Compression of Time Series by Extracting Major Extrema

Eugene Fink **Harith Suman Gandhi**

e.fink@cs.cmu.edu, www.cs.cmu.edu/~eugene

Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, United States

ABSTRACT

We formalize the notion of *important extrema* of a time series, that is, its major minima and maxima; analyze basic mathematical properties of important extrema; and apply these results to the problem of time-series compression. First, we define numeric importance levels of extrema in a series, and present algorithms for identifying major extrema and computing their importances. Then, we give a procedure for fast lossy compression of a time series at a given rate, by extracting its most important minima and maxima, and discarding the other points.

Keywords: Time series, lossy compression, major minima and maxima.

1. INTRODUCTION

A *time series* is a sequence of values measured at equal time intervals, such as stock prices and electrocardiograms; for example, the series in Figure 1 includes values 1, 3, 3, 5, and so on. We present algorithms for compressing a time series by extracting its major *extrema*, that is, minima and maxima. For example, we can compress the series in Figure 1 by extracting the circles extrema, along with the two endpoints, and discarding the other points. The described results are a continuation of the work by Pratt and Fink [2002] on the compression of time series and indexing a database of compressed series.

Researchers have studied series compression in the context of many different applications [Box et al., 2008; Montgomery et al., 2008; Cryer and Chan, 2009], from analysis of financial data [Fu et al., 2008; Dorr and Denton, 2009] to distributed monitoring systems [Di et al., 2007] and manufacturing applications [Eruhimov et al., 2006]. In particular, they have considered various feature sets for compressing series and measuring similarity between them. They have extensively studied Fourier transforms, which allow fast compression [Sheikholeslami et al., 1998; Singh and McAtackney, 1998; Stoffer, 1999; Yi et al., 2000], but have discovered that this technique has several disadvantages. It smoothes local minima and maxima, which may lead to a loss of

important information, and it does not work well for erratic series [Ikeda et al., 1999]. Chan and his colleagues applied Haar wavelet transforms [Burrus et al., 1997; Graps, 1995] to time series and pointed out several advantages of this technique over Fourier transforms [Chan and Fu, 1999; Chan et al., 2003], whereas Eurhimov et al. [2006] used Chebyshev polynomial expansion. Kriegel et al. [2008] developed a more general technique, which involved decomposing time series into patterns from a given database.

Guralnik and Srivastava [1999] considered the problem of detecting changes in the trend of a data stream, and developed a technique for finding "change points" in a series. Last et al. [2001] 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. Pratt and Fink presented a procedure for finding major extrema in a series, which are a special case of change points [Pratt, 2001; Pratt and Fink, 2002].

Researchers have also studies the use of small alphabets for series compression, and applied string matching [Gusfield, 1997] to the related pattern search [Agrawal et al., 1995; André-Jönsson and Badal, 1997; Lam and Wong, 1998; Qu et al., 1998; Huang and Yu, 1999; Park et al., 1999]. For example, Guralnik et al. [1998] compressed stock prices using a nine-letter alphabet. Singh and McAtackney [1998] represented stock prices, particle dynamics, and stellar light intensity by a three-letter alphabet. Lin and Risch [1998] used a two-letter alphabet to encode major spikes in a series. Das et al. [1998] utilized an alphabet of primitive shapes. These techniques give a high compression rate, but their descriptive power is limited, which makes them inapplicable in many domains.

Perng et al. [2000] investigated a compression technique based on extraction of "landmark points," which included local minima and maxima. Keogh and Pazzani [1998] used the endpoints of best-fit line segments to compress a series. Keogh et al. [2001] reviewed and compared the compression techniques based on approximation of a time series by a sequence of straight segments.

We present an alternative compression technique based on selection of major extrema and discarding the other points. First, we define four types of extrema and describe an algorithm that finds all extrema (Section 2). Then, we introduce the notion of *important extrema* and give a procedure for identifying them (Section 3). We also define the *importance levels* of extrema and present a technique for computing them (Section 4). Finally, we give an algorithm that compresses a series at a given rate (Section 5).

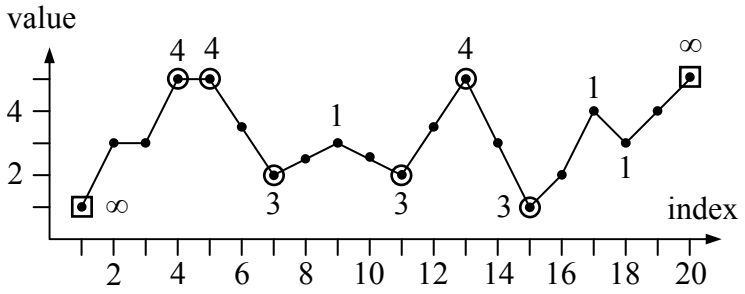


Figure 1: Example of a time series and its extrema. We show the importance of each extremum and mark the extrema with importances greater than 1.

