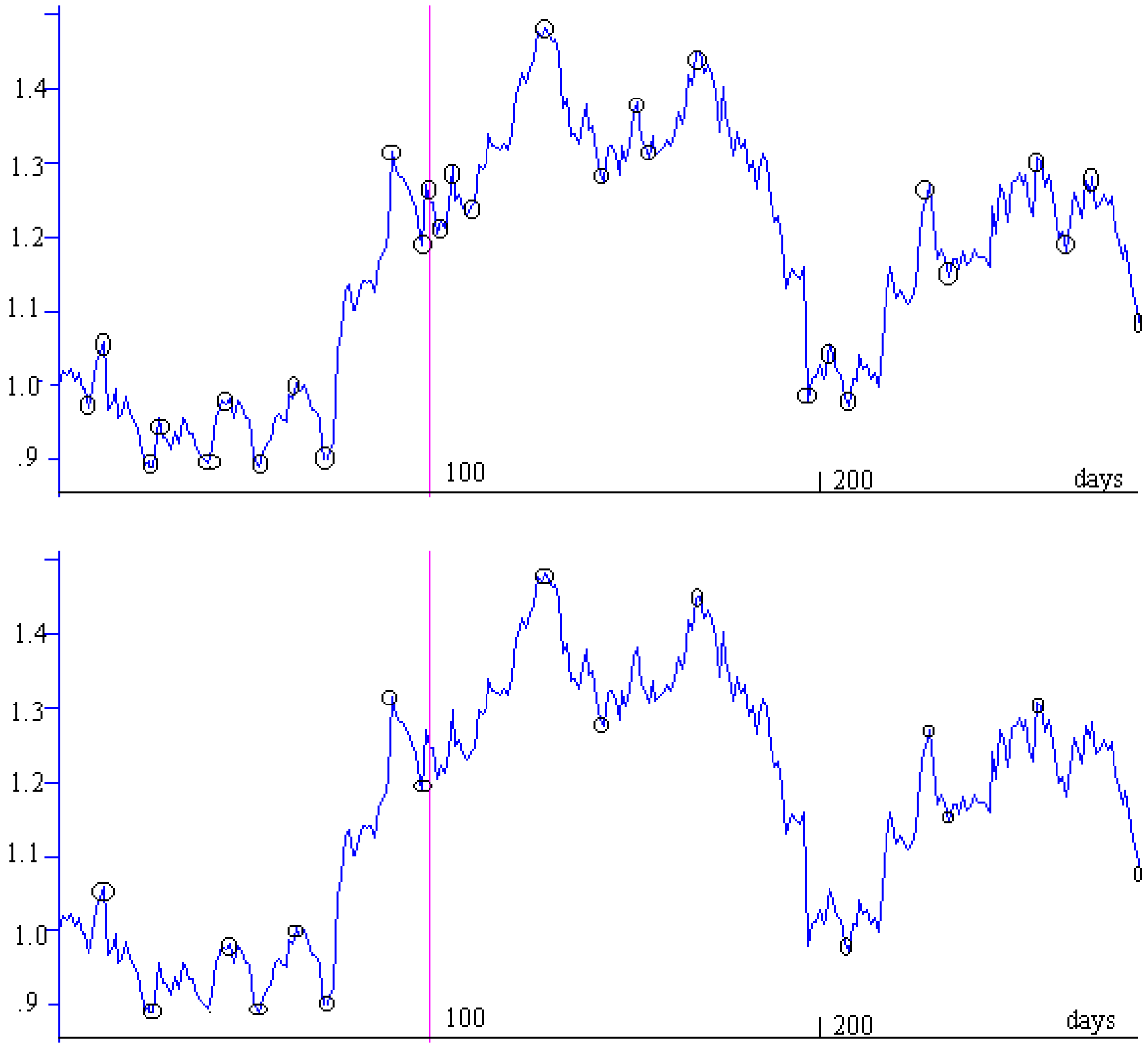
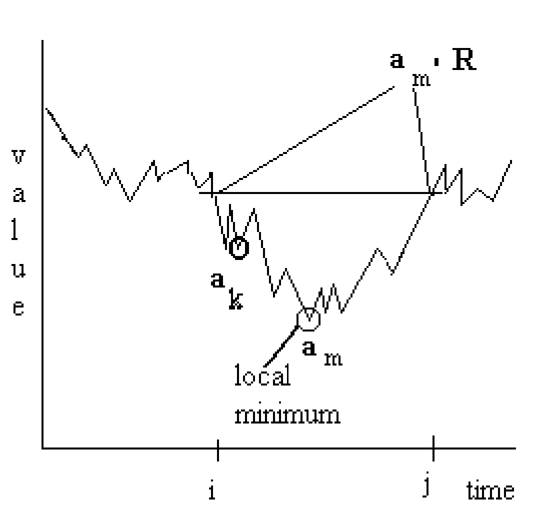


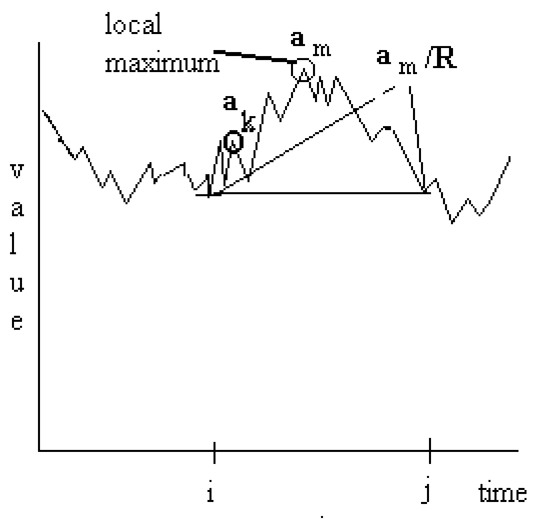
Figure 3.1: Important points in stock prices.

We circle important points for 10% compression (top) and 5% compression (bottom).

For example, a_m in Figure 3.2(b) is an important maximum, whereas a_k is *not* an important maximum.

In Figure 3.3, we give an algorithm for selecting important points, which performs one pass through the series and outputs the values and indices of the selected points. First, we apply FIND-FIRST-TWO and then alternately invoke FIND-MINIMUM and





(a) (b)

Figure 3.2: Examples of important minimum (left) and important maximum (right).

FIND-MAXIMUM. We can easily adapt the algorithm to process streaming data by replacing "**if** $i \le n$" with "**if** stream not terminated."