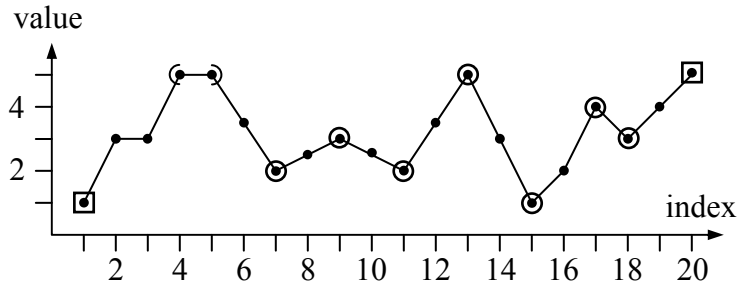
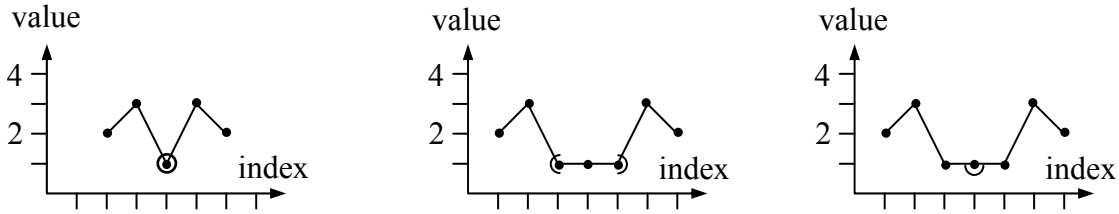


Figure 2: Compression by extracting all extrema. We show strict extrema by circles, left and right extrema by half-circles, and endpoints by squares.



(a) Strict minimum. (b) Left and right minima. (c) Flat minimum.

Figure 3: Four types of minima.

2. EXTREMA

We begin with a simple compression based on the extraction of all extrema (Figure 2). We distinguish four extrema types, called strict, left, right, and flat (Figure 3). We give a definition of strict, left, right, and flat minima; the definition for maxima is symmetric.

Definition 1 (Minima): Suppose that $a_1, ..., a_n$ is a time series, and a_i is its point such that $1 < i < n$.

- a_i is a *strict minimum* if $a_i < a_{i-1}$ and $a_i < a_{i+1}$.
- a_i is a *left minimum* if $a_i < a_{i-1}$ and there is an index $right > i$ such that $a_i = a_{i+1} = ... = a_{right} < a_{right+1}$.
- a_i is a *right minimum* if $a_i < a_{i+1}$ and there is an index $left < i$ such that $a_{left-1} > a_{left} = ... = a_{i-1} = a_i$.
- a_i is a *flat minimum* if there are indices $left < i$ and $right > i$ such that $a_{left-1} > a_{left} = ... = a_i = ... = a_{right} < a_{right+1}$.

In Figure 4, we give a procedure that identifies all extrema and determines their types, which is an extended version of the compression algorithm by Pratt and Fink [2002]; it takes linear time and constant memory. Note that it can process new points as they arrive, one by one, without storing the entire series in memory or on disk; for example, it can process a live electrocardiogram without waiting until the end of the data collection.

We can compress a series by extracting its strict, left, and right extrema, along with its two endpoints, and discarding the flat extrema and nonextremal points (Figure 2). We state an important property of this compression, called *monotonicity*, which can be used for indexing and fast retrieval of series, and also for evaluation of similarity between compressed series [Gandhi, 2004].

Definition 2 (Monotonic compression): Suppose that $a_1, ..., a_n$ is a time series, and $a_{i_1}, ..., a_{i_s}$ is its compressed version. The compression is *monotonic* if, for every consecutive indices i_c and i_{c+1} in the compressed series and every index i in the original series, if $i_c < i < i_{c+1}$, then either $a_{i_c} \le a_i \le a_{i_{c+1}}$ or $a_{i_{c+1}} \le a_i \le a_{i_c}$.

If we compress a series by selecting all strict, left, and right extrema, along with the two endpoints, the resulting compression is monotonic.

ALL-EXTREMA — Find all extrema
Input: Series $a_1, ..., a_n$
Output: Values, indices, and types of all extrema

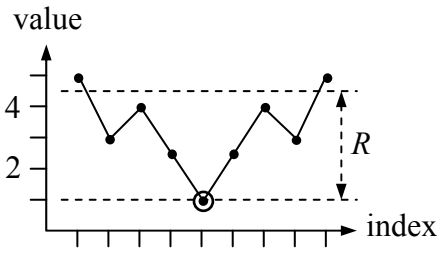
$i = 2$
while $i < n$ and $a_i = a_1$ **do** $i = i + 1$
if $i < n$ and $a_i < a_1$ **then** $i =$ FIND-MIN(i)
while $i < n$ **do**
  $i =$ FIND-MAX(i); $i =$ FIND-MIN(i)

FIND-MIN(i) — Find the first minimum after the ith point
left = i
while $i < n$ and $a_i \geq a_{i+1}$ **do**
  $i = i + 1$; **if** $a_{left} > a_i$ **then** *left* = i
if $i < n$ **then** OUTPUT-EXT(*left*, i, "min");
return $i + 1$

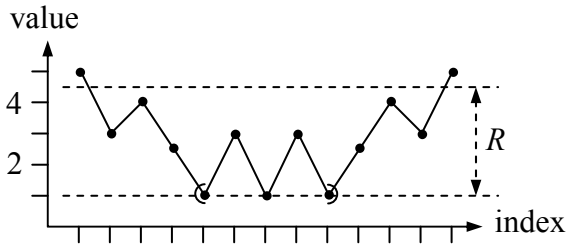
FIND-MAX(i) — Find the first maximum after the ith point
left = i
while $i < n$ and $a_i \leq a_{i+1}$ **do**
  $i = i + 1$; **if** $a_{left} < a_i$ **then** *left* = i
if $i < n$ **then** OUTPUT-EXT(*left*, i, "max")
return $i + 1$