The algorithm performs one pass through the series; its time complexity is linear, $\theta(n)$, and it takes constant memory. We have implemented it in Visual Basic 6.0 and tested on a 300 MHz PC. For an *n*-point sequence, the processing time is about $0.014 \bullet n$ milliseconds. The algorithm works well with erratic series, such as the series in Figure 3.1, where traditional thresholding [Sahoo, 1988] does not find local minima among the higher values in the center of the series, nor local maxima among the lower values on the left and right.

```
IMPORTANT-POINTS
♦ Top-level function for finding 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($i$)
        if $i < n$ then $i$ = FIND-MINIMUM($i$)

FIND-FIRST-TWO
♦ Find the first and second important points.
$iMax = 1$;  $iMin = 1$
while $i \le n$ and $a_{iMax}/a_i < R$ and $a_i/a_{iMin} < R$ do
        if $a_i > a_{iMax}$ then $iMax = i$
        if $a_i < a_{iMin}$ then $iMin = i$
        $i = i + 1$
if $i < n$ and ($a_{iMax}/a_i < R$ or $a_i/a_{iMin} < R$) then
        if $iMax < iMin$ then output($a_{iMax}$, $iMax$); output($a_{iMin}$, $iMin$)
        else output($a_{iMin}$, $iMin$); output($a_{iMax}$, $iMax$)
return $i$

FIND-MINIMUM($i$)
♦ Find the first important minimum after the $i$th element.
$iMin = i$
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 $i$th element.
$iMax = i$
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 3.3: **Compression algorithm.**

We process a global series $a_1,\ldots, a_n$, and use a global variable n that denotes the series' size. The algorithm outputs the values and indices of the selected important points.

3.2 Compression accuracy

We applied the compression algorithm to the data sets from Chapter 1, and compared it with two simpler techniques, specifically, equally spaced points and randomly selected points. For each of these techniques, we used the compressed data to interpolate the missing points, and measured the difference between the original sequence and the approximated sequence. We used three difference measures, listed in Figure 3.4.

We summarize the results in Table 3.1 and Figures 3.5–3.7, which show that important points are significantly more accurate than the other two methods. For example, if we apply these techniques to stock prices, then 5% compression with important points is as accurate as 16% compression by the other two techniques.

(a) Mean difference:

$$\frac{\sum_{j=1}^{m} \sum_{i=1}^{n} | a_{ij} - b_{ij} |}{m \cdot n}$$

(b) Maximum difference:

$$\frac{\sum_{j=1}^{m} \max_{i \in [1..n]} | a_{ij} - b_{ij} |}{m}$$

(c) Root mean square difference:

$$\frac{\sum_{j=1}^{m} \sqrt{\frac{\sum_{i=1}^{n} (a_{ij} - b_{ij})^2}{n}}}{m}$$

Figure 3.4: Measures of difference between original and compressed data.

Series *a* is the original data and series *b* is interpolation from the compressed data. We test compression on *m* data series, *n* points each, and average the resulting differences.

Table 3.1: Accuracy of three compression techniques, at different compression levels.

	Mean difference			Maximum difference			Root mean square difference		
	Important points	Fixed points	Random points	Important points	Fixed points	Random points	Important points	Fixed points	Random points
Five-percent compression									
Stocks	0.05	0.10	0.12	1.30	1.80	1.80	0.11	0.32	0.30
Air temperature	0.029	0.085	0.082	0.74	0.83	0.83	0.12	0.23	0.21
Sea temperature	0.030	0.079	0.079	0.78	0.85	0.85	0.12	0.23	0.21
Wind speed	0.047	0.042	0.044	0.075	1.09	1.10	0.070	0.081	0.081
Encephalogram	0.13	0.17	0.16	0.90	1.10	1.10	0.24	0.31	0.28
Ten-percent compression									
Stocks	0.03	0.06	0.07	1.10	1.70	1.70	0.08	0.21	0.21
Air temperature	0.022	0.050	0.050	0.64	0.80	0.78	0.08	0.16	0.14
Sea temperature	0.014	0.043	0.046	0.60	0.83	0.82	0.07	0.16	0.14
Wind speed	0.034	0.036	0.038	0.055	1.09	1.03	0.050	0.062	0.062
Encephalogram	0.08	0.13	0.12	0.82	1.10	1.09	0.17	0.27	0.24
Twenty-percent compression									
Stocks	0.02	0.03	0.04	0.70	1.70	1.60	0.05	0.14	0.14
Air temperature	0.010	0.030	0.030	0.33	0.77	0.72	0.03	0.01	0.01
Sea temperature	0.008	0.025	0.025	0.35	0.81	0.75	0.03	0.10	0.10
Wind speed	0.022	0.027	0.031	0.040	1.09	1.01	0.035	0.048	0.052
Encephalogram	0.03	0.06	0.07	0.68	1.08	1.00	0.10	0.18	0.17

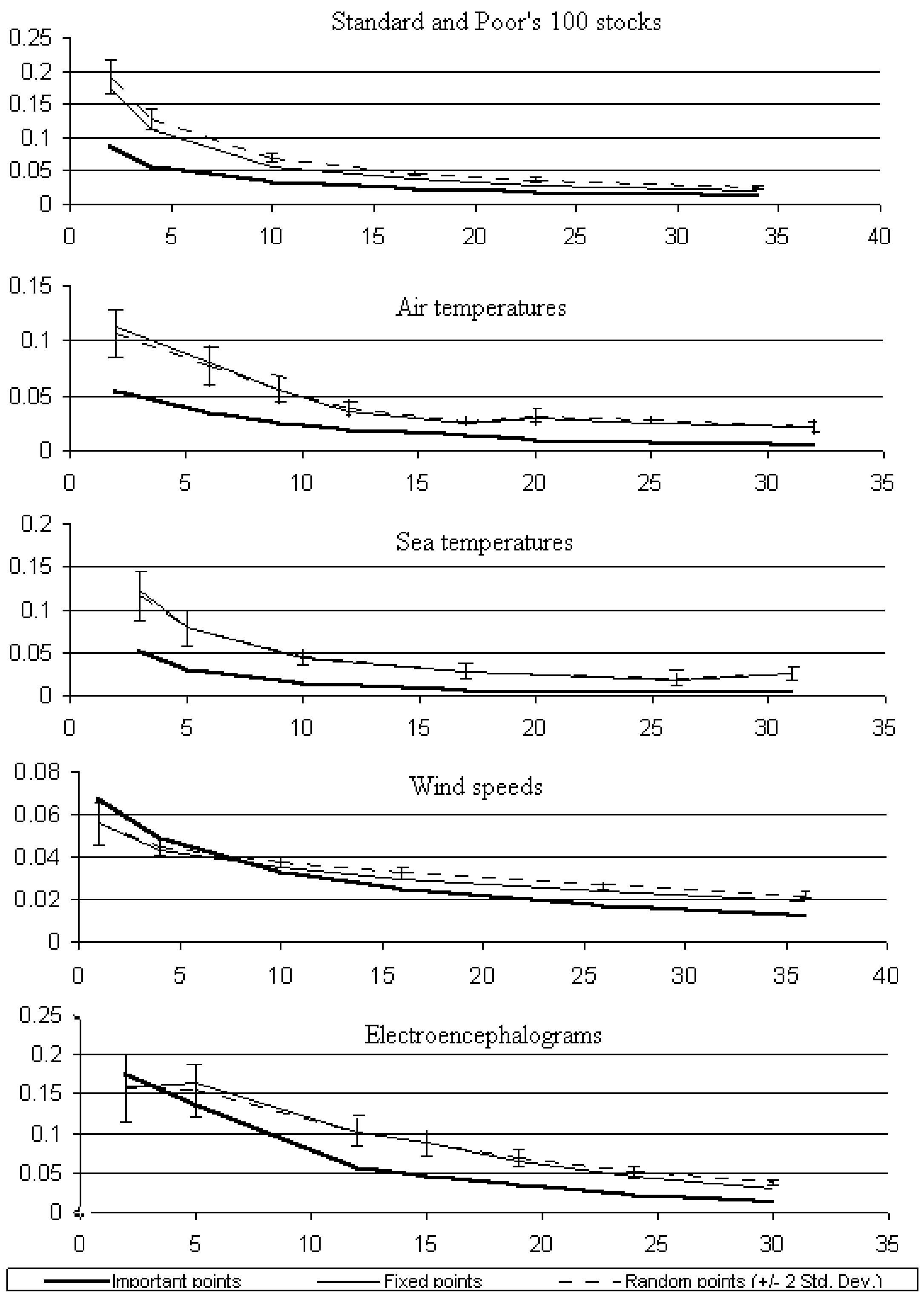


Figure 3.5: Accuracy of compression as determined by mean difference.

The horizontal axis shows the compression rate, whereas the vertical axis is the mean difference between the original series and compressed series.

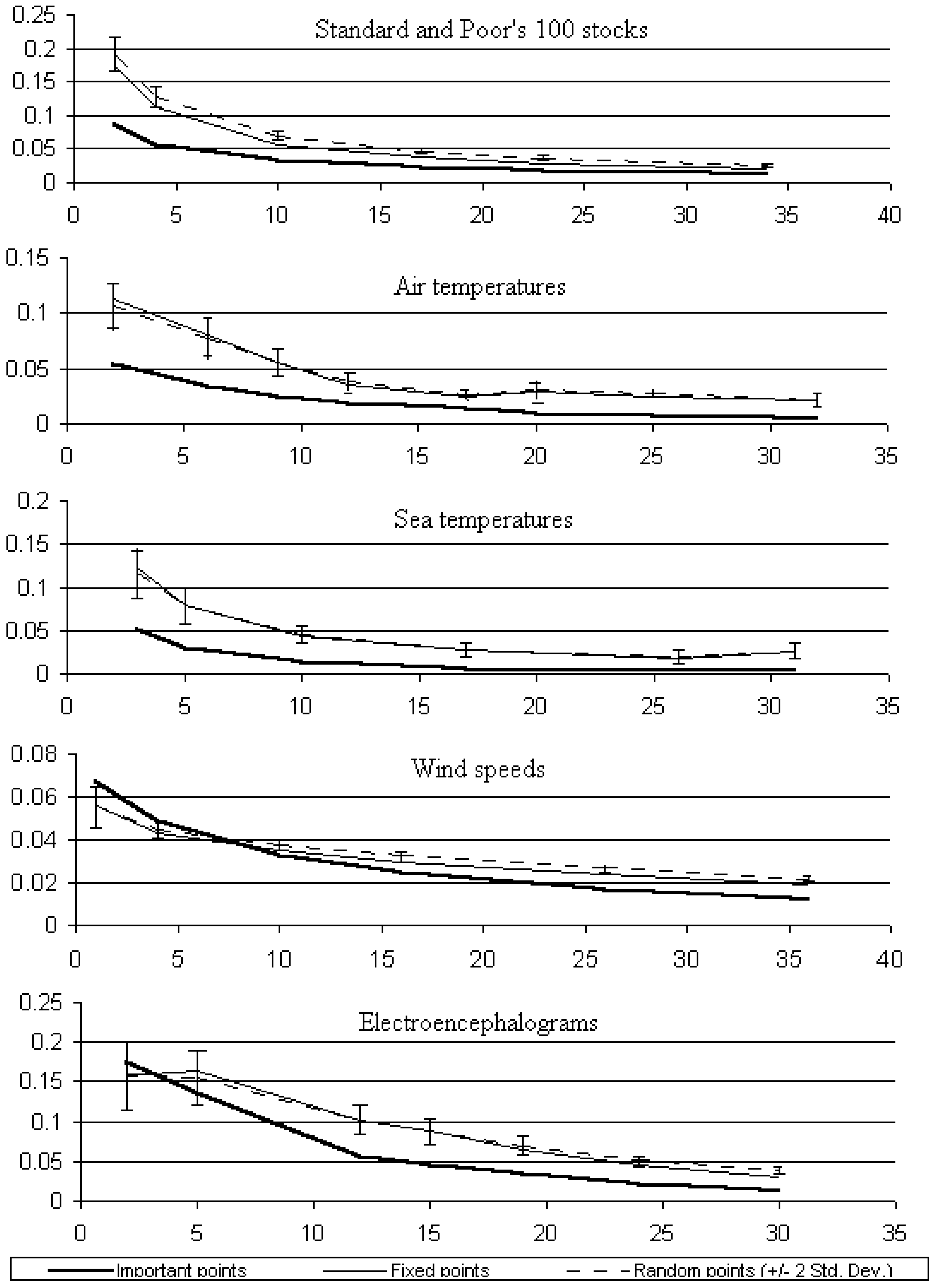


Figure 3.6: Accuracy of compression as determined by maximum difference.

The horizontal axis is the compression rate, and the vertical axis is the maximum difference between the original and compressed series.

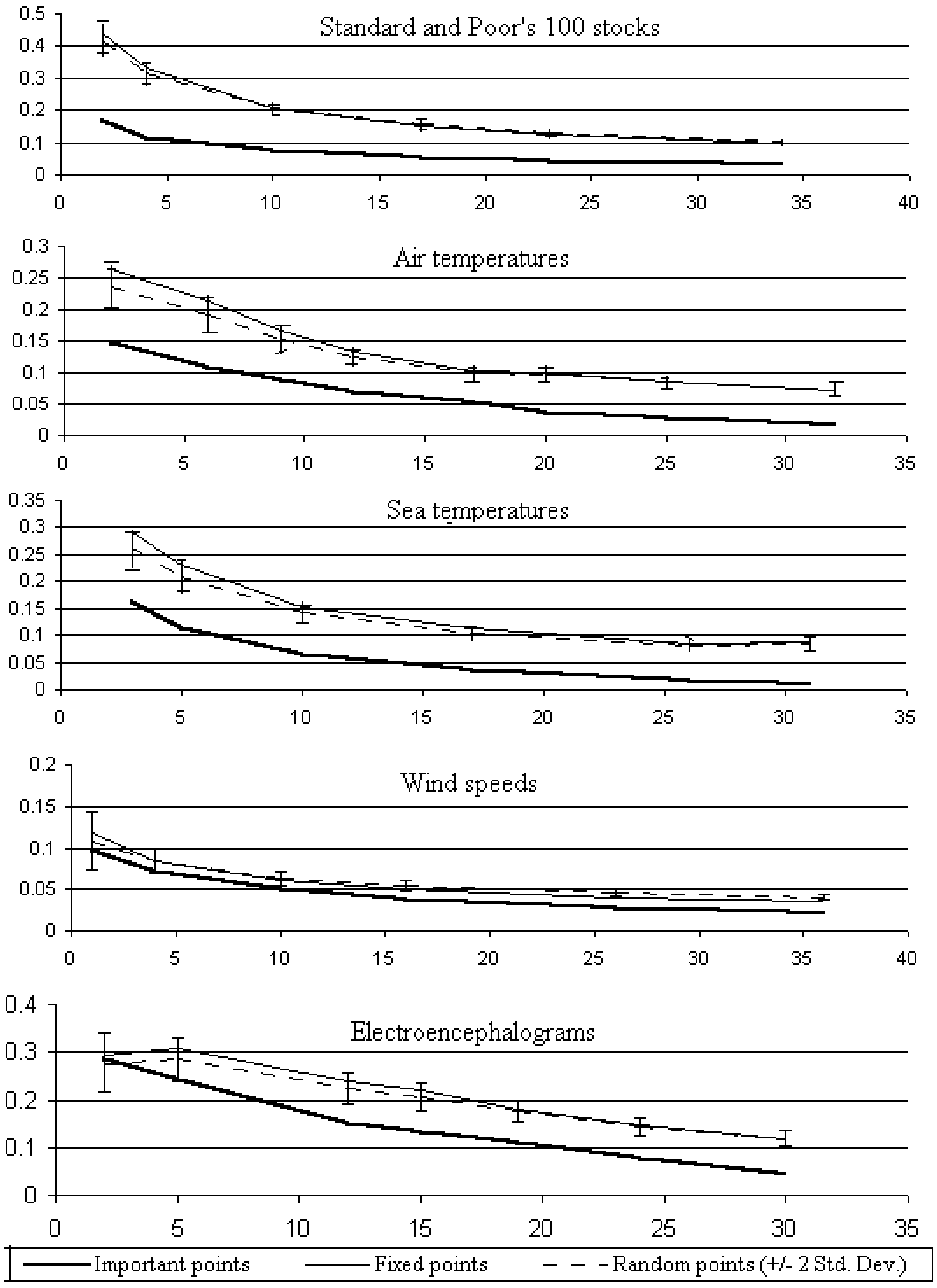


Figure 3.7: Accuracy of compression as determined by root mean square difference.

The horizontal axis is the compression rate, and the vertical axis is the root mean square difference between the original and compressed series.

CHAPTER 4 - MEASURING SIMILARITY

We consider four alternative measures of similarity between time-series, and then empirically evaluate their effectiveness. We illustrate application of these measures to the four time-series in Figure 4.1 and show the results in Table 4.1.

4.1 Standard similarity metrics

We measure similarity on a zero-to-one scale, where zero means no likeness and one means perfectly alike. Note that it differs from distance measures, which usually range from zero to infinity, with zero meaning perfect likeness. We use similarity rather than distance because a small similarity value in an outlier point does not skew the mean as much as a large distance.

Researchers have often measured similarity between time-series by aggregating point similarity. We review three aggregate metrics, which are based on mean, root mean square, and correlation coefficient, and then propose a new metric. We assume that all values of time-series are *positive*, and use this assumption in defining similarity measures. First, we define a similarity between two positive numeric values, *a* and *b*:

$$\mathrm{sim}(a, b) = 1 - 2 \bullet \frac{|a - b|}{a + b}$$

This definition is symmetric, that is, sim(a, b) = sim(b, a).

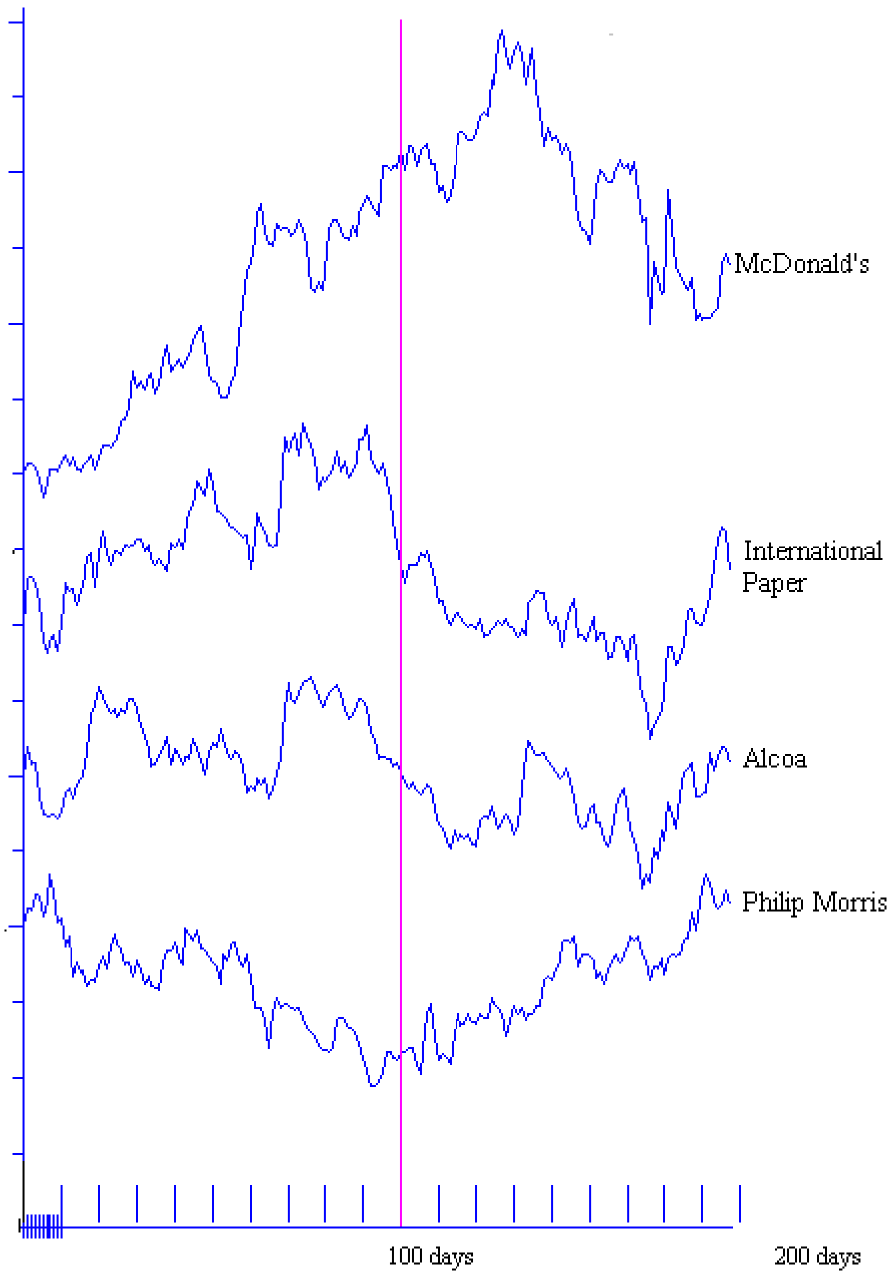


Figure 4.1: Example of similarity among stock charts.

We show four stock charts for the period from January 2, 1998 to September 30, 1998. We offset the curves vertically for easier visibility; all actually have the same initial value. Intuitively, we expect the International Paper and Alcoa curves are most similar.

Table 4.1: Comparison of similarity metrics.

We show the ranking of similarity for all pairs of stock charts in Figure 4.1. We rank the most similar pair as 1, and the least similar pair as 6.

	Mean similarity	Root mean square similarity	Correlation coefficient	Peak similarity
McDonald's – Int'l Paper	4	3	2	4
McDonald's – Alcoa	5	5	3	5
McDonald's – Philip Morris	6	6	5	6
Int'l Paper – Alcoa	1	1	1	1
Int'l Paper – Philip Morris	3	3	6	3
Alcoa – Philip Morris	2	2	4	2

The *mean similarity* between two series, $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$, is the mean of the point-by-point similarity:

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

Similarly, we may define the *root mean square similarity*:

$$\sqrt{\frac{\sum_{i=1}^{n} \mathrm{sim}(a_i, b_i)^2}{n}}.$$

We also consider correlation coefficient, which is a standard statistical method for measuring similarity of two sequences. It ranges between minus one and one, but we can readily convert it to the "traditional similarity range" by adding one and dividing by two. For two time-series, $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$, with mean values $m_a = (a_1 + \ldots + a_n)/n$ and $m_b = (b_1 + \ldots + b_n)/n$, the correlation coefficient is:

$$\frac{\sum_{i=1}^{n} (a_i - m_a) \bullet (b_i - m_b)}{\sqrt{\sum_{i=1}^{n} (a_i - m_a)^2 \bullet \sum_{i=1}^{n} (b_i - m_b)^2}}.$$

4.2 Peak similarity

We now define a new similarity metric, called *triangle similarity*, whose value also ranges from zero to one. The definition includes a positive knob parameter c, which allows us to adjust the resolution of this metric. The triangle similarity, between two positive numeric values, a and b, is as follows:

$$\text{tsim}(a, b) = \max(0, 1 - |a - b| / a \bullet c).$$

Note that the triangle similarity is not symmetric, that is, tsim(a, b) may be different from tsim(b, a). In Figure 4.2, we illustrate the intuitive meaning of this definition. We construct an equilateral triangle with the upper vertex (a, 1) and the other two vertices ($a \bullet (1 - c)$, 0) and ($a \bullet (1 + c)$, 0). To determine the similarity of a and b, we place b on the horizontal axis. If b is outside the triangle, the similarity is zero. On the other hand, if b is within the triangle, we draw a vertical line through b to obtain its intersection, b', with a side of the triangle. The ordinate of b' is the similarity between a and b.

We now define a *peak similarity* of two time-series, $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$, in terms of the triangle similarity of their points. The definition includes an additional knob parameter, w, which must be between zero and one:

$$w \bullet \max\left(\frac{\sum_{i=1}^{n} \text{tsim}(a_i, b_i)}{n}, \frac{\sum_{i=1}^{n} \text{tsim}(b_i, a_i)}{n}\right) + (1 - w) \bullet \max\left(\min_{i \in [1..n]} \text{tsim}(a_i, b_i), \min_{i \in [1..n]} \text{tsim}(b_i, a_i)\right)$$

Intuitively, the first part of the expression represents the mean triangle similarity of the points, whereas the second part is the smallest similarity of the points. The weight w

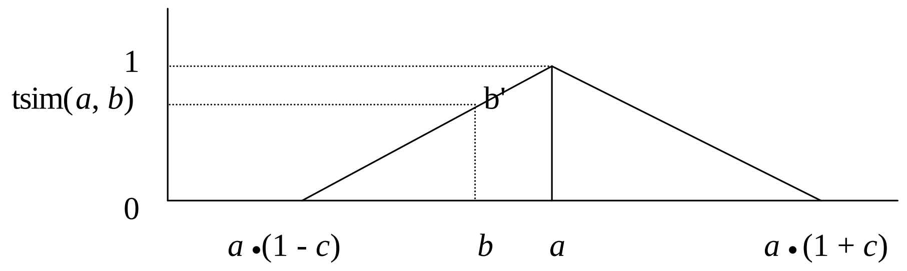


Figure 4.2: Triangle similarity of numeric values *a* and *b*.

determines the relative importance of these two parts in the aggregated similarity measure. The resulting similarity is symmetric; furthermore, it allows computing similarity between curves whose points do not exactly coincide.

The peak similarity is effective only when two sequences have the same starting value, that is, $a_1 = b_1$. If they do not satisfy this assumption, we re-scale the sequences before applying the metric; specifically, we divide all values in the first series by a_1, and all values in the second series by b_1.

The main advantage of peak similarity for this research is that it works well with compressed curves. The experiments show that it gives better results than other similarity metrics.

4.3 Empirical comparison

We next give empirical evaluation of the metrics described in sections 4.1 and 4.2. We applied these metrics to select similar series, and then measured the mean difference between similar series. For each given series, we found the five most similar series, and then determined the mean distance between the given series and the other five; we repeated this experiment for each similarity metric. When selecting similar series, we used compressed data, with two different compression rates, 5% and 10%.

In Table 4.2, we summarize the results, and compare them with the results of the perfect exhaustive-search selection, as well as with random selection. We conclude that the use of similarity with compressed data is much better than random selection, though it is not as good as exhaustive search. The results also show that the peak similarity performs somewhat better than other metrics, and that the correlation coefficient is the least effective. Peak similarity, mean similarity and root mean square similarity have similar running times; similarity based on correlation coefficient is about twice slower.

We also used the four metrics to identify close matches for each series, and compared the results with ground-truth neighborhoods. For stocks, we used expert opinion to define these neighborhoods: we consider stocks similar if they belong to the same industry group, according to the classification by Standard and Poor's (see Figure

Table 4.2: Differences between selected similar series.

For each given series, we selected the five most similar series, and measured the mean difference between the given series and the other five, using distance measures given in Figure 3.4. Smaller differences correspond to better selection of similar series. We also show the running time of selecting similar series, for each similarity metric.

Metric	Comp. rate	Stock			Sea temperatures			Air temperatures		
		Mean diff.	Max. diff.	Time (sec)	Mean diff.	Max. diff.	Time (sec)	Mean diff.	Max. diff.	Time (sec)
Exhaustive search		0.094	0.437		.016	.072		.024	.121	
Random selection		0.287	1.453		.078	.215		.070	.235	
Peak similarity	5%	0.110	0.534	.022	.019	.073	.019	.030	.136	.020
	10%	0.103	0.429	.024	.018	.068	.021	.029	.103	.022
Mean similarity	5%	0.126	0.570	.024	.033	.112	.021	.037	.152	.022
	10%	0.110	0.525	.026	.026	.092	.022	.031	.134	.022
Root mean square sim.	5%	0.115	0.588	.024	.031	.106	.021	.035	.147	.022
	10%	0.103	0.497	.026	.024	.090	.022	.030	.133	.022
Correlation coefficient	5%	0.210	1.101	.045	.063	.179	.042	.051	.224	.043
	10%	0.206	1.019	.048	.054	.162	.044	.051	.214	.046

Metric	Comp. rate	Wind speeds			Electroencephalograms		
		Mean diff.	Max. diff.	Time (sec)	Mean diff.	Max. diff.	Time (sec)
Exhaustive search		.021	.136		.038	.170	
Random selection		.029	.185		.072	.370	
Peak similarity	5%	.023	.148	.016	.063	.306	.015
	10%	.023	.138	.016	.052	.241	.015
Mean similarity	5%	.025	.152	.017	.066	.323	.014
	10%	.023	.137	.017	.055	.279	.016
Root mean square sim.	5%	.023	.153	.017	.064	.317	.014
	10%	.023	.134	.017	.051	.261	.016
Correlation coefficient	5%	.024	.154	.033	.068	.349	.028
	10%	.024	.138	.042	.056	.281	.030

4.3). We considered "small neighborhoods," formed by industry sub-categories, as well as "large neighborhoods," formed by industry groups. Gavrilov [2000] used similar classification as ground-truth in testing similarity measures and clustering techniques; however, he used an earlier classification which was different from Figure 4.3.

For air and sea temperatures, we used geographic proximity to define two ground-truth neighborhoods. The first neighborhood is a rectangle around the given buoy. The second neighborhood consists of the two buoys to the east, and the two buoys to the west

Energy

- Baker Hughes
- Halliburton
- Schlumberger
- Coastal
- Occidental Petroleum

Materials

- Du Pont
- Dow Chemical
- International Flavors and Fragrances
- Alcoa
- Homestake Mining
- Bethlehem Steel
- Weyerhauser
- International Paper
- Boise Cascade

Capital Goods

- Boeing
- Honeywell
- United Technologies
- General Dynamics
- Raytheon
- Fluor
- Rockwell International
- General Electric
- Minnesota Mining and Manufacturing
- Catepillar

Transportation

- FedEx
- Delta Air Lines
- Burlington Northern Santa Fe
- Norfolk Southern

Automobiles & Components

- General Motors
- Ford

Consumer Durables & Apparel

- Black & Decker
- Brunswick
- Eastman Kodak
- Polaroid

Hotels Restaurants & Leisure

- Harrah's Entertainment
- McDonald's

Media

- Disney
- Viacom

Retailing

- May Department Stores
- Sears
- Walmart
- K mart
- Limited
- Home Depot
- Toys R Us Holding

Food Beverage & Tobacco

- Coca Cola
- Pepsi
- Campbell Soup
- Heinz
- Ralston-Ralston Purina
- Sara Lee
- Philip Morris

Household & Personal Products

- Proctor & Gamble
- Colgate-Palmolive
- Avon Products

Health Care Equipment & Services

- Baxter Intemational
- Mallinckrodt
- CIGNA

Pharmaceuticals & Biotechnology

- Amgen
- Johnson and Johnson
- Merck
- Bristol-Myers Squibb

Banks

- J P Morgan
- Bank of America
- Bank One
- U. S. Bancorp
- Wells Fargo

Diversified Financials

- American Express
- Citigroup
- Merrill Lynch
- Morgan Stanley, Dean Witter

Insurance

- American General
- American International
- Hartford Financial Services

Software & Services

- America Online
- Ceridian
- Computer Sciences
- Unisys
- Microsoft
- Oracle

Technology Hardware & Equipment

- Cisco
- Lucent
- Nortel Networks Holding
- Hewlett-Packard
- International Business Machines
- EMC
- Tektronix
- Xerox
- Intel
- National Semiconductor
- Texas Instruments

Telecommunication Services

- American Telephone and Telegraph
- SBC Communications
- American Electric Power
- Entergy
- Southern
- Unicom
- Williams

Figure 4.3: Industry groups.

of the given buoy, as shown in Figure 4.5(a). For wind data, we also used geographic proximity. The first neighborhood included sites within 70 miles, and the second included sites within 140 miles. For electroencephalograms, the first neighborhood was a three-by-three neighborhood of electrodes; the second was a five-by-five neighborhood, as shown in Figure 4.5(b).

We applied the available similarity metrics to identify similar series, and then determined how many of the selected series belonged to the same neighborhood. For each compressed series, we found the five most similar ones, and then determined the average number of the series among them that belonged to the same neighborhood as the given series. In Table 4.3, we summarize the results, and compare them with the prefect selection and with random selection.

For stock data, and air and sea temperatures, similarity metrics clearly outperform random selection, with 99% confidence. On the other hand, the results for wind and electroencephalograms are mixed. Recall that the 10% compression does not always preserve electroencephalograms, which is a likely reason for poor selection.

We also measured the correlation between the peak similarity and the three distance measures given in Section 3.2. In Figures 4.5–4.9, we give the results of this experiment for different compression rates; specifically, we show the correlation scatter plot and give the correlation coefficient for the most similar 20% of the points. In Table 4.4, we summarize the correlations.

In Figures 4.10–4.16, we show correlation results for different values of the knob variable *w* in the definition of peak similarity. In Figures 4.17–4.19, we show similar results for different values of the knob *c* in the definition of triangle similarity. By adjusting these knobs, we can obtain high correlation for *high* similarity values. We are less interested in a correlation for low similarity values, since the purpose of the developed technique is retrieval of similar sequences.

Finally, we checked how well peak similarity of compressed data correlates with the similarity of uncompressed data (see Figure 4.20). We observed a good linear correlation, which degraded gracefully with increase of compression rate. The only exception is the electroencephalogram data, which gave poor correlation at 5% compression, which resulted from poor compression accuracy.