OUTPUT-EXT(*left*, *right*, *min-or-max*) — Output extrema
if *left* = *right*
  then output(a_{right}, *right*, *min-or-max*, "strict")
  else output(a_{left}, *left*, *min-or-max*, "left")
    for *flat* = *left* + 1 **to** *right* − 1 **do output**(a_{flat}, *flat*, *min-or-max*, "flat")
    output(a_{right}, *right*, *min-or-max*, "right")

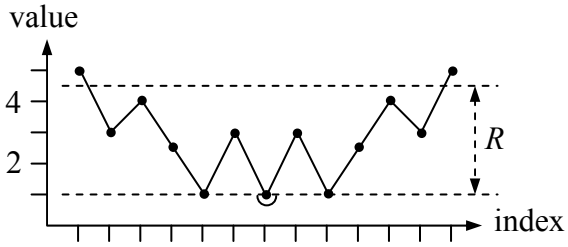
Figure 4: Identifying all extrema. We process a series $a_1, ..., a_n$ and use a global variable n to represent its size.



(a) Strict minimum.



(b) Left and right minima.



(c) Flat minimum.

Figure 5: Four types of important minima.

3. IMPORTANT EXTREMA

We can achieve a higher compression rate by selecting only certain important extrema. We first introduce the notion of *distance* between numeric values, and then use it to define these important extrema.

> **Definition 3 (Distance):** A *distance* between real values is a two-argument function, denoted $dist(a, b)$, that satisfies the following conditions:
>
> - For every value a, $dist(a, a) = 0$.
> - For every two values a and b, $dist(a, b) = dist(b, a)$.
> - For every three values a, b, and c, if $a \le b \le c$, then $dist(a, b) \le dist(a, c)$ and $dist(b, c) \le dist(a, c)$.

For example, the functions $|a - b|$, $\frac{|a-b|}{|a|+|b|}$, and $\frac{|a-b|}{\max(|a|,|b|)}$ are three different distance functions.

Although we use the term "distance," we do *not* assume that it satisfies the triangle inequality; thus, $dist(a, c)$ may be greater than $dist(a, b) + dist(b, c)$. Also, we do *not*

assume that the distance between two distinct values is nonzero; thus, $dist(a, b)$ may be zero for distinct a and b.

We next describe a method for combining distance functions into a more complex distance. We present it as a lemma, which readily follows from the definition.

Lemma 1 (Distance composition): Suppose that $f(d_1, ..., d_q)$ is a real-valued function on nonnegative real arguments such that $f(0, ..., 0) = 0$ and f is monotonically increasing on each of its arguments, and suppose further that $dist_1$, ..., $dist_q$ are distance functions; then, $f(dist_1(a, b), ..., dist_q(a, b))$ is also a distance function.

In particular, a weighted sum of distance functions is a distance. That is, if $dist_1$, ..., $dist_q$ are distance functions and w_1, ..., w_q are nonnegative reals, then $w_1 \cdot dist_1(a, b) + ... + w_q \cdot dist_q(a, b)$ is also a distance function.

We control the compression procedure by selecting a specific distance function, along with a positive parameter R that determines the compression rate; an increase of R leads to selection of fewer important extrema. We now give a definition of important minima, illustrated in Figure 5; the definition of important maxima is symmetric.

Definition 4 (Important minimum): For a given distance function *dist* and positive value R, a point a_i of a series a_1, ..., a_n is an *important minimum* if there are indices il and ir, where $il < i < ir$, such that

- a_i is a minimum among a_{il}, ..., a_{ir}, and
- $dist(a_i, a_{il}) \geq R$ and $dist(a_i, a_{ir}) \geq R$.

Intuitively, a_i is an important minimum if it is a minimal value of some segment a_{il}, ..., a_{ir}, and the endpoint values of this segment are much larger than a_i. We next define strict, left, right, and flat important minima.

Definition 5 (Strict important minimum): A point a_i of a series a_1, ..., a_n is a *strict important minimum* if there are indices il and ir, where $il < i < ir$, such that

- a_i is strictly smaller than a_{il}, ..., a_{i-1} and a_{i+1}, ..., a_{ir}, and
- $dist(a_i, a_{il}) \geq R$ and $dist(a_i, a_{ir}) \geq R$.

We give an example of a strict important minimum in Figure 5(a); intuitively, a point satisfies this definition if it is a *strict* minimum of some segment, and the endpoint values of the segment are much larger.

Definition 6 (Left important minimum): A point a_i of a series a_1, ..., a_n is a *left important minimum* if it is *not* a strict important minimum, and there are indices *il* and *ir*, where $il < i < ir$, such that

- a_i is strictly smaller than a_{il}, ..., a_{i-1},
- a_i is no larger than a_{i+1}, ..., a_{ir}, and
- $dist(a_i, a_{il}) \geq R$ and $dist(a_i, a_{ir}) \geq R$.

The definition of a right important minimum is symmetric; we show examples in Figure 5(b).

Definition 7 (Flat important minimum): A point is a *flat important minimum* if it is an important minimum, but *not* a strict, left, or right important minimum.

We illustrate this definition in Figure 5(c); intuitively, a flat important minimum is one of several equally important minima in some segment a_{il}, ..., a_{ir}.