```
x  x  x                                                          x  x  x  x  x
x  x  x                                       x  x  x       x  x  x  x  x
x  O x         x  x  O x  x                   x  O x        x  x  O x  x
x  x  x                                       x  x  x       x  x  x  x  x
x  x  x                                                     x  x  x  x  x
(a)                                           (b)
```

Figure 4.4: Buoy and electrode ground-truth neighborhoods.

(a) We considered two neighborhoods of buoys in the experiments with air and sea temperatures, 3x5 and 5x1 neighborhoods. We show the given buoy by an O and its neighbors by x's. (b) We considered 3x3 and 5x5 neighborhoods of electrodes in the experiments with electroencephalograms.

Table 4.3: Ability to find members of the same neighborhood.

For each compressed series, we found the five most similar series, and then determined the average number of the series among them that belong to the same neighborhood as the given series.

Metric	Comp. Rate	Stock		Sea temp.		Air temp.		Wind speed		Encephalogram	
		1	2	1	2	1	2	1	2	1	2
Perfect selection		1.29	4.05	10.5	3.34	10.5	3.34	2.67	8.33	5.84	16.6
Random selection		0.07	0.29	0.40	0.11	0.40	0.10	0.74	2.27	0.35	1.03
Peak similarity	5%	0.21	0.55	1.18	0.65	0.82	0.48	1.50	2.83	0.59	1.25
	10%	0.22	0.62	1.09	0.54	0.89	0.49	1.16	2.83	0.81	1.81
Mean similarity	5%	0.12	0.47	0.75	0.17	0.65	0.25	1.58	2.66	0.36	0.90
	10%	0.18	0.55	0.85	0.28	0.77	0.34	1.33	2.92	1.05	1.98
Root mean square sim.	5%	0.17	0.35	0.77	0.20	0.71	0.26	1.33	2.75	0.36	0.90
	10%	0.14	0.53	0.88	0.32	0.83	0.34	1.50	2.92	1.20	2.19
Correlation coefficient	5%	0.19	0.50	0.72	0.29	0.60	0.34	1.50	2.75	0.68	1.65
	10%	0.15	0.39	0.82	0.25	0.74	0.49	1.33	2.92	1.16	2.24

Table 4.4: Correlation between peak similarity and distance.

We show the correlation of the peak similarity with three distance measures, which include the mean distance, the maximum distance, and the root mean square distance. The correlation is negative, since greater similarity corresponds to smaller distance.

	Stock			Sea temperatures			Air temperatures		
Compression	mean	maximum	root mean square	mean	maximum	root mean square	mean	maximum	root mean square
none	-0.76	-0.65	-0.79	-0.86	-0.93	-0.92	-0.66	-0.82	-0.73
20%	-0.71	-0.63	-0.74	-0.75	-0.84	-0.79	-0.14	-0.39	-0.17
10%	-0.64	-0.56	-0.65	-0.62	-0.75	-0.66	0.21	-0.05	0.19
5%	-0.54	-0.46	-0.55	-0.37	-0.58	-0.40	0.33	0.09	0.32

	Wind			Electroencephalograms		
Compression	mean	maximum	root mean square	mean	maximum	root mean square
none	-0.84	-0.94	-0.89	-0.93	-0.93	-0.96
20%	-0.60	-0.47	-0.58	-0.63	-0.62	-0.64
10%	-0.49	-0.47	-0.51	-0.04	-0.06	-0.05
5%	0.07	-0.42	0.03	0.11	0.10	0.12

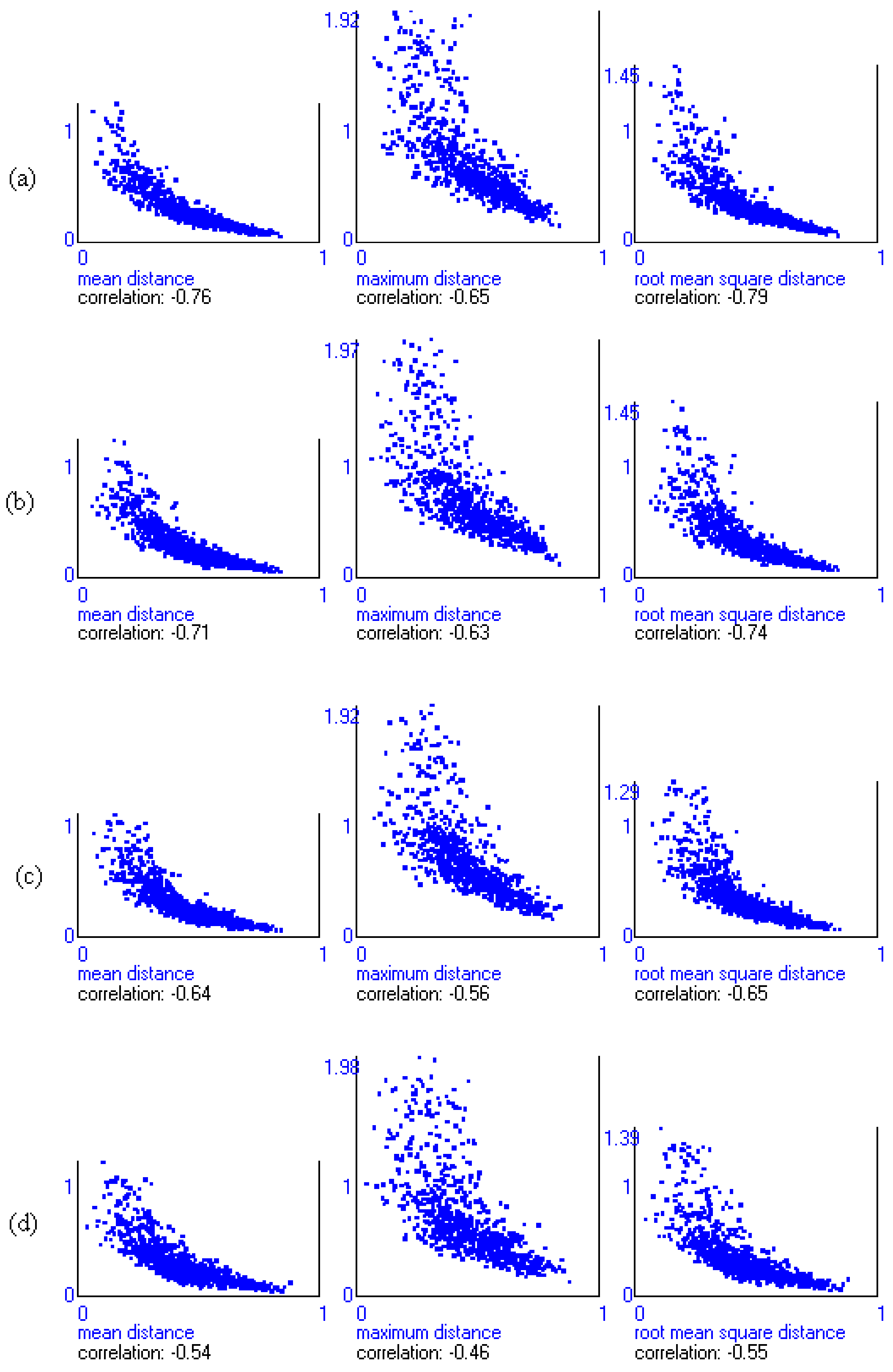


Figure 4.5: Peak similarity versus distance for stock prices.
(a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

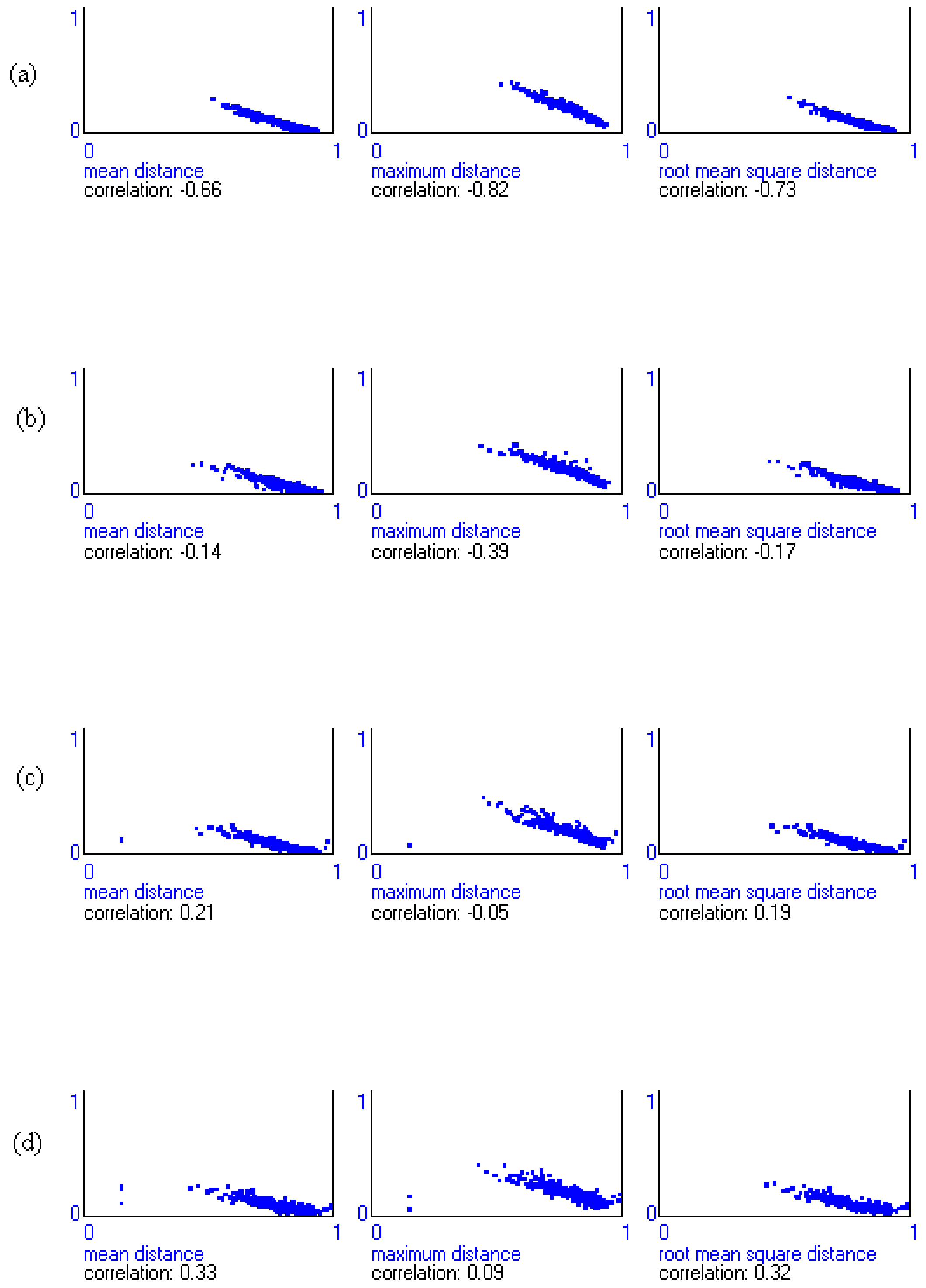


Figure 4.6: Peak similarity versus distance for air temperatures.
(a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

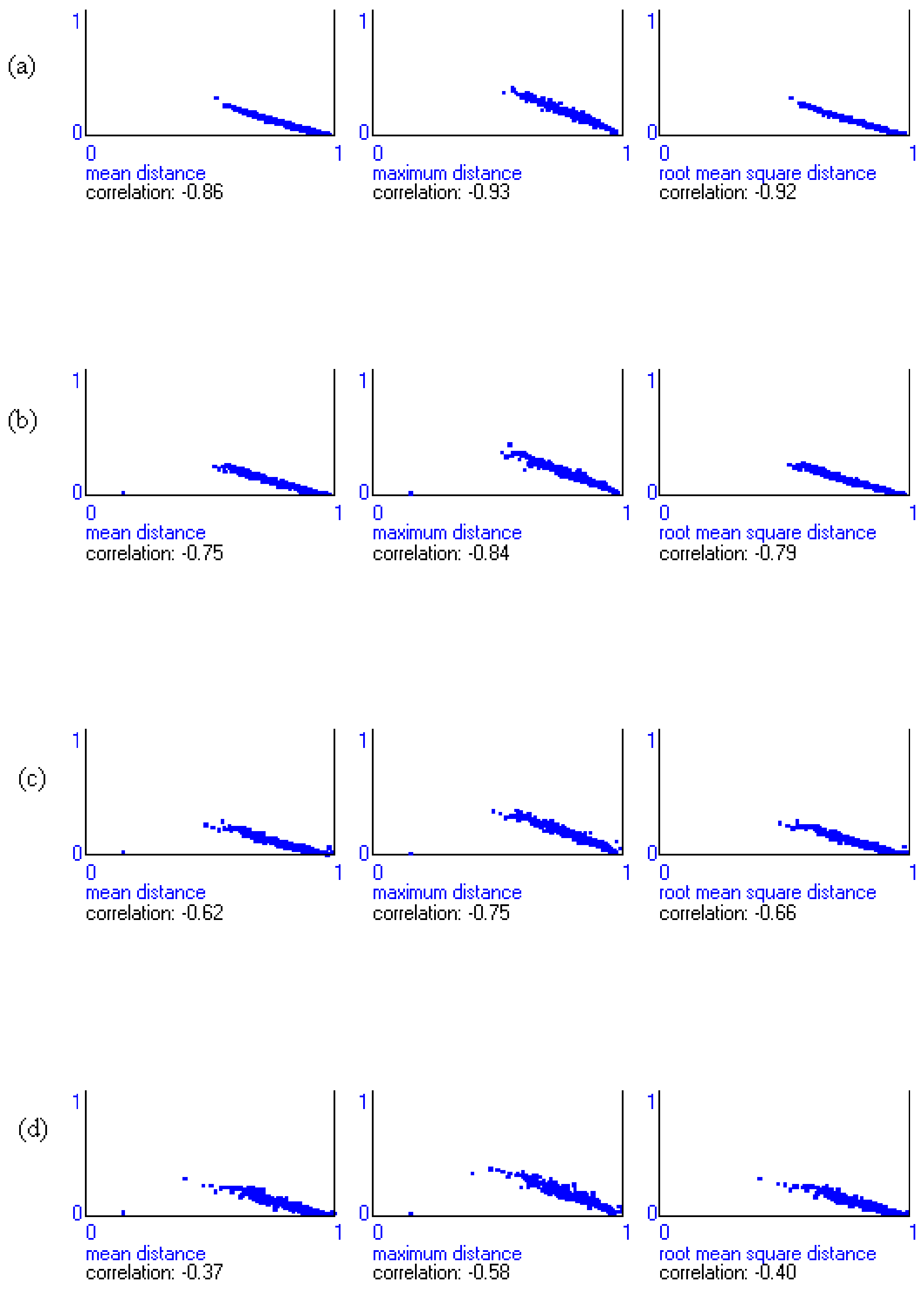


Figure 4.7: Peak similarity versus distance for sea temperatures.
(a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

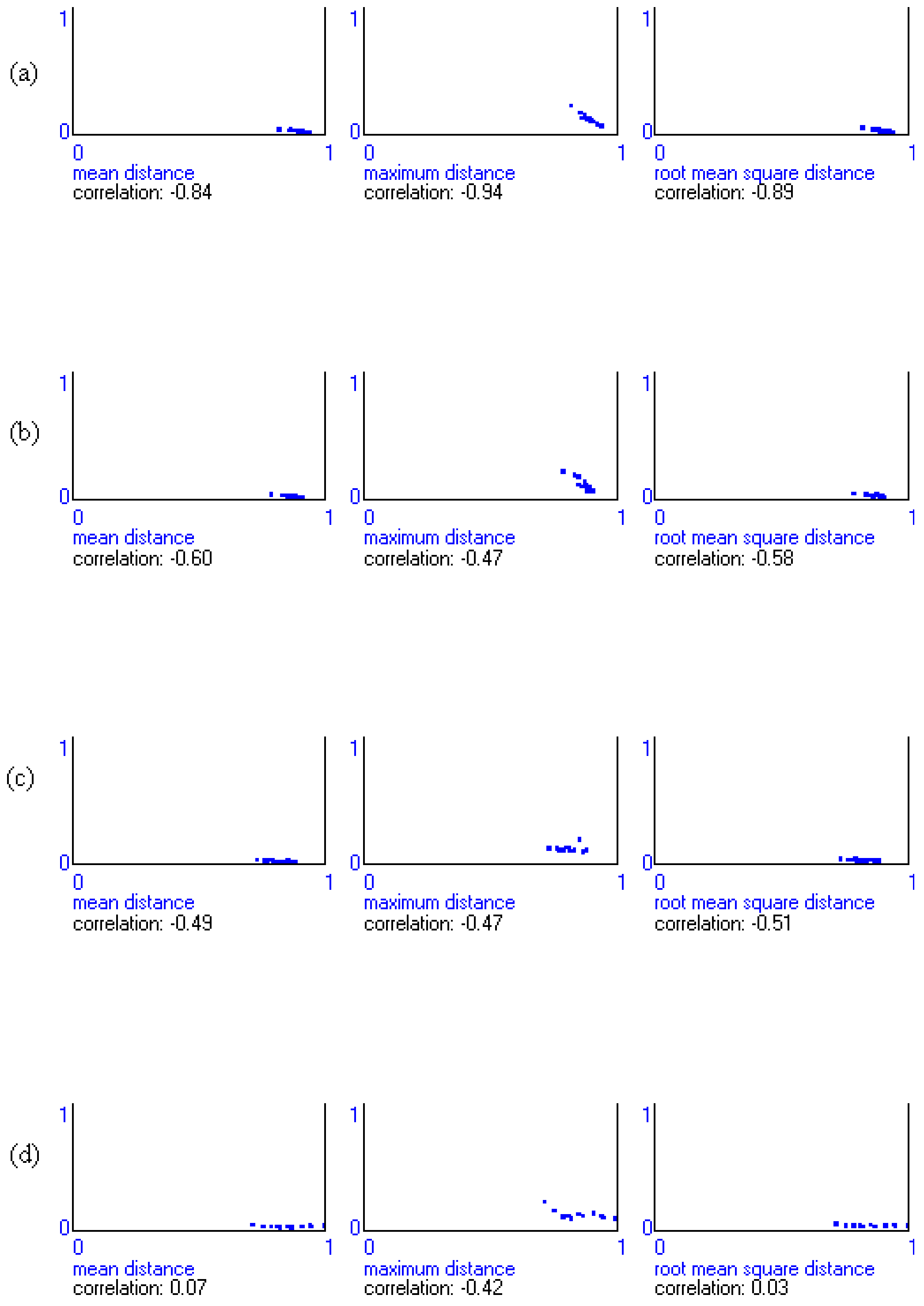


Figure 4.8: Peak similarity versus distance for wind speeds.
(a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

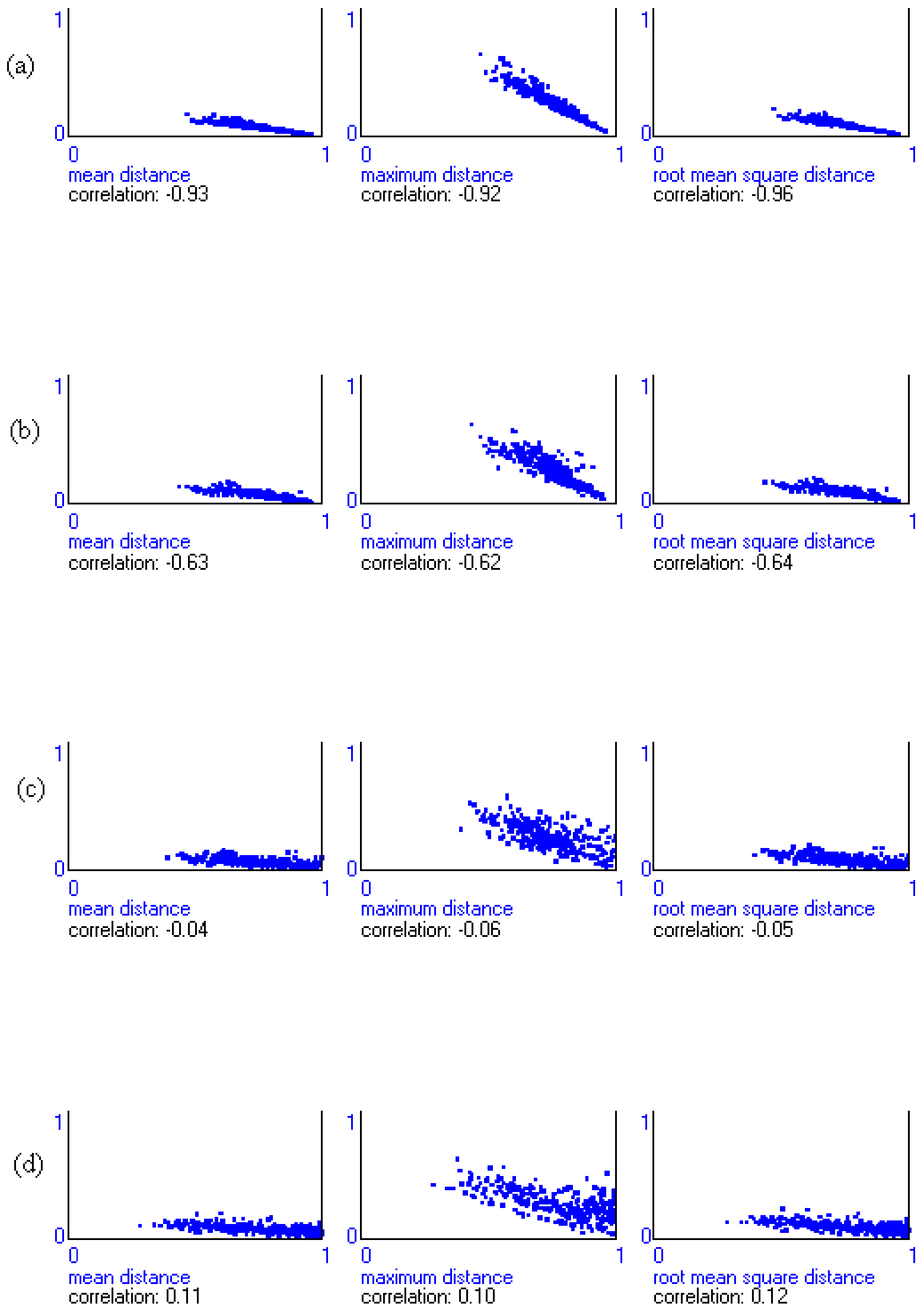


Figure 4.9: Peak similarity versus distance for electroencephalograms.
(a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

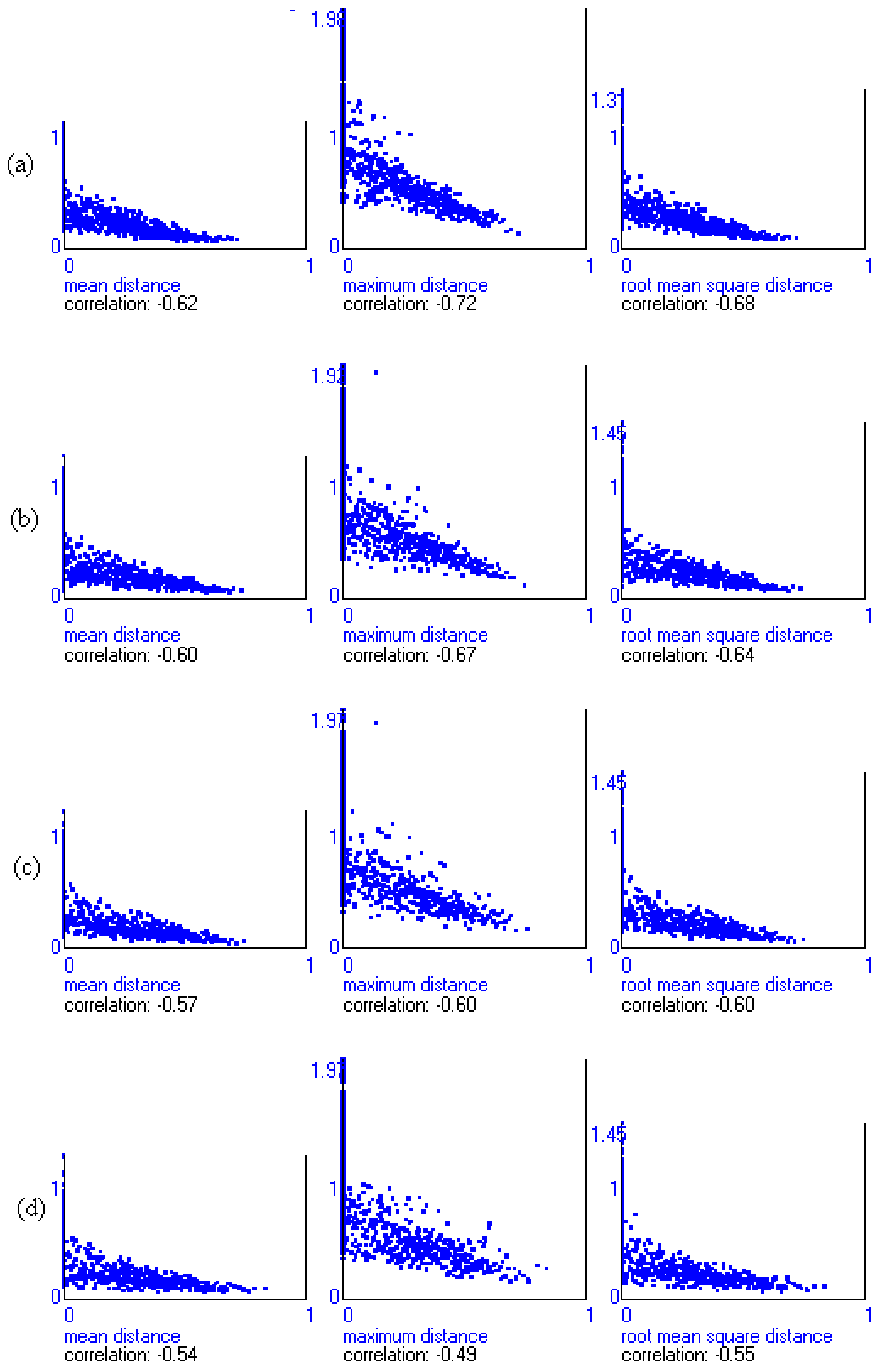


Figure 4.10: Peak similarity versus distance for stock prices, with *w* = 0.
(a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

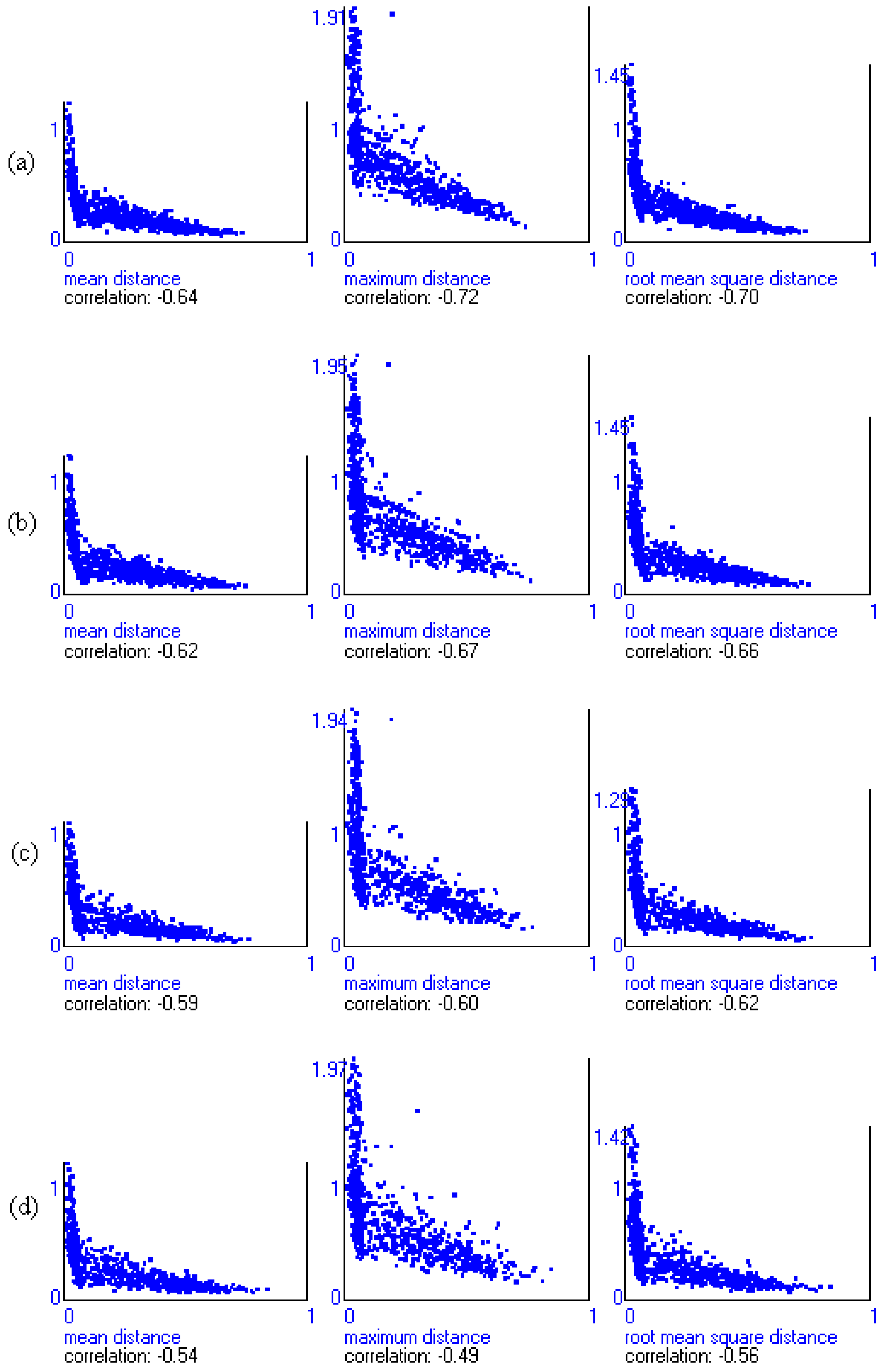


Figure 4.11: Peak similarity versus distance for stock prices, with *w* = 0.10.
(a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

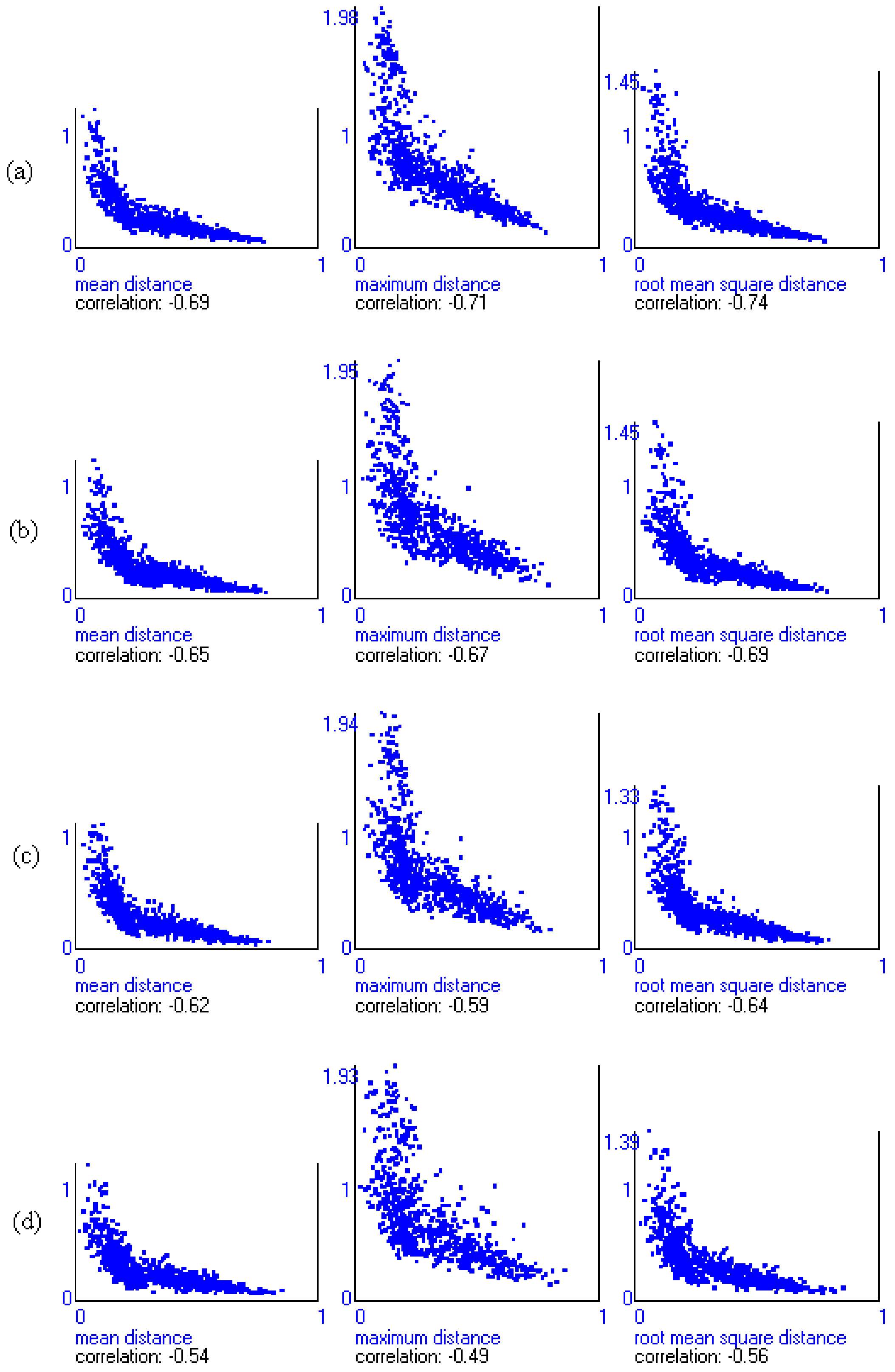