In Figure 6, we give a procedure that identifies the important extrema and determines their types. We can compress a series by selecting its strict, left, and right important extrema, along with its two endpoints, and discarding the other points. In Figure 7, we show the selected extrema for the distance $|a - b|$ with $R = 3$. Note that this compression may not be monotonic; for example, the points a_7 and a_{11} in Figure 7 are consecutive important extrema, but the value of a_9 is *not* between the values of a_7 and a_{11}.

We can achieve a higher compression rate by selecting only strict and left extrema, or only strict and right extrema. The resulting compression is monotonic, but it may not preserve information about flat and near-flat regions, such as the segment a_7, ..., a_{11} in Figure 7.

Lemma 2 (Monotonicity): If we compress a series by selecting all strict important extrema, all left (or right) important extrema, and the endpoints, then the resulting compression is monotonic.

We can recompress an already compressed series with a larger R using the same distance function, which produces the same result as the compression of the original series with the same larger R. We use the procedure in Figure 6 with minor modifications for this recompression. Specifically, we input the compressed series just like the original series. For each selected important extremum, the modified procedure outputs its index in the original series instead of its index in the input compressed series.

IMPORTANT-EXTREMA — Find the important extrema
Input: Series $a_1, ..., a_n$, distance function *dist*, and R value
Output: Values, indices, and types of the important extrema

i = FIND-FIRST
if $i < n$ and $a_i < a_1$ **then** i = FIND-MIN(i)
while $i < n$ **do**
  i = FIND-MAX(i); i = FIND-MIN(i)

FIND-FIRST — Find the first important extremum
$i = 1$; $leftMin = 1$; $rightMin = 1$; $leftMax = 1$; $rightMax = 1$
while $i < n$ and $dist(a_{i+1}, a_{leftMax}) < R$ and $dist(a_{i+1}, a_{leftMin}) < R$ **do**
  $i = i + 1$
  if $a_{leftMin} > a_i$ **then** $leftMin = i$
  if $a_{rightMin} \geq a_i$ **then** $rightMin = i$
  if $a_{leftMax} < a_i$ **then** $leftMax = i$
  if $a_{rightMax} \leq a_i$ **then** $rightMax = i$
$i = i + 1$
if $i < n$ and $a_i > a_1$ **then** OUTPUT-EXT($leftMin$, $rightMin$, "min")
if $i < n$ and $a_i < a_1$ **then** OUTPUT-EXT($leftMax$, $rightMax$, "max")
return i

FIND-MIN(i) — Find the first important minimum after the ith point
$left = i$; $right = i$
while $i < n$ and ($a_{i+1} < a_{left}$ or $dist(a_{i+1}, a_{left}) < R$) **do**
  $i = i + 1$
  if $a_{left} > a_i$ **then** $left = i$
  if $a_{right} \geq a_i$ **then** $right = i$
OUTPUT-EXT($left$, $right$, "min")
return $i + 1$

FIND-MAX(i) — Find the first important maximum after the ith point
$left = i$; $right = i$
while $i < n$ and ($a_{i+1} > a_{left}$ or $dist(a_{i+1}, a_{left}) < R$) **do**
  $i = i + 1$
  if $a_{left} < a_i$ **then** $left = i$
  if $a_{right} \leq a_i$ **then** $right = i$
OUTPUT-EXT($left$, $right$, "max")
return $i + 1$

OUTPUT-EXT($left$, $right$, *min-or-max*) — Output extrema
if $left = right$
  then output(a_{right}, $right$, *min-or-max*, "strict")
  else output(a_{left}, $left$, *min-or-max*, "left")
    for $flat = left + 1$ **to** $right - 1$ **do**
      if $a_{flat} = a_{left}$ **then output**(a_{flat}, $flat$, *min-or-max*, "flat")
    output(a_{right}, $right$, *min-or-max*, "right")

Figure 6: Selecting the important extrema.

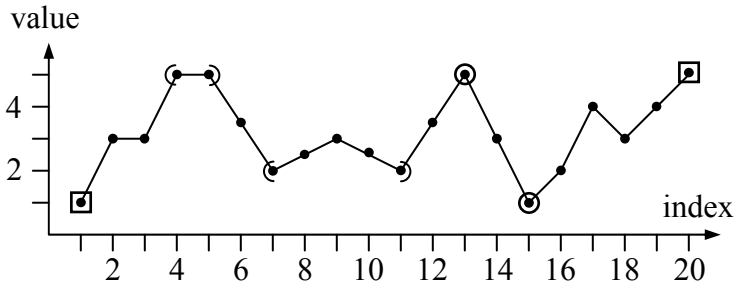
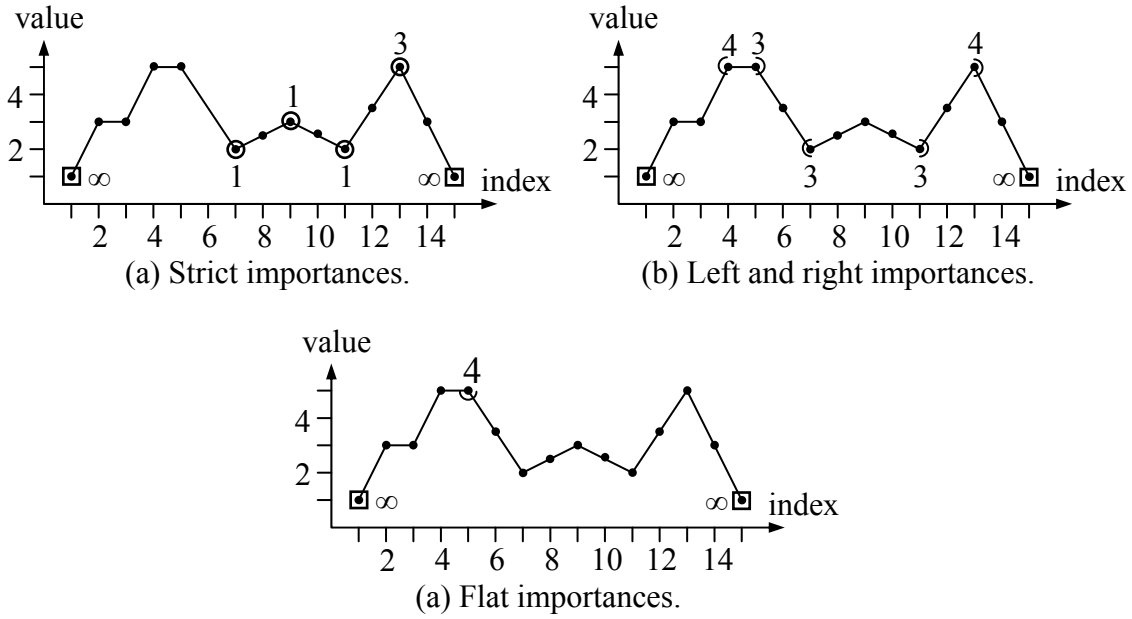


Figure 7: Important extrema for the distance $|a - b|$ with $R = 3$. We show the strict extrema by circles, left and right extrema by half-circles, and endpoints by squares.



(a) Strict importances. (b) Left and right importances.

(a) Flat importances.

Figure 8: Importances of extrema for the distance $|a - b|$. We show the strict, left, right, and flat importances.

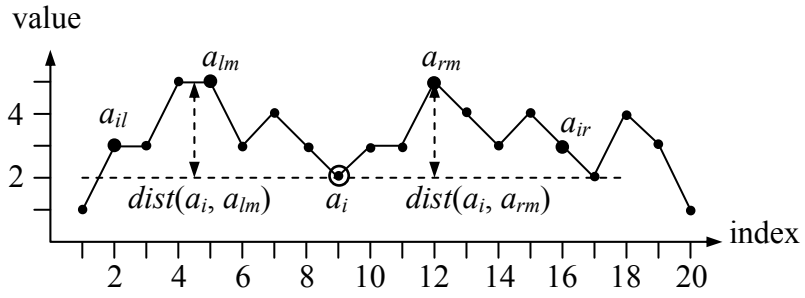


Figure 9: Computation of the strict importance of a minimum. To determine the importance of a_i, we identify the maximal segments a_{il}, ..., a_{i-1} and a_{i+1}, ..., a_{ir} whose values are strictly greater than a_i, find the maximal values a_{lm} and a_{rm} in these segments, and compute the importance of a_i as the smaller of $dist(a_i, a_{lm})$ and $dist(a_i, a_{rm})$.

ALL-IMPORTANCES — Find the importances of all minima
Input: Series $a_1, ..., a_n$ and distance function *dist*
Output: Values, indices, and importances of all minima

initialize an empty linked list L of indices
initialize an empty stack S_1 of indices
initialize empty stacks S_2, S_3, and S_4 of values
$i = 1$; *left* = 1; *max-value* = a_1
while $i < n$ **do**
  while $i < n$ and $a_i \geq a_{i+1}$ **do**
    $i = i + 1$; **if** $a_{left} > a_i$ **then** *left* = i
  if *left* > 1 **then** PUSH-MIN(*left*, *max-value*)
  for *flat* = *left* +1 **to** $i - 1$ **do** PUSH-MIN(*flat*, a_{left})
  if $i >$ *left* and $i < n$ **then** PUSH-MIN(i, a_{left})
  $i = i + 1$
  while $i < n$ and $a_i \leq a_{i+1}$ **do** $i = i + 1$
  max-value = a_i; $i = i + 1$; *left* = i
  if S_1 is not empty **then** TOP(S_3) = *max-value*
while S_1 is not empty **do** *max-value* = POP-MIN(*max-value*)
for every i in the list L **do output**(a_i, i, *strict-imp*$_i$, *left-imp*$_i$, *right-imp*$_i$, *flat-imp*$_i$)

PUSH-MIN(i, *max-value*) — Push the ith point onto the stack
while S_1 is not empty and $a_i < a_{\text{TOP}(S_1)}$ **do** *max-value* = POP-MIN(*max-value*)
PUSH(S_1, i); PUSH(S_2, *max-value*); PUSH(S_3, a_i)
if S_4 is not empty **then** PUSH(S_4, max(TOP(S_4), *max-value*)) **else** PUSH(S_4, *max-value*)
add i to the end of the list L

POP-MIN(*max-value*) — Pop a point and set its importances
i = POP(S_1); *left-max* = POP(S_2); *right-max* = POP(S_3); *other-max* = POP(S_4)
if $a_i <$ *left-max* and $a_i <$ *right-max*
  then *strict-imp*$_i$ = min(*dist*(a_i, *left-max*), *dist*(a_i, *right-max*))
if $a_i <$ *left-max* and *right-max* < *max-value*
  then *left-imp*$_i$ = min(*dist*(a_i, *left-max*), *dist*(a_i, *max-value*))
if $a_i <$ *right-max* and *left-max* < *other-max*
  then *right-imp*$_i$ = min(*dist*(a_i, *other-max*), *dist*(a_i, *right-max*))
if *left-max* < *other-max* and *right-max* < *max-value*
  then *flat-imp*$_i$ = min(*dist*(a_i, *other-max*), *dist*(a_i, *max-value*))
if *max-value* < *left-max* **then** *max-value* = *left-max*
if S_1 is not empty and $a_{\text{TOP}(S_1)} < a_i$ **then** TOP(S_3) = *max-value*
return *max-value*

Figure 10: Importance calculation. The procedure outputs all minima in order; for each minimum, the output includes its value and index, as well as its strict, left, right, and flat importances.