Figure 4.12: Peak similarity versus distance for stock prices, with *w* = 0.34. (a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

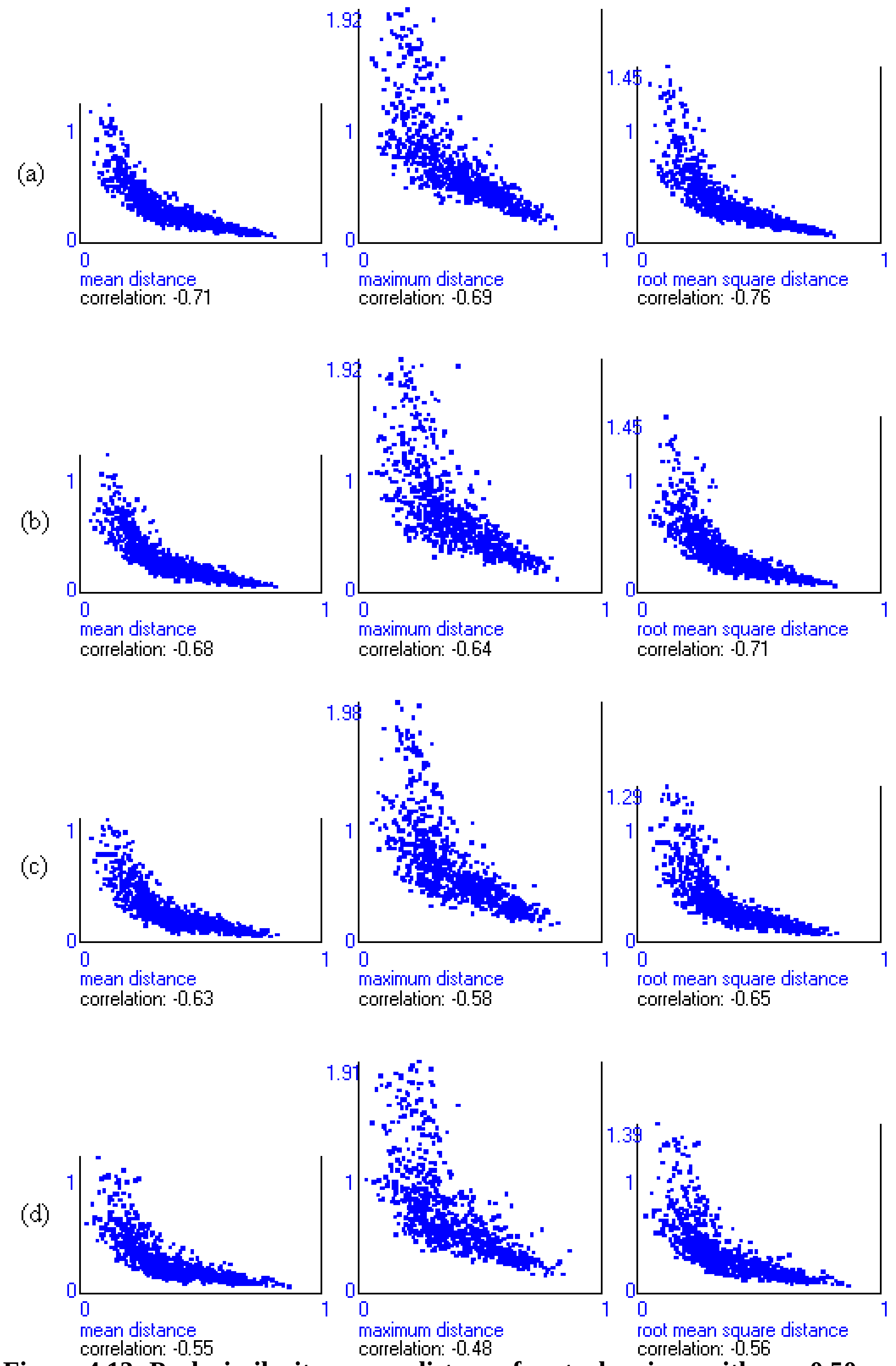


Figure 4.13: Peak similarity versus distance for stock prices, with *w* = 0.50.
(a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

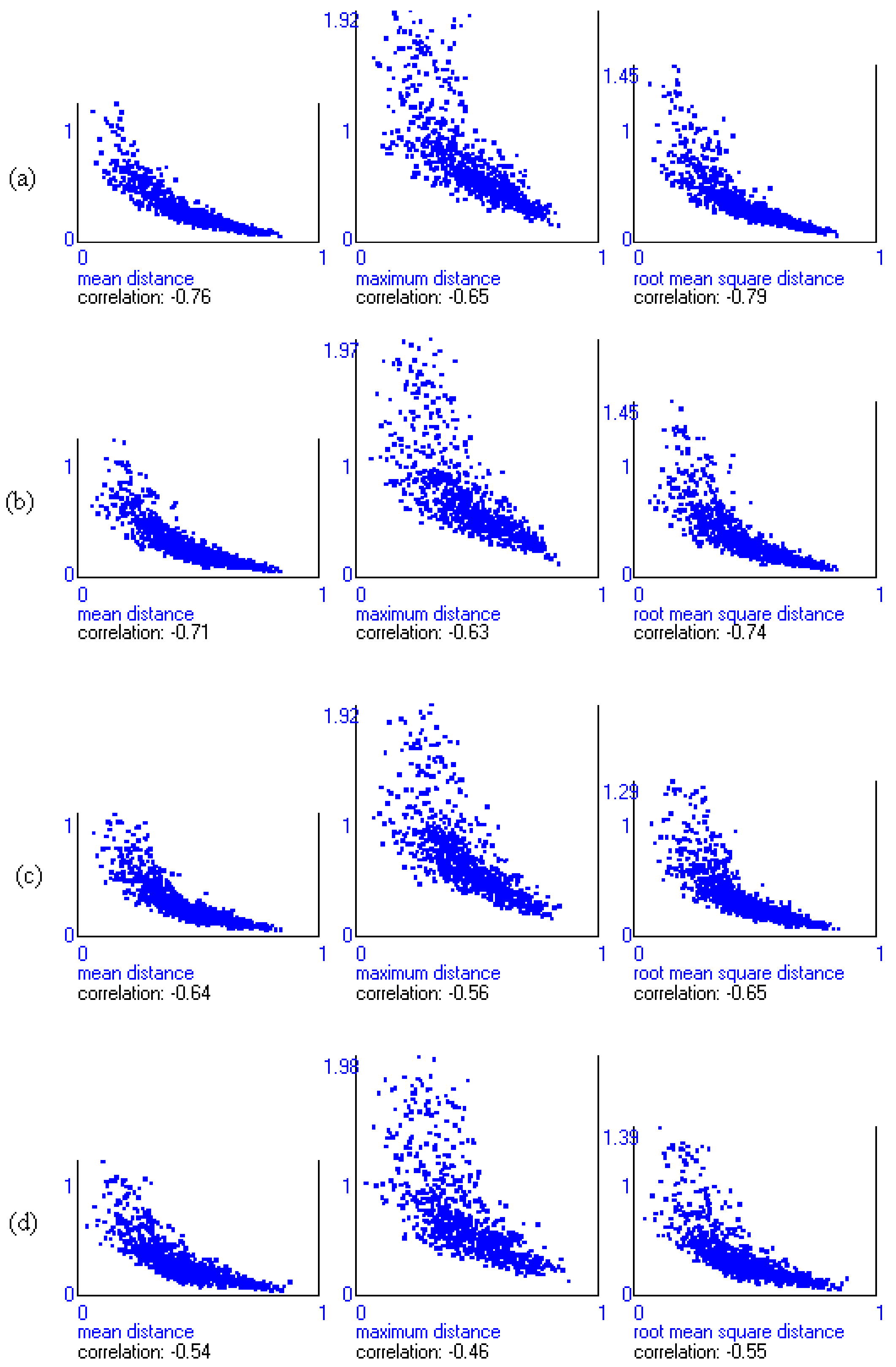


Figure 4.14: Peak similarity versus distance for stock prices, with *w* = 0.66.
(a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

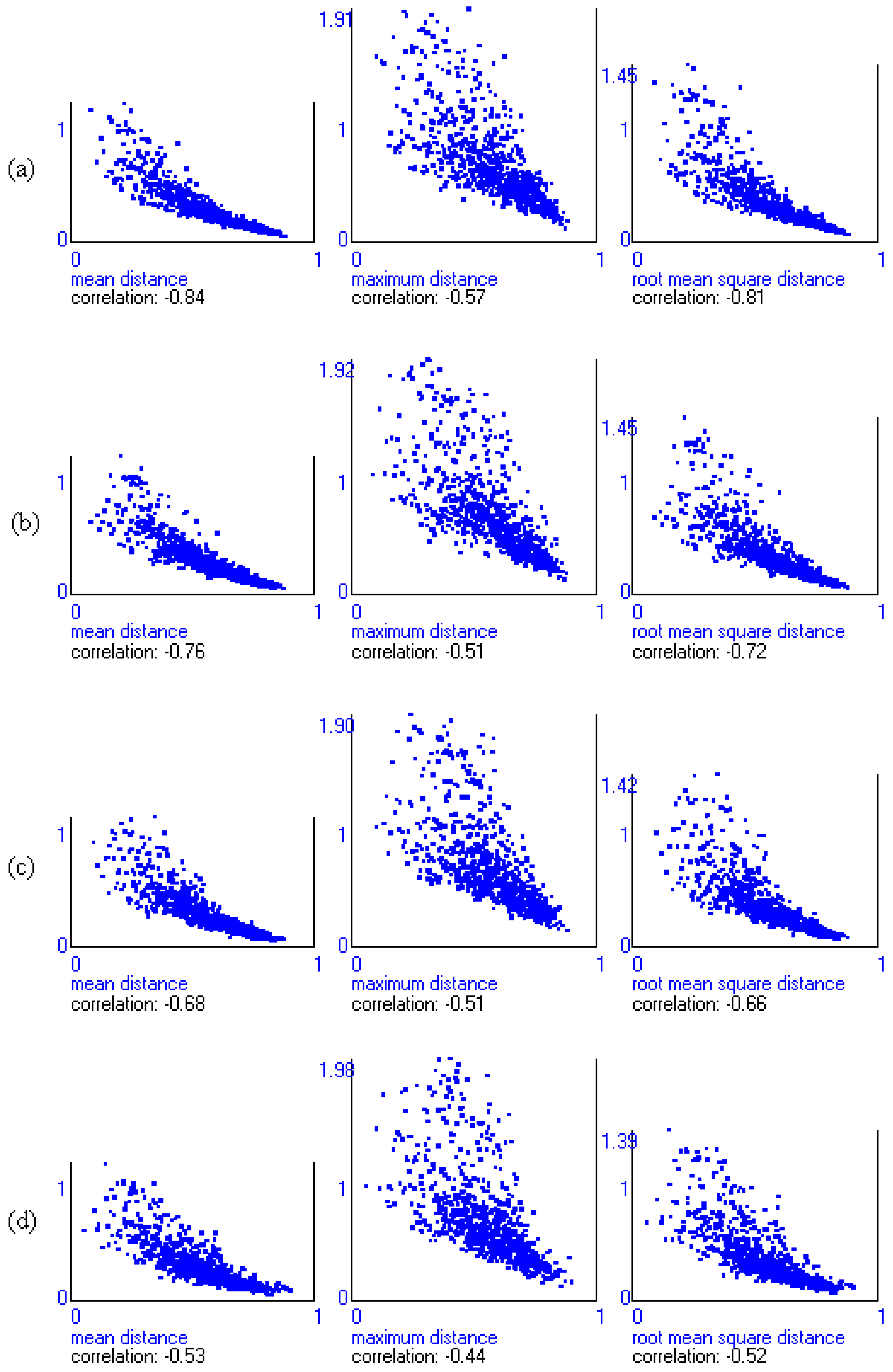


Figure 4.15: Peak similarity versus distance for stock prices, with *w* = 0.90.
(a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

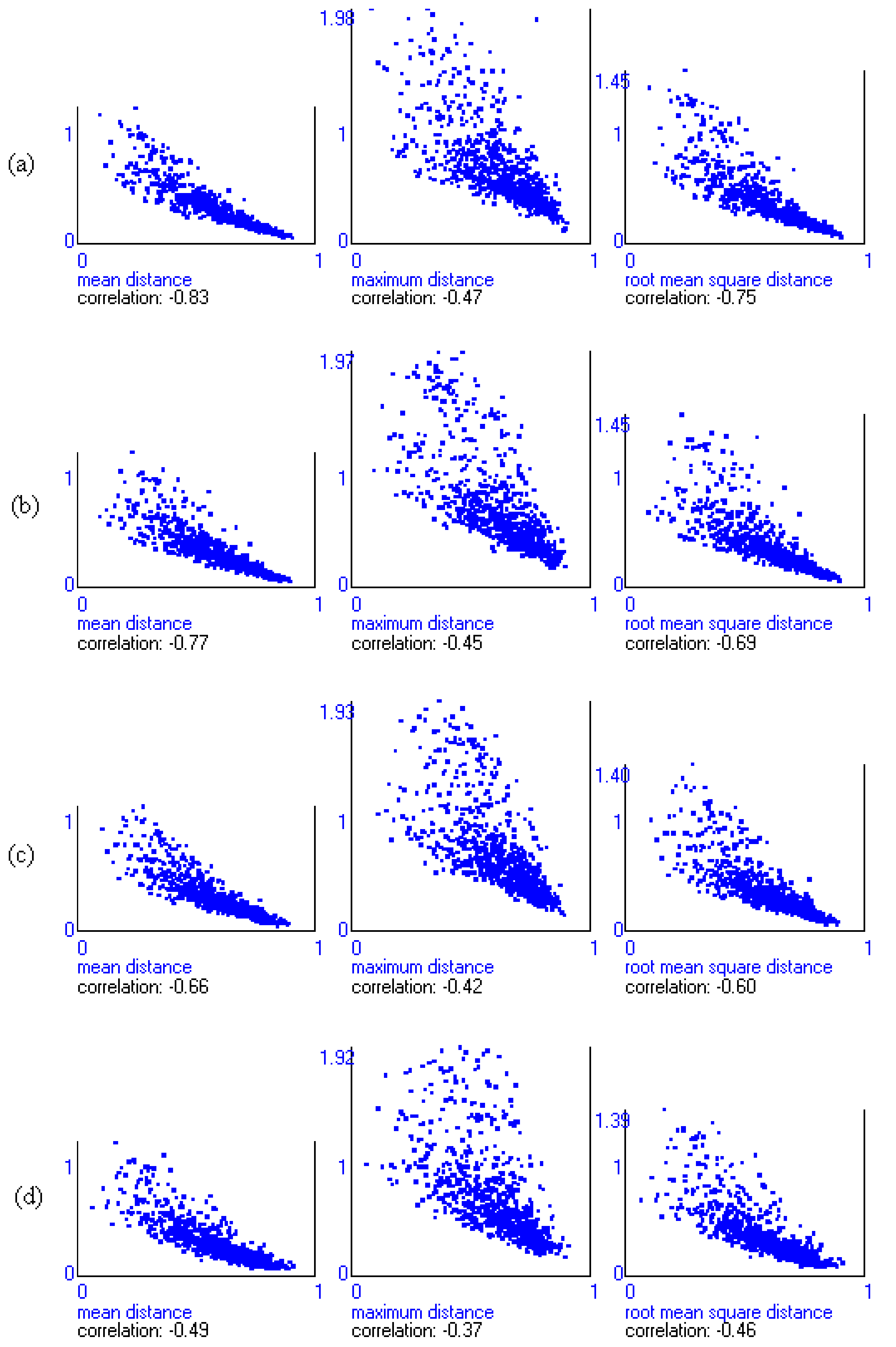


Figure 4.16: Peak similarity versus distance for stock prices, with *w* = 1.00.
(a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

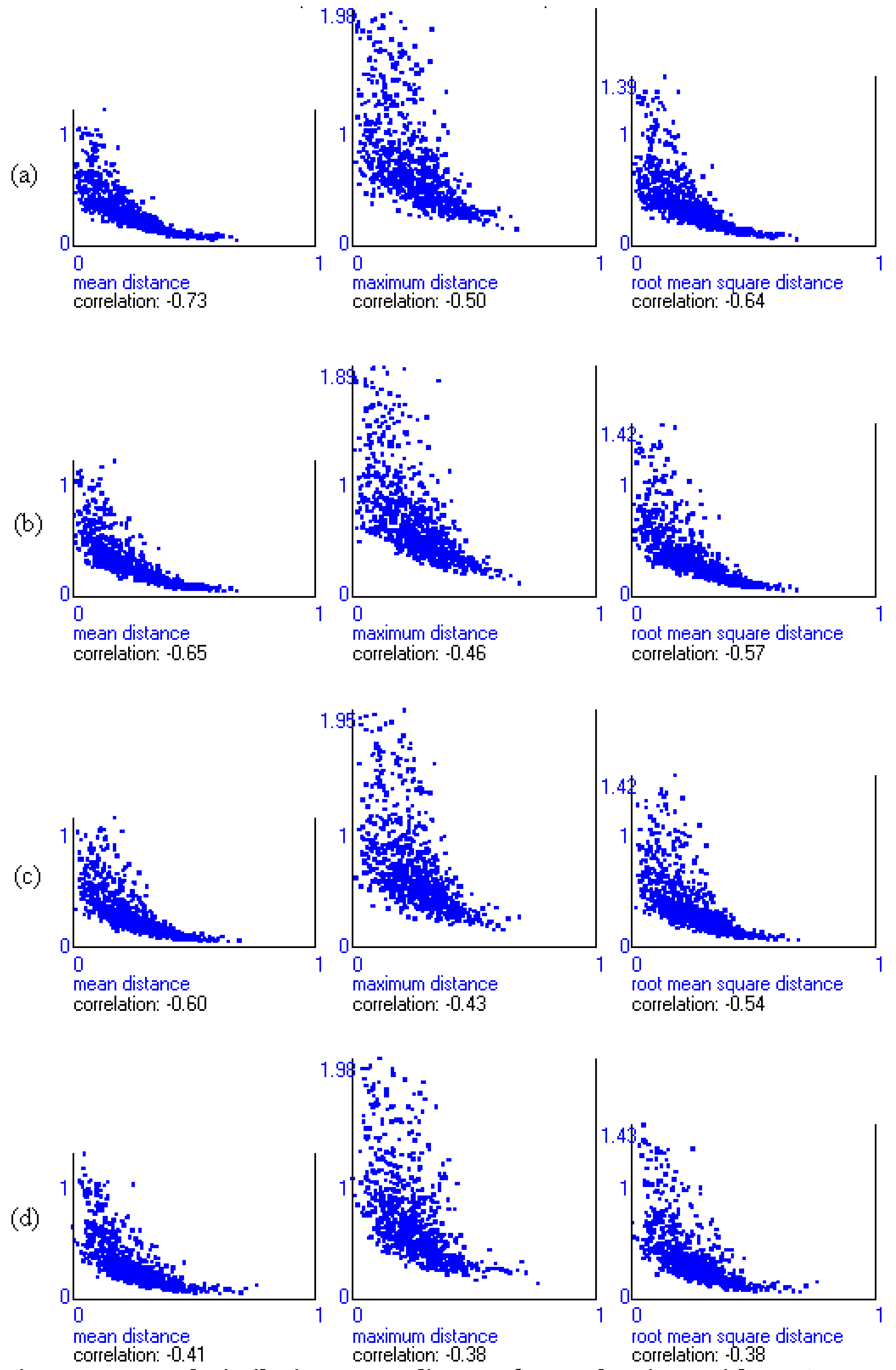


Figure 4.17: Peak similarity versus distance for stock prices, with *c* = 0.25. (a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

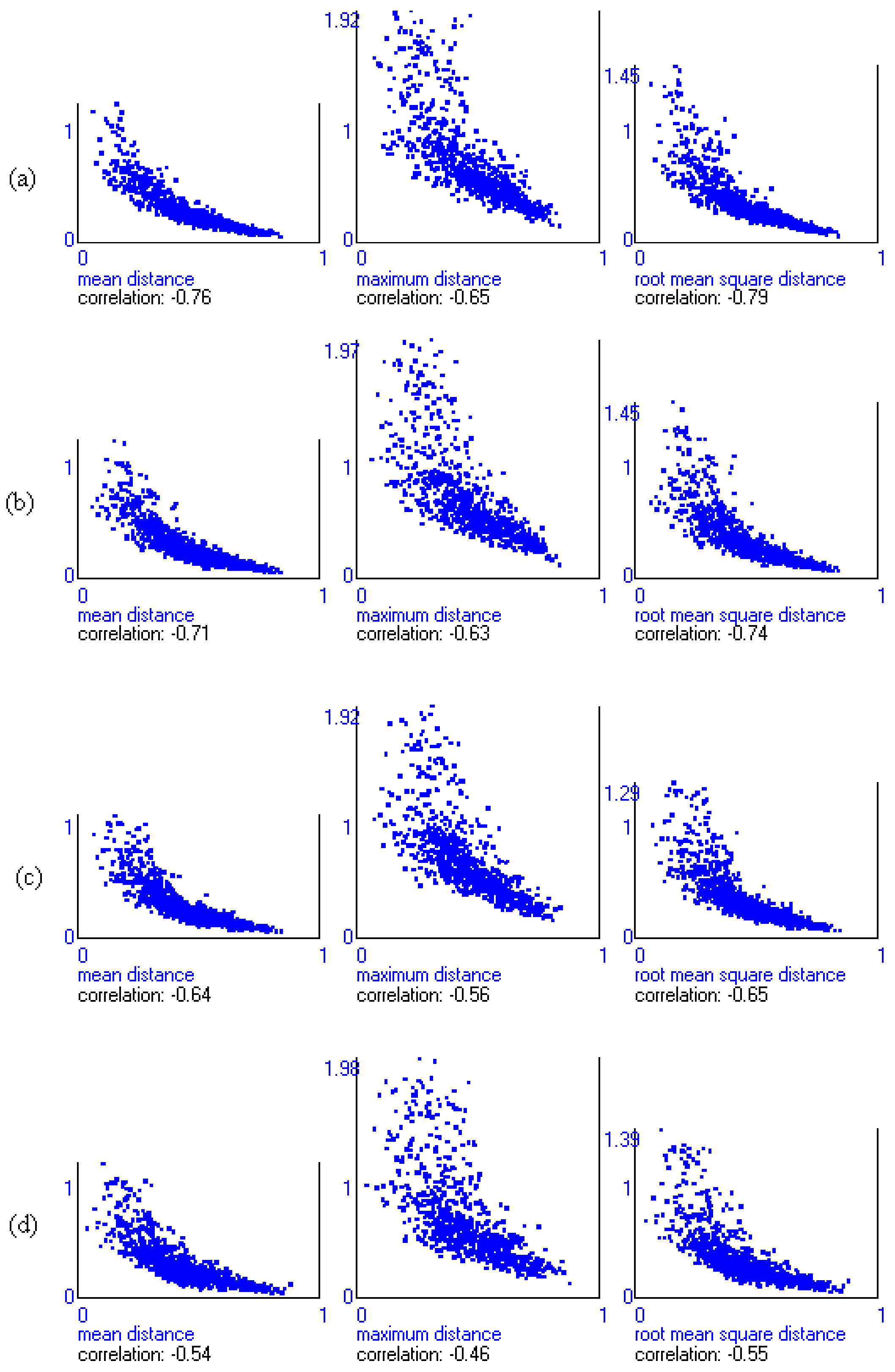


Figure 4.18: Peak similarity versus distance for stock prices, with *c* = 0.50.
(a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

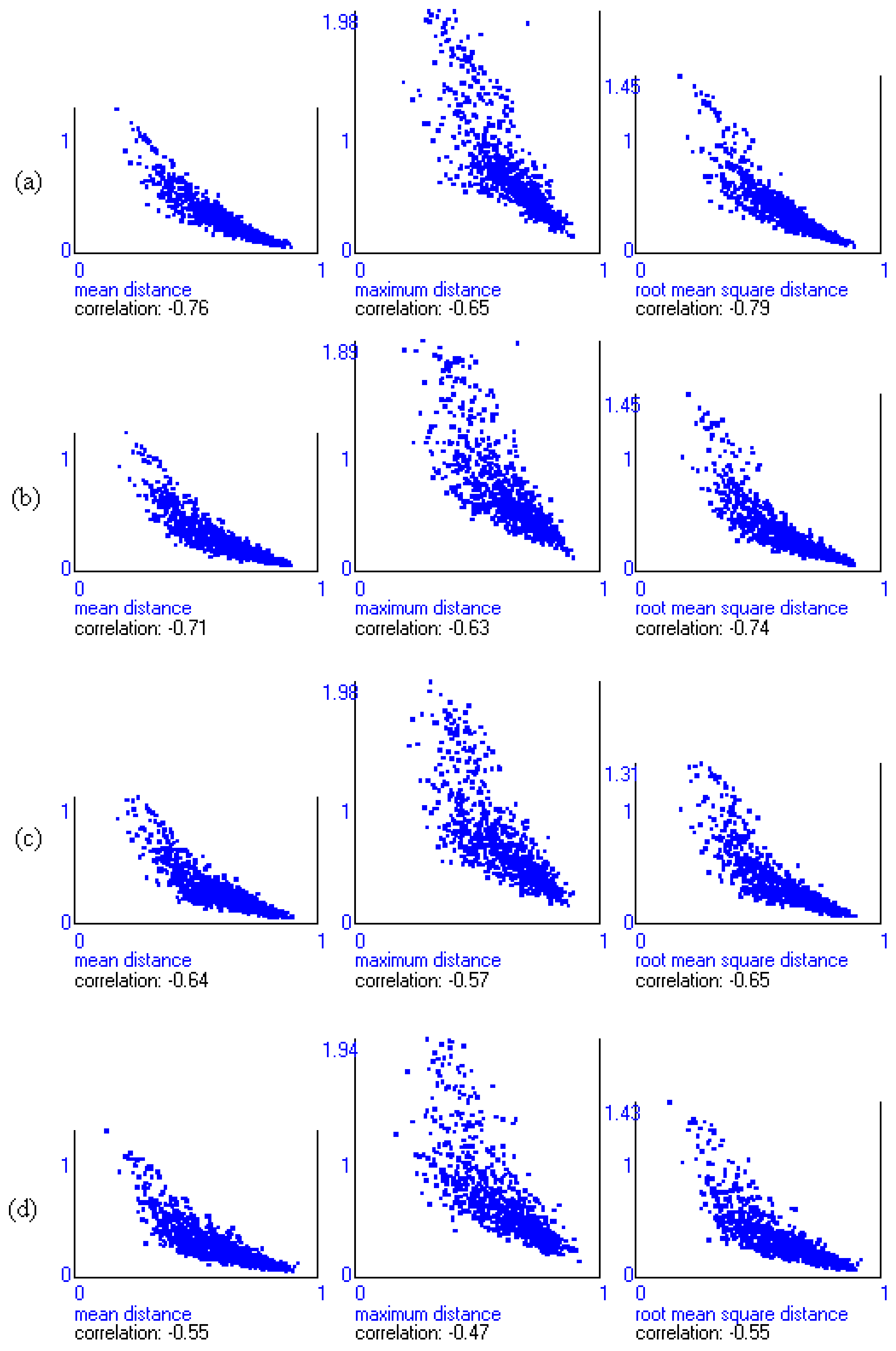


Figure 4.19: Peak similarity versus distance for stock prices, with *c* = 0.75. (a) uncompressed; (b) 20% compression; (c) 10% compression; (d) 5% compression.

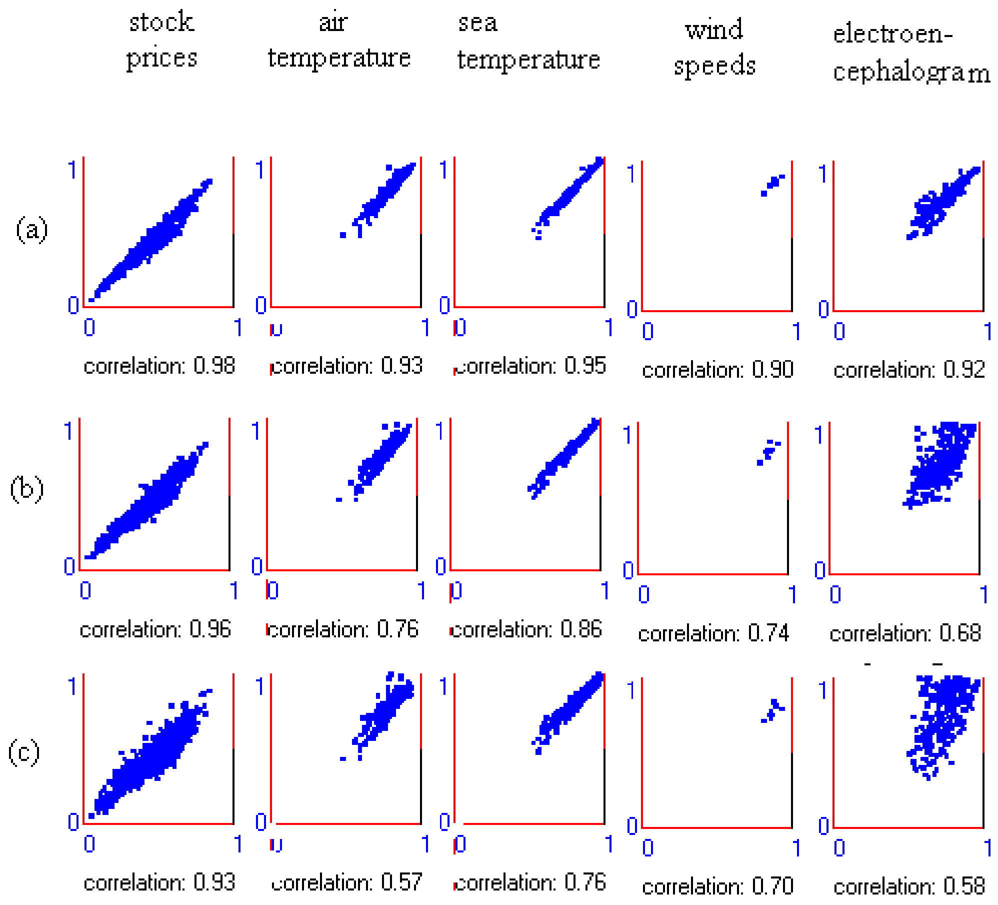


Figure 4.20: Correlation between peak similarity of compressed data and peak similarity of uncompressed data.
(a) 25% compression; (b) 10% compression; (c) 5% compression.

CHAPTER 5 - PATTERN RETRIEVAL

We give an algorithm that inputs a pattern series and retrieves similar series from a database, and then describe the results of applying it to the test domains.

5.1 Retrieval algorithm

The retrieval technique includes three main steps: identifying the "prominent feature" in a given pattern, finding similar features in the series stored in the database, and comparing the pattern with each series containing 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 of a compressed series, we store the values summarized in Figure 5.1, denoted as *vl*, *vr*, *il*, *ir*, *ratio*, and *length*. We give an example of these values in Figure 5.2. The *prominent leg* of a pattern series is the leg with the greatest ratio; for example, the prominent leg in Figure 5.2 is leg 4.

The retrieval algorithm inputs a compressed pattern series, and outputs similar segments of compressed series in the database. We summarize the algorithm in Figure 5.3.

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 in the original sequence
ir	index of the right important point in the original sequence
ratio	ratio of the end-points, defined as (*vr/vl)*
length	length of the segment, defined as (*ir – il)*

Figure 5.1: Basic data for a leg.

For each leg, we store the values related to pattern retrieval.

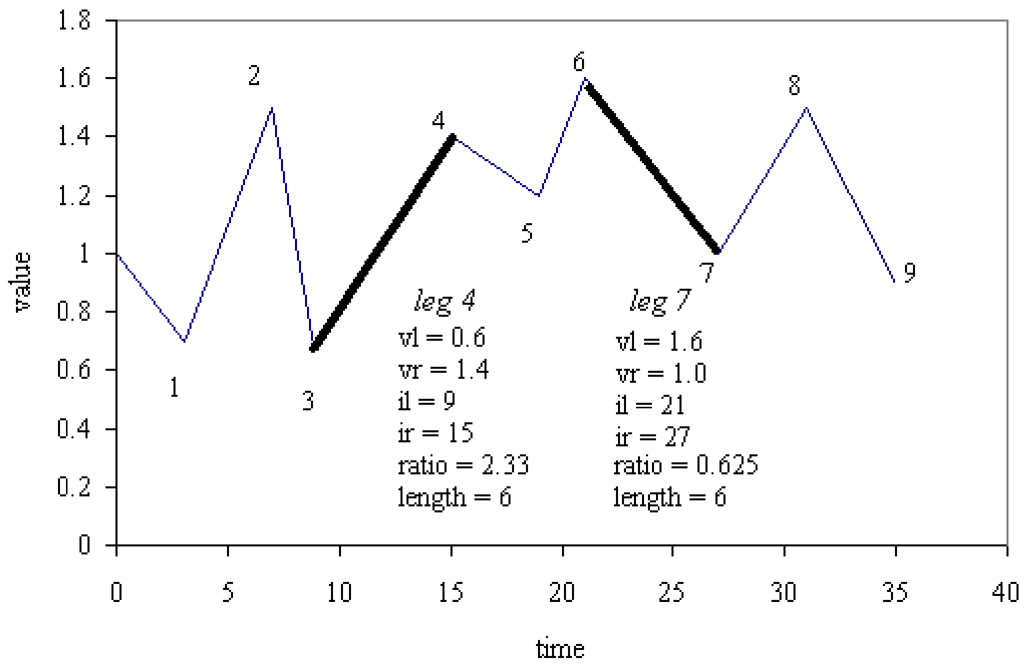


Figure 5.2: **Example of legs.**

A leg is a segment of a compressed series between two important points. We show the basic data for the two legs marked by thick lines.

First, the algorithm searches the pattern for the leg with the greatest end-point ratio, denoted *ratio*$_p$, and identifies all legs in the database that have a similar ratio. A ratio is considered similar to *ratio*$_p$ if its value is between $ratio_p/C$ and $ratio_p \cdot C$, where C is a parameter for controlling the matching process.

To ensure efficient retrieval, we index all legs in the database by their ratio, using a red-black binary search tree. If the total number of legs of all series in the database 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 algorithm discards those legs whose length is not similar to that of the pattern's prominent leg. The length is considered similar when it is between $length_p/D$ and $length_p \cdot D$, where $length_p$ is the length of the pattern leg and D is another knob parameter.