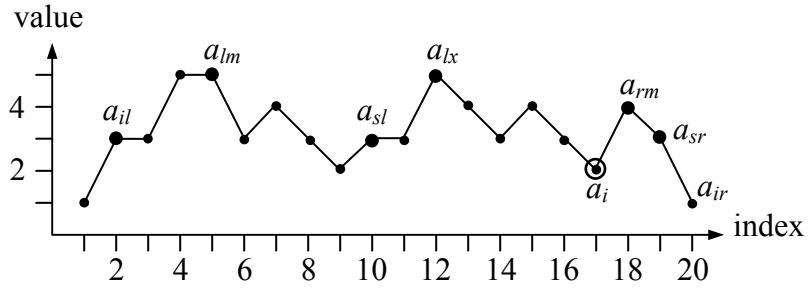


Figure 11: Illustration of the local maxima surrounding a minimum a_i. These maxima are used in computing the importances of a_i. The notation is the same as in Figure 9. The point a_{lx} is the maximum of the segment a_{sl}, ..., a_{i-1}, whose values are strictly greater than a_i; the algorithm in Figure 10 stores this maximum in stack S_2. Similarly, a_{rm} is the maximum of the segment a_{i+1}, ..., a_{sr}, whose values are strictly greater than a_i; the algorithm stores it in stack S_3. Finally, a_{lm} is the maximum of the segment a_{il}, ..., a_{i-1}, whose values are no smaller than a_i; the algorithm stores it in stack S_4.

4. IMPORTANCE LEVELS

We define numeric importances of extrema and give an algorithm for computing them.

> **Definition 8 (Importance):** If a point is a strict (left, right, flat) extremum for some value of R, then its strict (left, right, flat) importance is the maximal value of R for which it is a strict (left, right, flat) extremum.

In other words, the strict (left, right, flat) importance of an extremum is I if it is a strict (left, right, flat) extremum for $R = I$, but not for any $R > I$. Note that this importance value depends on a specific distance function. In Figure 8, we show strict, left, right, and flat importances for the distance $|a - b|$.

If a point is not a strict (left, right, flat) important extremum for any value of R, we say that it has no strict (left, right, flat) importance. For example, the point a_7 in Figure 8 has no right or flat importance, whereas its strict importance is 1 and left importance is 3.

If we use the strict, left, and right extrema in compression, then we define the overall importance of an extremum as the maximum of its strict, left, and right importances. If we use only the strict and left extrema, then the overall importance is the maximum of the strict and left importances. For convenience, we define the importance of the two endpoints as infinity, which means that we always include them in a compressed series.

We now give basic properties of importances, which readily follow from the definition.

Lemma 3:

- An extremum has a strict importance if and only if it is a strict extremum.
- A left (right) extremum has a left (right) importance; furthermore, if an extremum has a left (right) importance, then it is either a left (right) or strict extremum.
- A flat extremum has a flat importance; furthermore, it has no strict, left, or right importance.

We give a simple procedure for determining the strict importance of a minimum, stated as a lemma, which is illustrated in Figure 9. The procedure for determining the strict importance of a maximum is symmetric.

Lemma 4 (Strict importance): Suppose that a_i is a strict minimum, and let $a_{il}, ..., a_{i-1}$ and $a_{i+1}, ..., a_{ir}$ be the maximal segments to the left and to the right of a_i whose values are strictly greater than a_i. Let a_{lm} be the maximal value among $a_{il}, ..., a_{i-1}$, and a_{rm} be the maximal value among $a_{i+1}, ..., a_{ir}$. Then, the strict importance of a_i is the minimum of the distances $dist(a_i, a_{lm})$ and $dist(a_i, a_{rm})$.

We can use similar procedures for determining the left, right, and flat importances of minima and maxima. We state the results for the left and flat importances of minima; the other procedures are symmetric.

Lemma 5 (Left importance): Suppose that a_i is a strict or left minimum, and let $a_{il}, ..., a_{i-1}$ be the maximal segment to the left of a_i whose values are strictly greater than a_i, and $a_{i+1}, ..., a_{ir}$ be the maximal segment to the right of a_i whose values are no smaller than a_i. If all values among $a_{i+1}, ..., a_{ir}$ are strictly greater than a_i, then a_i has no left importance.

Otherwise, let a_{rt} be the first point among $a_{i+1}, ..., a_{ir}$ with value equal to a_i. Furthermore, let a_{lm} be the maximal value among $a_{il}, ..., a_{i-1}$, and let a_{rm} be the maximal value among $a_{rt+1}, ..., a_{ir}$. If some value among $a_{i+1}, ..., a_{rt-1}$ is no smaller than $\min(a_{lm}, a_{rm})$, then a_i has no left importance; else, its left importance is the minimum of the distances $dist(a_i, a_{lm})$ and $dist(a_i, a_{rm})$.

Lemma 6 (Flat importance): Suppose that a_i is a strict, left, right, or flat minimum, and let a_{il}, ..., a_{i-1} and a_{i+1}, ..., a_{ir} be the maximal segments to the left and to the right of a_i whose values are no smaller than a_i. If all values among a_{il}, ..., a_{i-1} are strictly greater than a_i, or all values among a_{i+1}, ..., a_{ir} are strictly greater than a_i, then a_i has no flat importance.

Otherwise, let a_{lt} be the last point among a_{il}, ..., a_{i-1} with value equal to a_i, and let a_{rt} be the first point among a_{i+1}, ..., a_{ir} with value equal to a_i. Furthermore, let a_{lm} be the maximal value among a_{il}, ..., a_{lt-1}, and let a_{rm} be the maximal value among a_{rt+1}, ..., a_{ir}. If some value among a_{lt+1}, ..., a_{i-1} or among a_{i+1}, ..., a_{rt-1} is no smaller than $\min(a_{lm}, a_{rm})$, then a_i has no flat importance; else, its flat importance is the minimum of the distances $dist(a_i, a_{lm})$ and $dist(a_i, a_{rm})$.

The following properties of importances readily follow from Lemmas 5 and 6.

Lemma 7:

- If a point has strict (left, right, flat) importance for some distance function, then it has strict (left, right, flat) importance for any other distance function.
- If a point has right importance, then it has no left importance; similarly, if a point has left importance, then it has no right importance.
- If a point has strict and left (right) importances, then its strict importance is strictly smaller than its left (right) importance.
- If a point has strict and flat importances, then its strict importance is strictly smaller than its flat importance.
- If a point has left (right) and flat importances, then its left (right) importance is strictly smaller than its flat importance.

If we define a new distance function through a composition of distances, we can calculate the importances of extrema for this new distance based on the importances for the original distances.

Lemma 8: Suppose that $f(d_1, ..., d_q)$ is a real-valued function on nonnegative real arguments such that $f(0, ..., 0) = 0$ and f is monotonically increasing on each of its arguments. Suppose further that $dist_1, ..., dist_q$ are distance functions, and that the strict (left, right, flat) importance of a given extremum for $dist_1$ is I_1, its strict (left, right, flat) importance for $dist_2$ is I_2, and so on. Then, the strict (left, right, flat) importance of this extremum for the distance function $f(dist_1(a, b), ..., dist_q(a, b))$ is $f(I_1, ..., I_q)$.

In Figure 10, we give a fast procedure for calculating the strict, left, right, and flat importances of all minima in a series; the procedure for maxima is symmetric. It runs in linear time, and it uses the memory in proportion to the number of extrema; that is, if the original series includes n points, and m of them are extrema, then the procedure runs in $O(n)$ time and uses $O(m)$ memory.

It computes the importances of all minima in one pass through the series. When it identifies a minimum a_i, it puts i onto stack S_1 and into list L, and then it puts three local maxima surrounding a_i (Figure 11) onto stacks S_2, S_3, and S_4. When it later reaches the end of the interval $a_i, ..., a_{ir}$ (Figure 11), it removes i from the stack and calculates the importances of a_i. We denote the strict importance of a_i by $strict\text{-}imp_i$, left importance by $left\text{-}imp_i$, right importance by $right\text{-}imp_i$, and flat importance by $flat\text{-}imp_i$.

5. COMPRESSION RATE

We next consider the problem of selecting important extrema according to a given *compression rate,* which is the percentage of points removed during the compression. For example, if a series includes hundred points and a given rate is 90%, then we select ten points, which include the two endpoints and the eight most important extrema. As another example, the compression rate in Figure 1 is 60% since we have selected eight out of twenty points and removed the other twelve points.

We assume that a given compression rate is no smaller than the percentage of nonextremal points in a series; for example, if a hundred-point series includes eighty nonextremal points, a rate must be at least 80%. If the series includes n points, the total number of points in its compressed version must be $\lfloor n \cdot (1 - rate) \rfloor$, including the two

endpoints. Thus, the number of extrema selected during the compression must be $s = \lfloor n \cdot (1 - rate) \rfloor - 2$.

We describe two algorithms that compress a series at a given rate. The first is a linear-time algorithm that performs three passes through the series. The second is slower, but it can process new points as they arrive, without storing the entire series in memory or on disk.

Three-pass algorithm: The linear-time algorithm includes three main steps (Figure 12). First, it calls the procedure in Figure 10, which calculates the importances of all minima, and also applies the symmetric procedure for calculating the importances of all maxima. Second, it finds the *sth order statistic*, that is, the *s*th greatest value, among the importances of all extrema. Third, it selects all extrema with importances no smaller than the *s*th greatest value. For an *n*-point series with *m* extrema, it runs in $O(n)$ time and takes $O(m)$ memory.

The resulting compression rate may not be exactly equal to the given *rate* value. If the series has multiple extrema with the importance equal to the *s*th greatest value, then the procedure selects all these extrema, which means that the actual compression rate may be somewhat lower than the given rate. For example, suppose that we compress the series in Figure 1 and the given rate is 70%. Then, $s = \lfloor 20 \cdot (1 - 0.7) \rfloor - 2 = 4$, the *s*th order statistic of importances is 3, and the algorithm selects all points with importances no smaller than 3. The series includes eight such points, which means that the resulting rate is 60%.