```
PATTERN-RETRIEVAL
Identify the pattern leg $p$ with the greatest end-point ratio, denoted $ratio_p$.
Find all legs in the database with end-point 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 remaining selected legs:
    Identify the segment corresponding to the pattern (see Figure 5.4).
    Compute the similarity between this segment and the pattern.
    If the similarity is above the threshold T, then output the segment.
```

Figure 5.3: Search for segments similar to a given pattern.

The algorithm inputs a compressed pattern series, and searches for matches in a database of compressed time-series. We use three knobs to control the search: maximal ratio deviation *C*, maximal length deviation *D*, and similarity threshold *T*.

Finally, the algorithm compares the pattern segment with segments that contain the selected legs, and measures similarity to the pattern. In Figure 5.4, we illustrate the procedure for identifying the segment of a series that may match a pattern. If the similarity is above a given threshold *T*, the algorithm outputs the segment as a match.

Although we define prominence of a leg as its end-point ratio, we may use the same algorithm with different prominence measures. We have experimented with the alternative measures summarized in Figure 5.5; we give the results of using them in Section 5.3.

5.2 Extended legs

The described algorithm can miss matching series that do not have a leg corresponding to the pattern's prominent leg. We illustrate this problem in Figure 5.6, where the prominent leg of the pattern has no equivalent in the matching series.

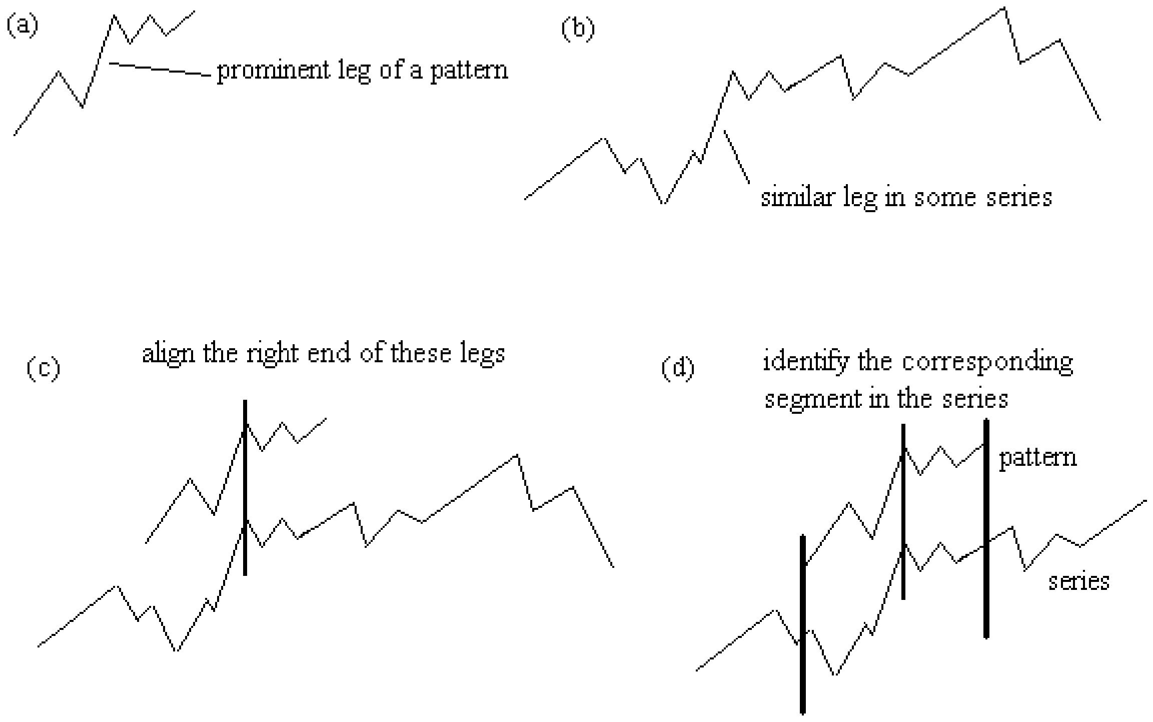

(a)
prominent leg of a pattern
(b)
similar leg in some series
(c)
align the right end of these legs
(d)
identify the corresponding segment in the series
pattern
series

Figure 5.4: Identifying a segment that may match the pattern.

(a) *Slope of a leg*: $(vr - vl)/(ir - il)$

(b) *Length of a leg, with time-scale knob s:* $\sqrt{(vl - vr)^2 + s^2 \cdot (il - ir)^2}$

(c) *Height of the spike formed by two adjacent legs:* $vr_1 - (vl_1 + vr_2) / 2$

Figure 5.5: Alternative prominence measures.

To avoid this problem, we introduce the notion of *extended legs*. Intuitively, a segment of a sequence is an extended leg if it would be a leg under a higher compression rate. Since a compressed series includes all maxima and minima that would be part of the series under higher compression, we can identify all extended legs, as shown in Figure 5.6(c).

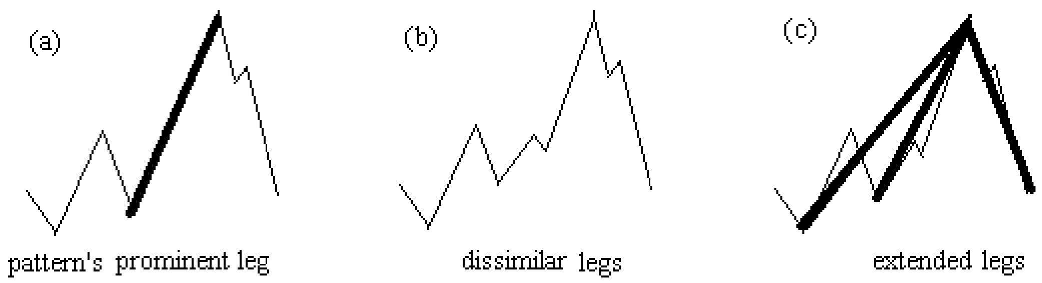


Figure 5.6: Example of extended legs.

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

Formally, two points *i* and *j* of a compressed series $a_1, \ldots, a_n$ form an extended upward leg if

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

The definition of an extended downward leg is symmetric.

We identify all extended legs of all series in the database, compute the same values as for normal legs, and use them in indexing and retrieval in the same way as normal legs. The advantage of this approach is more accurate retrieval, and the disadvantage is larger storage space. In the worst case, a compressed *n*-leg series can give rise to $n^2/2$ extended legs; however, if sequences in the database do not have an upward or downward trend, the average number of extended legs is $\theta(n \cdot lg\ n)$.

In Figure 5.7, we give an algorithm for identifying upward extended legs in a time-series; the procedure for finding downward legs is symmetric. The algorithm consists of two parts, called NEXT-POINTS and EXTENDED-LEGS. We assume that normal upward legs in the input series are numbered from 1 to *n*, and the main loop of each procedure processes them in order.

```
NEXT-POINTS
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[TOP(S)] = $k$; POP(S)
    PUSH($k$)
while S is not empty do
    next[TOP(S)] = NIL; POP(S)

EXTENDED-LEGS
initialize an empty list of extended legs
for $k$ = 1 to $n$ 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 5.7: Identifying extended legs of a compressed series.

The first part processes local maxima of the compressed series; for each maximum ir_k, it identifies the *next larger* maximum in the series, and stores the index of the next larger maximum in next[k]. Its running time is linear in the length of the compressed series. The second part uses this information to identify extended legs. Its running time is linear in the total number of extended legs.

5.3 Search results

To evaluate the retrieval accuracy, we compared search results with the segments identified by a slow exhaustive-search procedure. We show the patterns used in this experiment in Figure 5.8, and summarize the results in Figures 5.9–5.16. We ranked the matches found by the algorithm, from most to least similar, and enumerated them in this order. In Figures 5.9–5.16, we plotted the numbers of matches found by the fast algorithm versus the numbers of exhaustive-search matches. For instance, if the fast retrieval algorithm missed the two best matches and marked the third closest match as the best one, then the graph would include the point (1,3). As another example if the fast retrieval algorithm found only three among seven closest matches, and marked the seventh closest match as the third best, then the graph would include the point (3,7). A perfect result would be a forty-five degree line, indicating that the fast retrieval algorithm had found the same segments as the exhaustive search. If the fast procedure missed some of the segments, the graph slope was steeper.

We ran this experiment with the end-point ratio prominence, as well as with the three alternative prominences listed in Figure 5.5, and with three different value of the knob *C*. We summarize the results in Table 5.1, which shows that the end-point ratio and spike prominence give better results than the other two prominence measures. It also shows that the increase of *C* leads to more accurate identification of similar patterns, at the expense of greater search time.

The retrieval time grows linearly with the pattern length and with the number of candidate sequences identified at the first two steps of the retrieval algorithm. In Figure 5.17, we show the dependency of the running time on these two parameters, for the Visual Basic 6.0 implementation on a 300 MHz PC. If the pattern includes *m* legs and the algorithm identifies *k* candidate matches, then the retrieval time is about $0.07 \bullet m \bullet k$ milliseconds per match. For a database with a total of 5000 legs, the retrieval takes from 0.25 to 3.25 seconds, depending on the pattern length and *C* value.

In Figure 5.18, we give examples of sequences retrieved from a stock database, for a six-leg pattern.

Table 5.1: Experiments with different prominence definitions and values of *C*.

For each experiment, we give the number of candidate matches and the mean similarity of the best ten matches among the candidates. If we increase *C*, the algorithm finds more candidates and misses fewer matches; however, the retrieval time is proportional to the number of candidates.

	Ratio prominence		Slope prominence		Length prominence		Spike prominence	
value of *C*	Num. candi- dates	Similar- ity of best ten	Num. candi- dates	Similar- ity of best ten	Num. candi- dates	Similar- ity of best ten	Num. candi- dates	Similar- ity of best ten
Stock prices: 5413 legs in the database; retrieval of pattern in Figure 5.8(a).								
1.5	335	.9994	852	.9991	555	.9990	475	.9994
2	659	.9993	1360	.9992	972	.9991	926	.9996
5	1957	.9994	2361	.9995	2817	.9995	2254	.9998
Stock prices: 5413 legs in the database; retrieval of pattern in Figure 5.8(b).								
1.5	640	1	719	.9999	716	1	1202	1
2	979	1	1201	.9999	1297	1	1774	1
5	2228	1	2292	1	3117	1	2467	1
Stock prices: 5413 legs in the database; retrieval of pattern in Figure 5.8(c).								
1.5	1114	.9998	984	.9999	163	.9998	1182	.9999
2	1705	.9999	1544	.9999	375	.9998	1783	.9999
5	2562	.9999	2361	.9999	1701	.9999	2375	.9999
Stock prices: 5413 legs in the database; retrieval of pattern in Figure 5.8(d).								
1.5	647	.9985	339	.9980	474	.9982		
2	964	.9986	604	.9982	941	.9983		
5	2383	.9987	1676	.9984	2237	.9985		
Air and sea temperatures combined: 5557 legs in the database; retrieval of pattern in Figure 5.8(e).								
1.5	257	.9999	184	.9994	115	.9999	769	.9995
2	518	1	327	.9995	210	1	1184	.9995
5	2245	1	862	.9996	637	1	1489	.9995
Sea temperatures: 200 legs in the database; retrieval of pattern in Figure 5.8(f).								
1.5	11	.9971	14	.9977	31	.9966	40	.9978
2	17	.9979	16	.9978	54	.9970	69	.9981
5	78	.9981	55	.9981	132	.9974	93	.9981
Wind speeds: 10,591 legs in the database; retrieval of pattern in Figure 5.8(g).								
1.5	1941	.9904	584	.9909	522	.9885	723	.9905
2	3378	.9905	1163	.9916	1306	.9891	1529	.9911
5	5025	.9905	3869	.9918	6527	.9903	4324	.9918
Electroencephalogram: 2898 legs in the database; retrieval of pattern in Figure 5.8(h).								
1.5	150	.9979	380	.9997	102	.9981	339	.9982
2	289	.9980	656	.9998	159	.9985	572	.9983
5	891	.9988	1193	.9999	595	.9992	1023	.9989

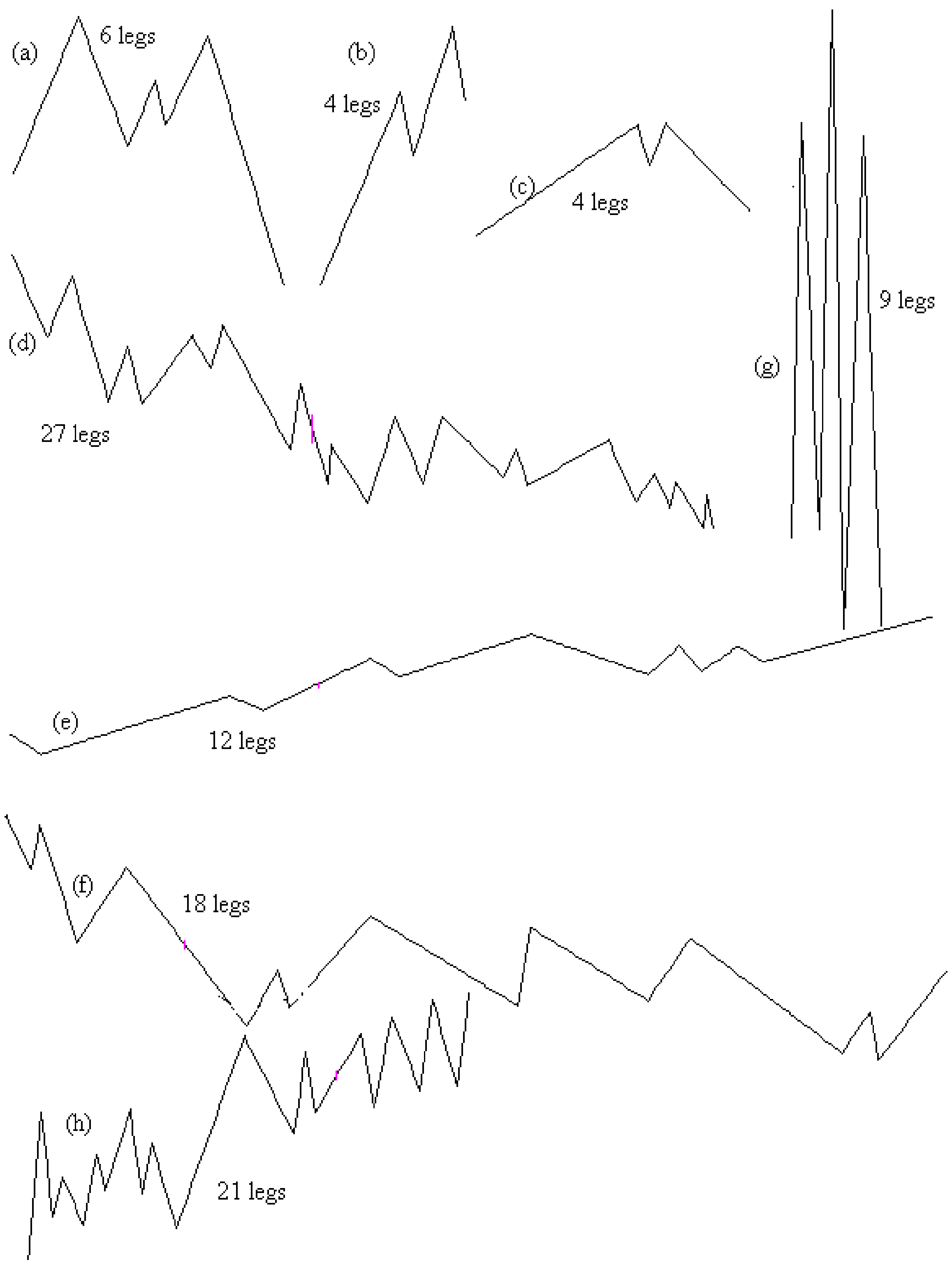


Figure 5.8: Patterns used in the retrieval experiments.

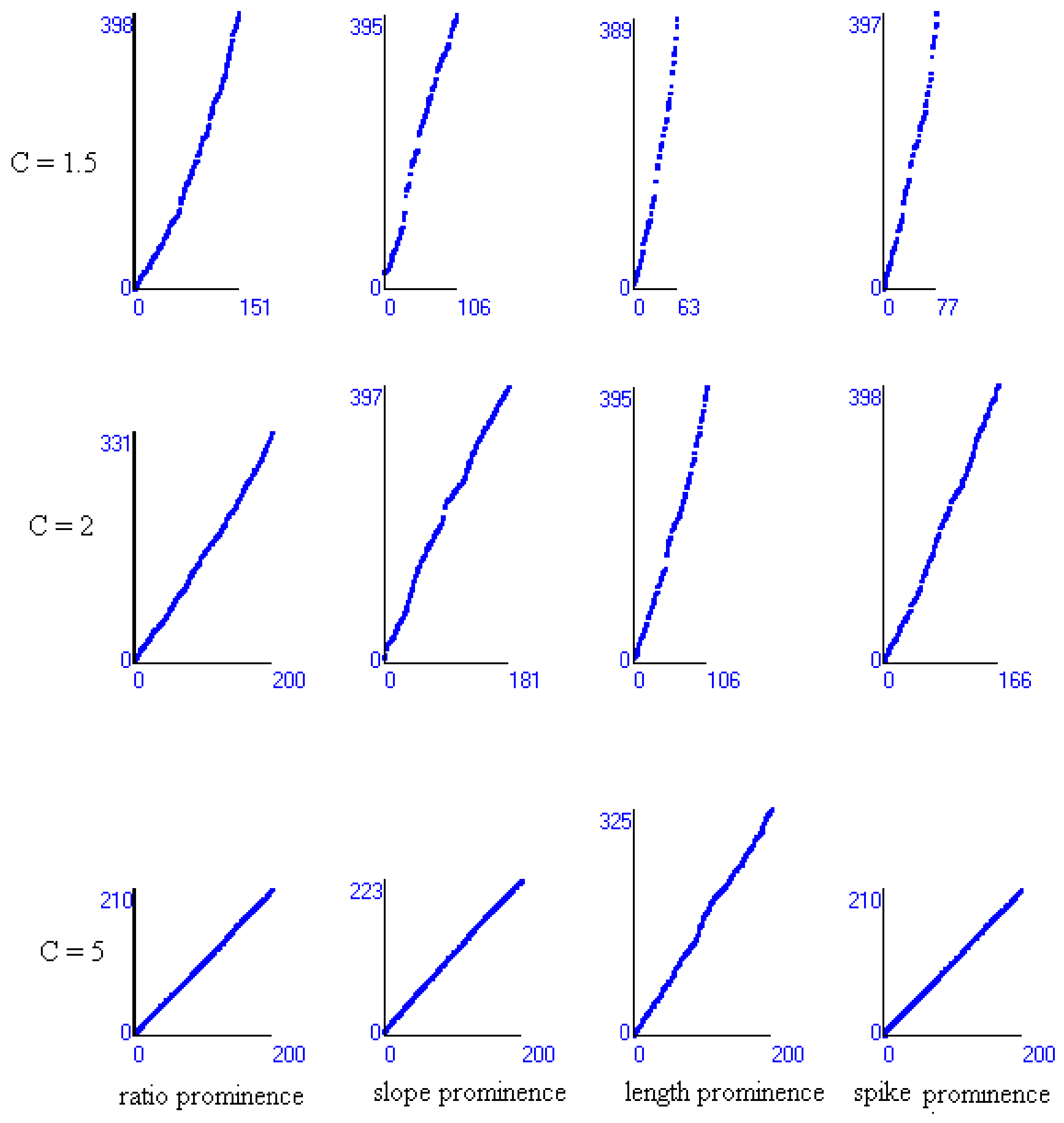


Figure 5.9: Retrieval of stock charts matching the pattern in Figure 5.8(a).

The horizontal axes show the number assigned to the retrieved matches by the fast retrieval algorithm, in the best-to-worst order. The vertical axes are the numbers assigned to the same matches by the exhaustive-search algorithm. If the fast algorithm has found all close matches, then the graph is a forty-five degree line. On the other hand, if the algorithm missed some matches, the line is steeper.

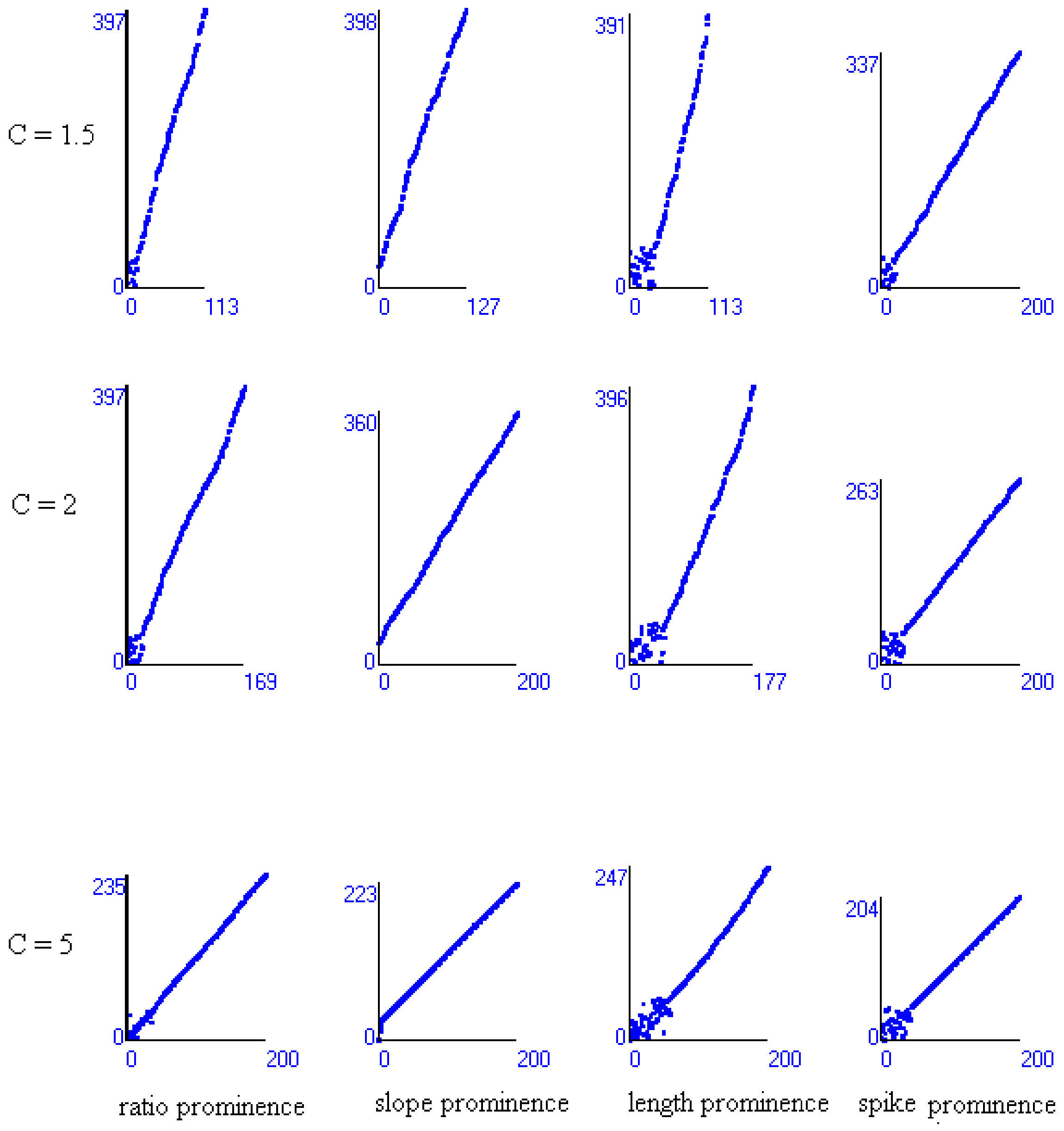


Figure 5.10: Retrieval of stock charts matching the pattern in Figure 5.8(b).

The horizontal axes show the similarity numbers assigned by the fast retrieval algorithm; the vertical axes are the exhaustive-search similarity numbers for the same points.

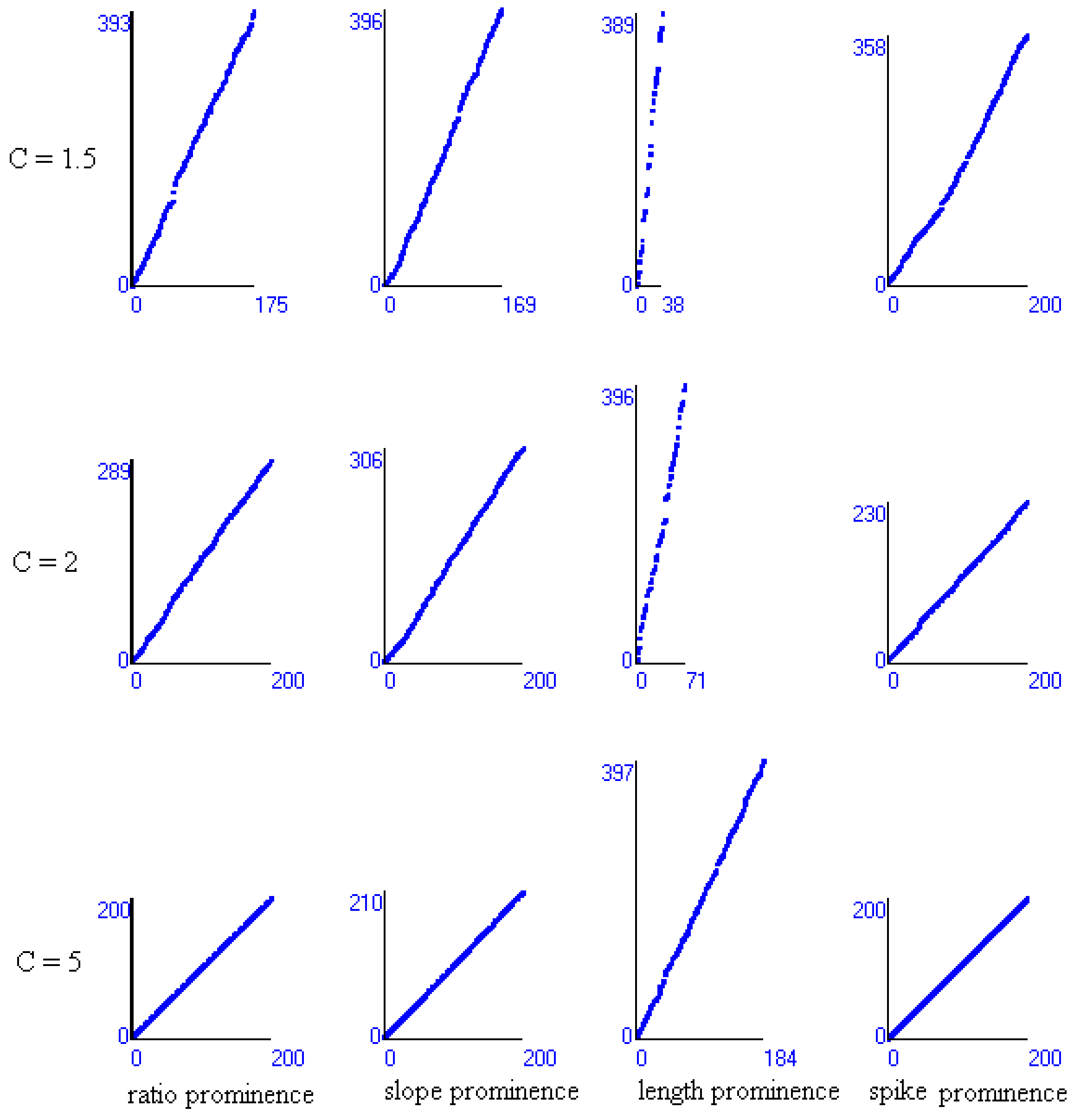


Figure 5.11: Retrieval of stock charts matching the pattern in Figure 5.8(c).

The horizontal axes show the similarity numbers assigned by the fast retrieval algorithm; the vertical axes are the exhaustive-search similarity numbers for the same points.

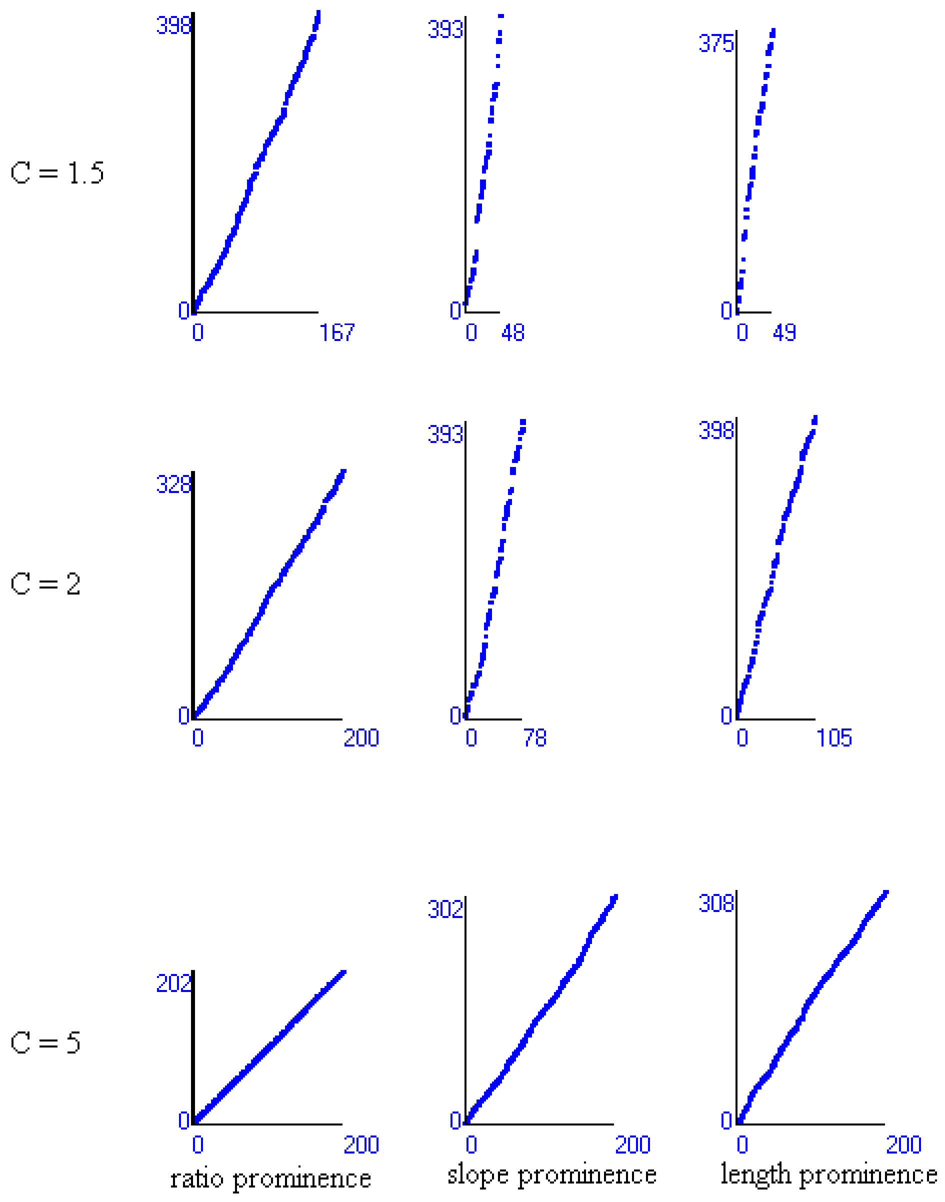


Figure 5.12: Retrieval of stock charts matching the pattern in Figure 5.8(d).

The horizontal axes show the similarity numbers assigned by the fast retrieval algorithm; the vertical axes are the exhaustive-search similarity numbers for the same points.

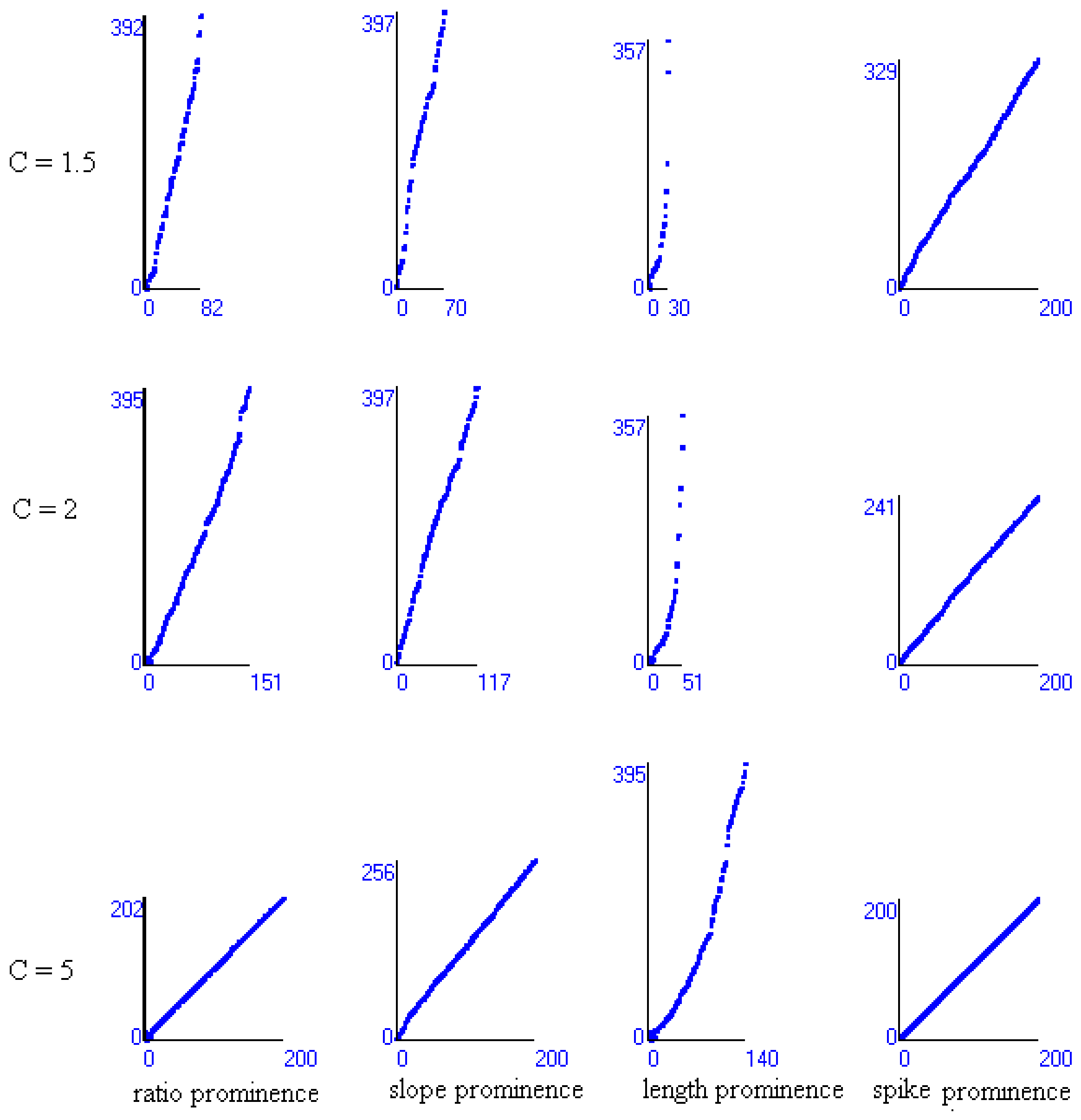


Figure 5.13: Retrieval of air and sea temperature combined segments matching the pattern in Figure 5.8(e).

The horizontal axes show the similarity numbers assigned by the fast retrieval algorithm; the vertical axes are the exhaustive-search similarity numbers for the same points.

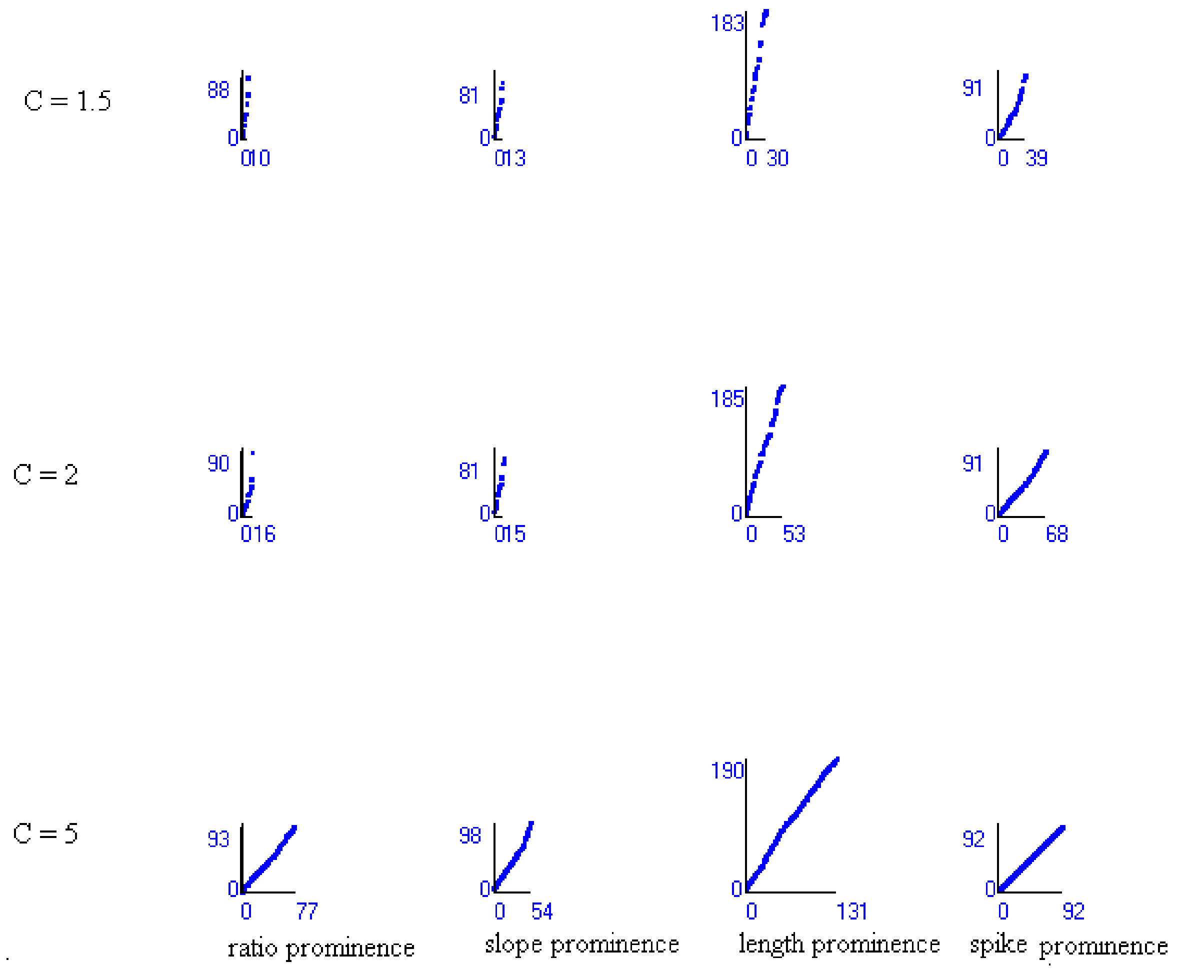


Figure 5.14: Retrieval of sea temperature segments matching the pattern in Figure 5.8(f).

The horizontal axes show the similarity numbers assigned by the fast retrieval algorithm; the vertical axes are the exhaustive-search similarity numbers for the same points.

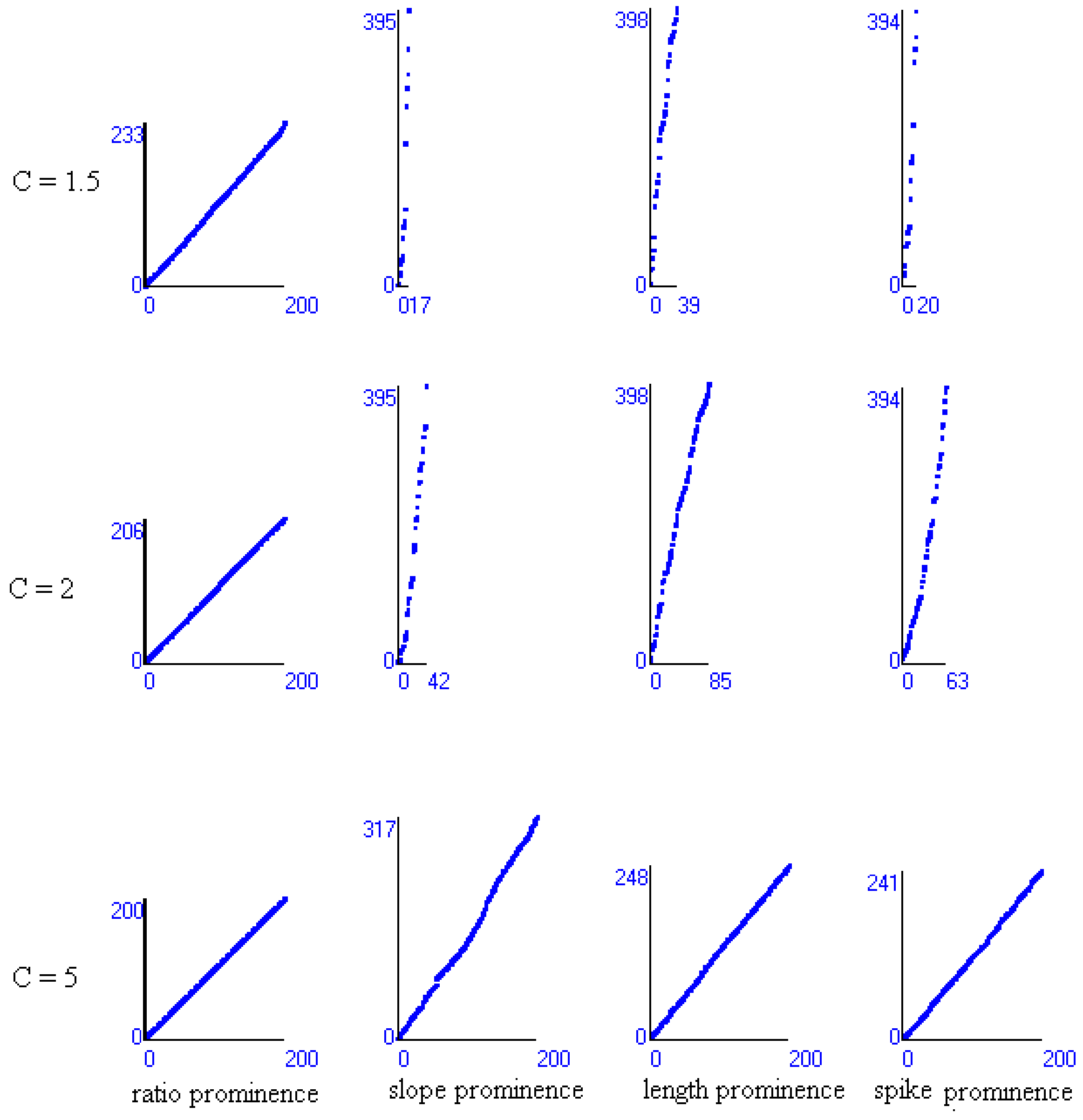


Figure 5.15: Retrieval of wind speeds segments matching the pattern in Figure 5.8(g).

The horizontal axes show the similarity numbers assigned by the fast retrieval algorithm; the vertical axes are the exhaustive-search similarity numbers for the same points.

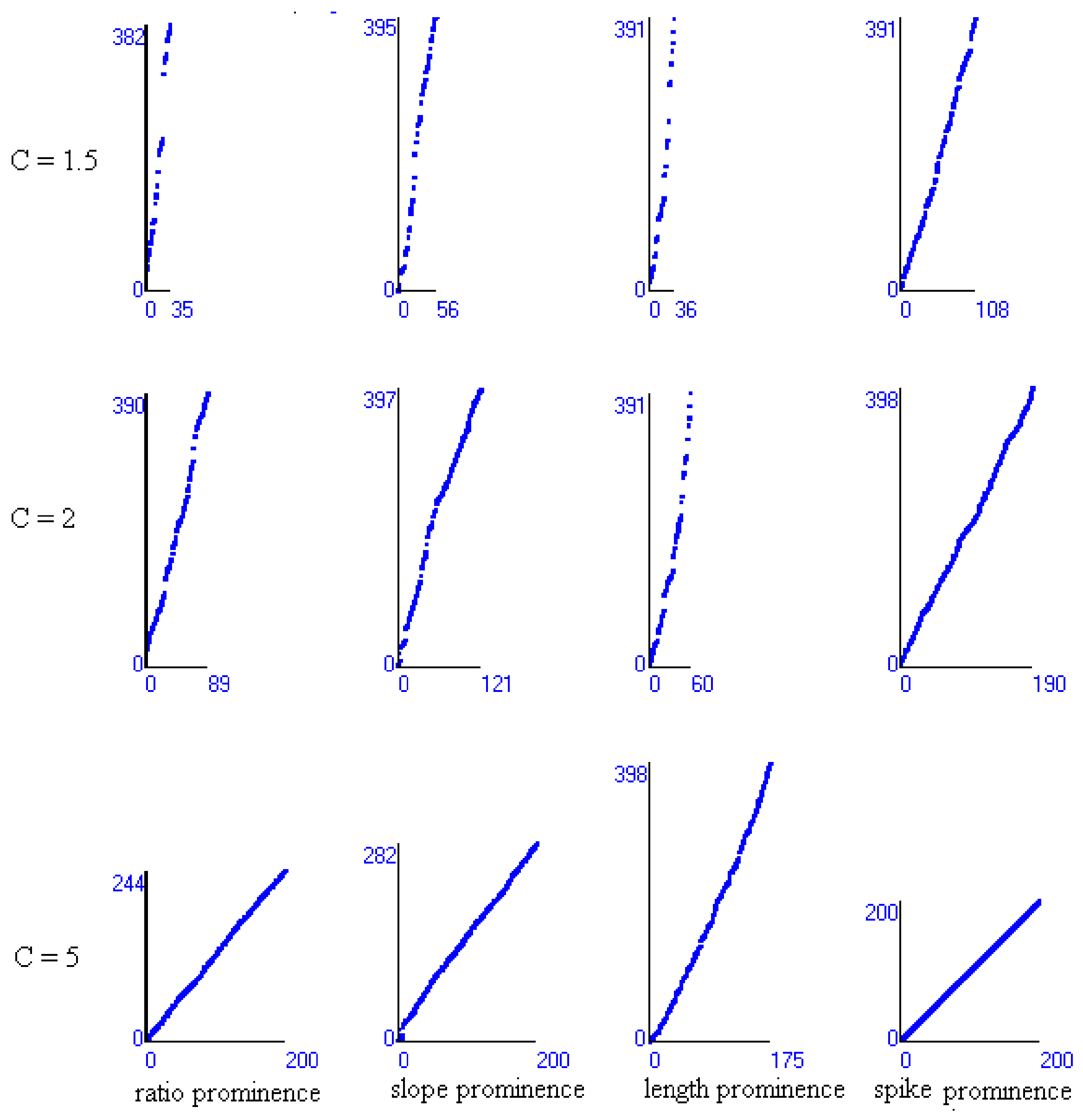


Figure 5.16: Retrieval of electroencephalogram segments matching the pattern in Figure 5.8(h).

The horizontal axes show the similarity numbers assigned by the fast retrieval algorithm; the vertical axes are the exhaustive-search similarity numbers for the same points.

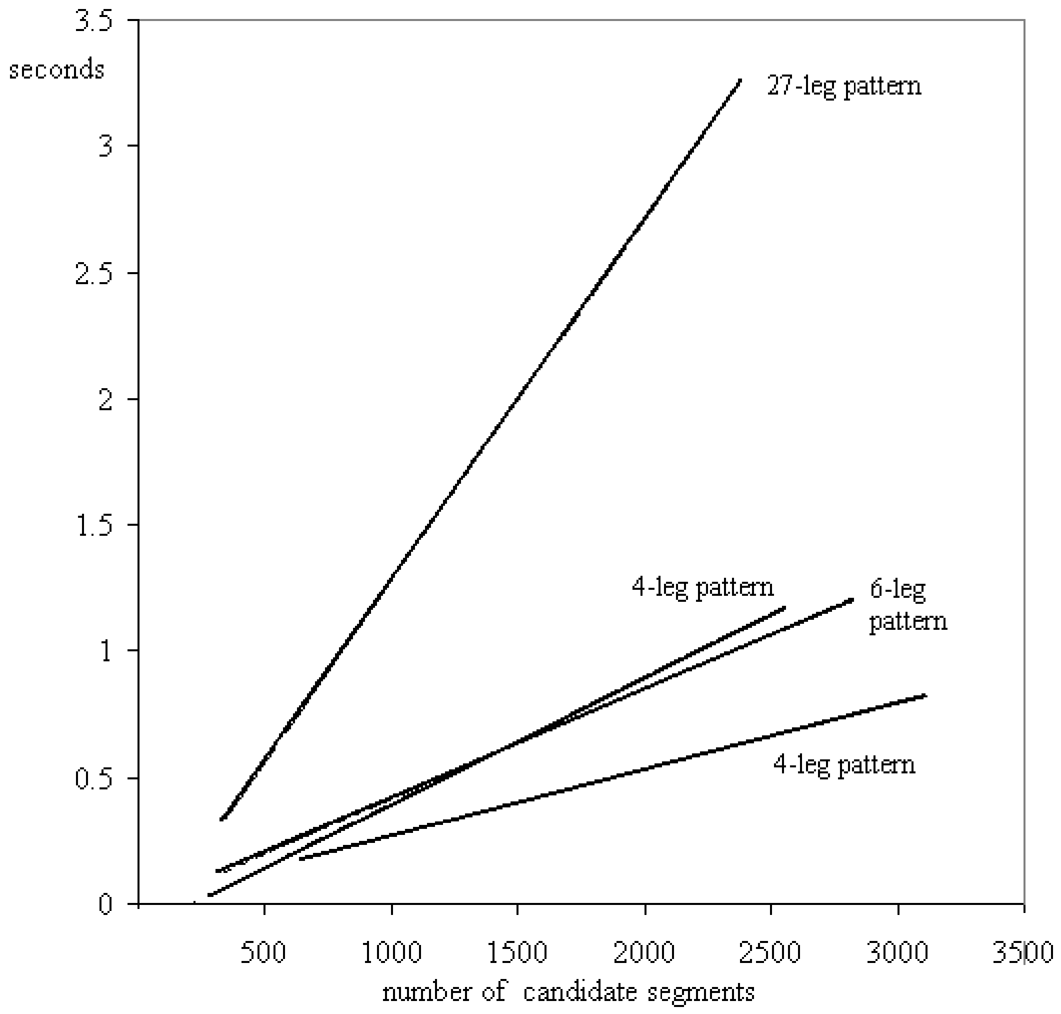


Figure 5.17: Retrieval time.

We show the dependency of the retrieval time on total number of candidate segments, identified by the first two steps of the retrieval algorithm. The time grows linearly with the number of candidate segments; it is also linear in the size of the pattern

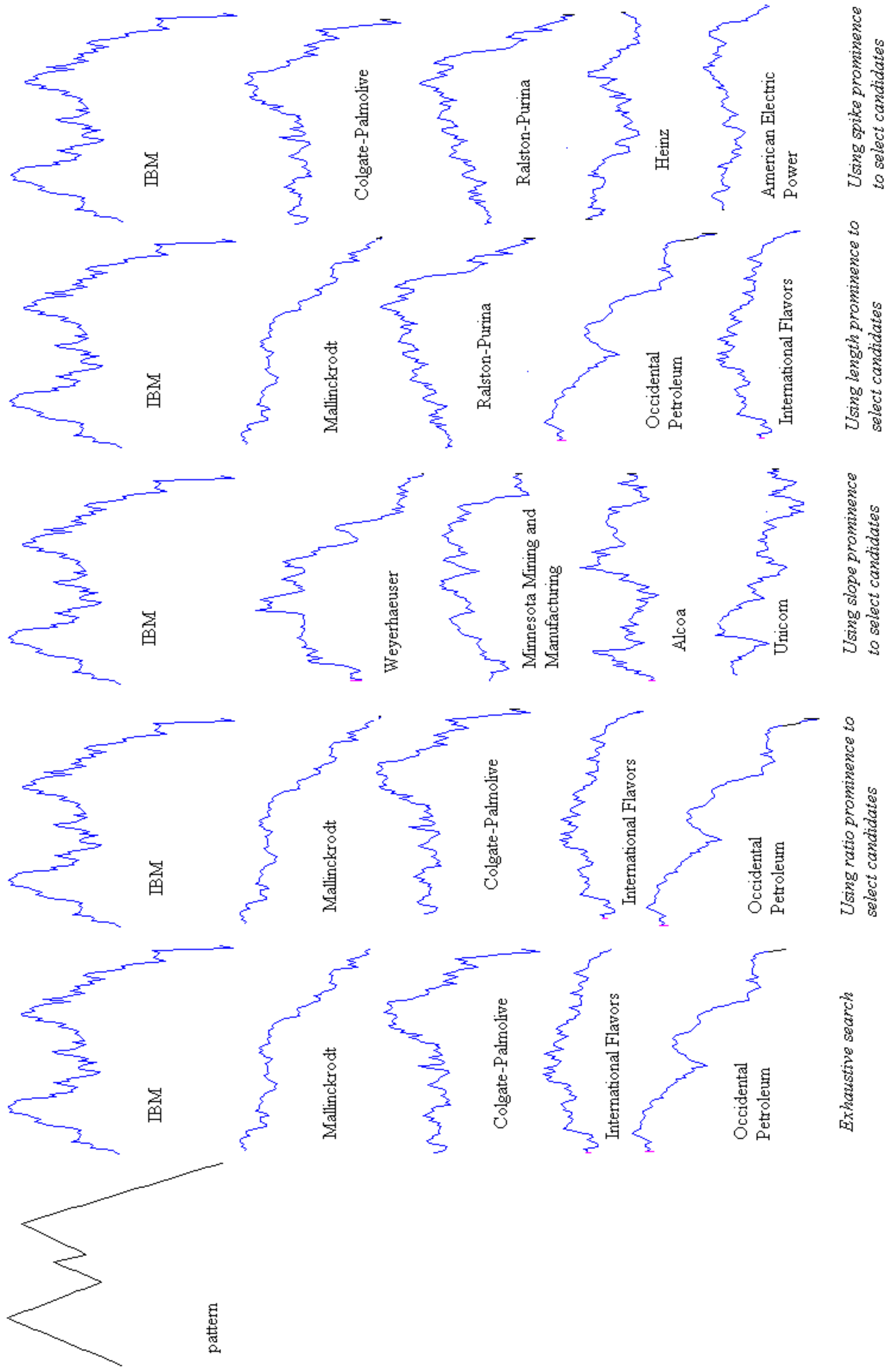


Figure 5.18: Examples of retrieved stock charts.

CHAPTER 6 - CONCLUDING REMARKS

The contributions of the described work include an algorithm for compressing time-series, and the use of this compression for indexing and retrieval of time-series. The compression technique is based on selection of important points. We gave a fast compression algorithm and showed that the compressed sequences closely approximated the original data, and that its quality gracefully degraded with the compression rate. We then defined a new similarity metric and showed that it was often more accurate than similarity based on mean, root mean square, and correlation coefficient.

These compression and similarity techniques enabled us to develop a novel algorithm for finding a given pattern in a database of time-series. The key idea is to index time-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 identifying patterns in stock prices, meteorological data, and electrocardiograms. The implemented algorithm found a given pattern in a database with 60,000 points in less than a second.

This work leaves many open problems, which include application of the developed technique to other time-series domains, investigation of its limitations, extending of this technique to finding patterns that are stretched over time, and applying it to identifying periodic patterns, such as weather cycles. Another open problem is to apply statistical and machine-learning techniques to tune the knobs of the described algorithms.

REFERENCES

[Aggarwal, 2000] C. C. Aggarwal and P. S. Yu. The IGrid Index: Reversing the Dimensionality Curse for Similarity Indexing in High Dimensional Space. In *Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining,* pages 119–129, 2000.

[Agrawal, 1996] R. Agrawal, M. Mehta, J. Shafer, and R. Srikant. The Quest Data Mining System. In *Proceedings of the Association for Computing Machinery Second International Conference on Knowledge Discovery and Data Mining,* pages 244–249, 1996.

[Beckmann, 1990] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. R-Tree, an Efficient and Robust Access Method for Points and Rectangles. *Association of Computing Machinery Special Interest Group on Management of Data Record*, 19(2), pages 322–331, 1990.

[Berchtold, 1998] Stefan Berchtold, Christian Bohm, and Hans-Peter Kriegel. The Pyramid-Tree: Breaking the Curse of Dimensionality. In *Proceedings of the Association for Computing Machinery International Conference on Management of Data*, pages 142–153, 1998.

[Bollobas, 1997] B. Bollobas, Gautam Das, Dimitrios Gunopulos, and H. Mannila. Time-Series Similarity Problems and Well-Separated Geometric Sets. In *Proceedings of the Association for Computing Machinery Thirteenth Annual Symposium on Computational Geometry,* pages 454–476, 1997.

[Bozkaya, 1997] Tolga Bozkaya, Nasser Yazdani, and Meral Ozsoyoglu. Matching and Indexing Sequences of Different Lengths. In *Proceedings of the Association for Computing Machinery Sixth International Conference on Information and Knowledge Management,* pages 128–135, 1997.

[Bozkaya, 1999] Tolga Bozkaya and Meral Ozsoyoglu. Indexing Large Metric Spaces for Similarity Search Queries. *Association for Computing Machinery Transactions on Database System,* pages 1–34, 1999.

[Brockwell, 1996] P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting.* Springer-Verlag, New York, NY, 1996.

[Caraca-Valente, 2000] J. P. Caraca-Valente and I. Lopez-Chavarrias. Discovering Similar Patterns in Time Series. In *Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining,* pages 497–505, 2000.

[Chi, 1995] E. H.-H. Chi, P. Barry, E. Shoop, J. V. Carlis, E. Retzel, and J. Riedl. Visualization of Biological Sequence Similarity Search Results. In *Proceedings of the IEEE Conference on Visualization*, pages 44–51, 1995.

[Cormen, 1998] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1998.

[Cortes, 2000] C. Cortes, K. Fisher, D. Pregibon, and A. Rogers. Hancock: A Language for Extracting Signatures from Data Streams. In *Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining,* pages 9–17, 2000.

[Das, 1997] Gautam Das, Dimitrios Gunopulos, and Heikki Mannila. Finding Similar Time Series. In *Proceedings of the First Conference on Principles of Knowledge Discovery and Data Mining,* pages 88–100, 1997.

[Das, 1998] Gautam Das, King-Ip Lin, Heikki Mannila, Gopal Renganathan, and Padhraic Smyth. Rule Discovery from Time-Series. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 16–22, 1998.

[Deng, 1998] K. Deng. OMEGA: On-Line Memory Based General Purpose System Classifier. *Technical Report* CMU-RI-TR-98-33, Robotics Institute, Carnegie Mellon University, 1998.

[Domingos, 2000] P. Domingos and G. Hulten. Mining High-Speed Data Streams. In *Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining,* pages 71–80, 2000.

[Duffy, 1994] F. H. Duffy, J. R. Hughes, F. Miranda, P. Bernad, and P. Cook. Status of Quantified EEG (qEEG) in Clinical Practice. *Clinical Electroencephalography,* 25, pages 6–22, 1994.

[Fountain, 2000] T. Fountain, T. Dietterich, and B. Sudyka. Mining IC Test Data to Optimize VLSI Testing. In *Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining*, pages 19–29, 2000.

[Franses, 1998] P. H. Franses. *Time Series Models for Business and Economic Forecasting*. University of Cambridge Press, Cambridge, United Kingdom, 1998.

[Galka , 2000] Andreas Galka. *Topics in Nonlinear Time Series Analysis with Implications for EEG Analysis*. World Scientific, Singapore, 2000.

[Gavrilov, 2000] M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani. Mining the Stock Market: Which Measure is Best? In *Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining,* pages 487–496, 2000.

[Ge, 2000] X. Ge and P. Smyth. Deformable Markov Model Templates for Pattern Matching. In *Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining,* pages 81–90, 2000.

[Geva, 1999] Amir B. Geva. Hierarchical-Fuzzy Clustering of Temporal-Patterns and its Application for Time-Series Prediction. *Pattern Recognition Letters,* 20, pages 1519–1532, 1999.

[Goldin, 1995] D. Q. Goldin and P. 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.

[Grenander, 1996] U. Grenander. *Elements of Pattern Theory*. Johns Hopkins University Press, Baltimore, MD, 1996.

[Gunopulos, 2000] Dimitrios Gunopulos and Gautam Das. Time Series Similarity Measures. In *Tutorial Notes of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining,* pages 243–307, 2000.

[Guralnik, 1998] V. Guralnik, D. Wijesekera, and J. Srivastava. Pattern Directed Mining of Sequence Data. In *Proceedings of the Fourth International Conference on Knowledge Discovery in Databases*, pages 51–57, 1998.

[Han, 2000] J. Han, J. Pei, B. Mortazavi-Asi, Q. Chen, U. Dayal, and M.-C. Hsu. FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining. In *Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining,* pages 355–359, 2000.

[Han, 1998] J. Han, W. Gong, and Y. Yin. Mining Segment-Wise Periodic Patterns in Time-Related Databases. In *Proceedings of the Association for Computing Machinery Fourth International Conference on Knowledge Discovery and Data Mining,* pages 214–218, 1998.

[Haslett, 1989] J. Haslett and A. E. Raftery. Space-Time Modeling with Long-Memory Dependence: Assessing Ireland's Wind Power Resource. *Applied Statistics,* 38, pages 1–50, 1989.

[Ikeda , 1999] K. Ikeda, B. Vaughn, and S. Quint. Wavelet Decomposition of Heart Period Data. In *Proceedings of the IEEE First Joint BMES/EMBS Conference*, pages 3–11, 1999.

[Kamel, 1993] Ibrahim Kamel and Christos Faloutsos. Hilbert R-Tree: An Improved R-Tree Using Fractals. *Technical Research Report*, Institute for Systems Research, University of Maryland, College Park, MD, 1993.

[Kantz, 1997] H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis.* Cambridge University Press, Cambridge, United Kingdom, 1997.

[Keogh, 1997] Eamonn Keogh. Fast Similarity Search in the Presence of Longitudinal Scaling in Time Series Databases. In *Proceedings of the IEEE Ninth International Conference on Tools with Artificial Intelligence*, pages 578–584, 1997.

[Keogh, 1998] Eamonn 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 Association for Computing Machinery Fourth International Conference on Knowledge Discovery and Data Mining*, pages 239–243, 1998.

[Keogh, 2000] Eamonn J. Keogh and Michael J. Pazzani. Scaling up Dynamic Time Warping for Data Mining Applications. In *Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining*, pages 285–289, 2000.

[Lin, 1998] L. Lin and T. Risch. Querying Continuous Time Sequences. In *Proceedings of the Twenty-Fourth Very Large Database Conference*, pages 170–181, 1998.

[Lee, 2000] S.-L. Lee, S.-J. Chun, D.-H. Kim, J.-H. Lee, and C.-W. Chung. Similarity Search for Multidimensional Data Sequences. In *Proceedings of the IEEE Sixteenth International Conference on Data Engineering*, pages 599–608, 2000.

[Lu, 1997] H. Lu, J. Han, and L. Feng. Stock Movement Prediction and *N*-Dimensional Inter-Transaction Association Rules. In *Proceedings of the Association for Computing Machinery SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 12:1–7, 1998.

[Maurer, 1991] K. Maurer and T. Dierks. *Atlas of Brain Mapping.* Springer-Verlag, Berlin, Germany, 1991.

[Murphy, 1996] J. J. Murphy. *Visual Investor.* Wiley, New York, NY, 1996.

[Niedermeyer,1993] E. Niedermeyer and F. L. Da Silva. *Electroencephalography Basic Principles, Clinical Applications, and Related Fields*, Third Edition. Williams & Wilkins, Baltimore, MD, 1993.

[Park, 2000] S. Park, W. W. Chu, J. Yoon, and C. Hsu. Efficient Searches for Similar Subsequences of Different Lengths in Sequence Databases. In *Proceedings of the IEEE Sixteenth International Conference on Data Engineering*, pages 23–32, 2000.

[Perng, 2000] C.-S. Perng, H. Wang, S. R. Zhang, and D. S. Parker. Landmarks: A New Model for Similarity-Based Pattern Querying in Time Series Databases. In *Proceedings of the IEEE Sixteenth International Conference on Data Engineering*, pages 33–42, 2000.

[Plane, 1986] D. R. Plane and E. B. Opperman. *Business and Economic Statistics*, Third Edition. Business Publications, Plano, TX, 1986.

[Policker, 2000] S. Policker and A. B. Geva. Nonstationary Time Series Analysis by Temporal Clustering. *IEEE Transactions on Systems, Man and Cybernetics,* 30(2), pages 339–343, 2000.

[Popivanov, 1998] D. Popivanov, A. Mineva, and J. Dushanova. Tracking EEG Signal Dynamics During Mental Tasks. *IEEE Engineering in Medicine and Biology,* 17(2), pages 89–95, 1998.

[Priestley, 1988] M. B. Priestley. *Nonlinear and Nonstationary Time Series Analysis.* Academic Press, London, United Kingdom,1988.

[Sahoo, 1988] P. K. Sahoo, S. Soltani, and A. K. C. Wong. Survey of Thresholding Techniques. *Computer Vision, Graphics and Image Processing*, 41, pages 233–260, 1988.

[Sheikholeslami, 1998] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. In *Proceedings of the Twenty-Fourth Very Large Database Conference*, pages 428–439, 1998.

[Singh, 1998] S. Singh and P. McAtackney. Dynamic Time-Series Forecasting Using Local Approximation. In *Proceedings of the IEEE Tenth International Conference on Tools with Artificial Intelligence*, pages 392–399, 1998.

[Sonka, 1999] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision,* Second Edition. PWS Publishing, Pacific Grove, CA, 1999.

[Spiegel, 1996] M. R. Spiegel. *Theory and Problems of Statistics,* Second Edition. McGraw-Hill, New York, NY, 1996.

[Stoffer, 1999] D. S. Stoffer. Detecting Common Signals in Multiple Time Series Using the Spectral Envelope. *Journal of the American Statistical Association,* 94(448), pages 1341–1356, 1999.

[Tsai, 1999] C.-C. Tsai and S.-J. Wu. A Study for Second-Order Modeling of Fuzzy Time Series. In *Proceedings of the IEEE International Fuzzy Systems Conference,* pages 719–725, 1999.

[Yaffee, 2000] R. A. Yaffee and M. McGee. *Introduction to Time Series Analysis and Forecasting*. Academic Press, San Diego, CA, 2000.

[Yi, 2000] B.-K. Yi, N. D. Sidiropoulos, T. Johnson, A. Biliris, H. V. Jagadish and C. Faloutsos. Online Data Mining for Co-Evolving Time Sequences. In *Proceedings of the IEEE Sixteenth International Conference on Data Engineering*, pages 13–22, 2000.