One-pass algorithm: The other algorithm is a modified version of the procedure in Figure 10, which includes two main changes. First, we extend it to calculate the importances of both minima and maxima in one pass through the series. Second, we replace the linked list *L* with a red-black tree, which contains only the *s* most important extrema, indexed on their importances.

When the procedure determines the importance of an extremum a_i, it makes the respective update to the tree. If the tree includes fewer than *s* extrema, the procedure adds a_i to the tree. If it includes *s* extrema and they all are more important than a_i, the procedure does not change the tree. Finally, if it includes *s* extrema and some of them are

less important than a_i, then the procedure adds a_i and removes the least important extremum from the tree.

After processing the series, the procedure sorts the s selected extrema by their indices, and outputs their values, indices, and importances. For an n-point series with m extrema, the running time is $O(m \cdot \lg s + n)$, and the required memory is $O(m)$.

Different distance functions: The selection of important extrema depends not only on the compression rate but also on the specific distance function. That is, the use of different distances may lead to different compression results. For example, suppose that we are compressing the series in Figure 13 with the 30% rate. If the distance function is |a − b|, the selected extrema are as shown in Figure 13(a). On the other hand, if the distance is $\frac{|a-b|}{|a|+|b|}$, the compression procedure selects the extrema marked in Figure 13(b).

We next state a condition under which different distance functions lead to the same compression.

> **Lemma 9:** Suppose that $dist_1(a, b)$ is a distance function, $f(d)$ is a strictly increasing function on nonnegative reals, and $dist_2(a, b) = f(dist_1(a, b))$. Then, for every series and every compression rate, the selection of extrema for $dist_1$ is identical to that for $dist_2$.

In other words, a monotonic transformation of a distance function does not affect the selection of the most important extrema. For example, the functions $|a - b|$ and $(a - b)^2$ lead to the same compression.

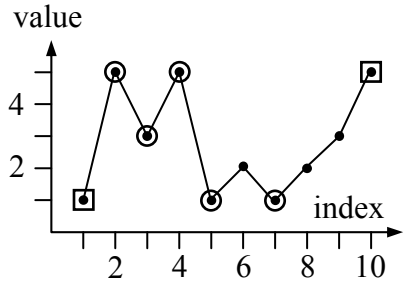
6. CONCLUDING REMARKS

We have formalized the concept of major minima and maxima in a time series, and developed a fast algorithm for computing the importance levels of minima and maxima, which allows fast lossy compression of a series. We can also apply the same concept of importance levels to indexing a database of time series by their minima and maxima, which allows fast retrieval of series similar to a given pattern. We have described this indexing technique in Gandhi's masters thesis [Gandhi, 2004], which is a continuation of the work by Fink and Pratt [2003] on indexing of time series.

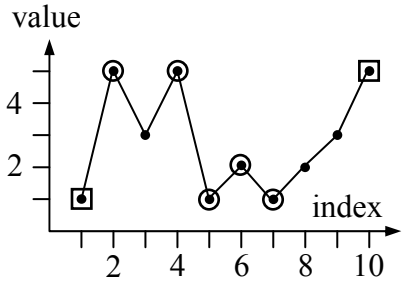
THREE-PASS-COMPRESSION — Compression at a given rate
Input: Series a_1, ..., a_n, function *dist*, and compression *rate*
Output: Compressed series

- Calculate the importances of all minima and maxima, using the procedure in Figure 10
- Determine the target number of important extrema, which is $s = \lfloor n \cdot (1 - rate) \rfloor - 2$, and find the sth greatest importance, that is, the sth order statistic of all importances
- Output the extrema whose importances are no smaller than the sth greatest importance

Figure 12: Three-pass compression at a given rate. This procedure requires storing an entire series in memory or on disk.



(a) Compression for the distance $|a - b|$.



(a) Compression for $|a - b| / (|a| + |b|)$.

Figure 13: Example of compression with different distance functions. If we use the 30% compression rate and the distance $|a - b|$, we select the extrema shown on the left. If we use the same rate with the distance $\frac{|a-b|}{|a|+|b|}$, we get the extrema marked on the right.

ACKNOWLEDGEMENTS

We are grateful to Yongwen Liang for his help with implementing and evaluating the described technique. We thank Helen Mukomel for her detailed comments, which helped to improve the presentation. The described work was supported in part by the Berkman Faculty Development Fund at Carnegie Mellon University.

REFERENCES

[Agrawal et al., 1995] Rakesh Agrawal, Guiseppe 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.

[André-Jönsson and Badal, 1997] 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.

[Box et al., 2008] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. Time Series Analysis: Forecasting and Control. John Wiley and Sons, Hoboken, NJ, fourth edition, 2008.

[Burrus et al., 1997] C. Sidney Burrus, Ramesh A. Gopinath, and Haitao Guo. Introduction to Wavelets and Wavelet Transforms: A Primer. Prentice Hall, Englewood Cliffs, NJ, 1997.

[Chan and Fu, 1999] Franky 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.

[Chan et al., 2003] Franky Kin-Pong Chan, Ada Wai-chee Fu, and Clement Yu. Haar wavelets for efficient similarity search of time-series: With and without time warping. IEEE Transactions of Knowledge and Data Engineering, 15(3), pages 686–705, 2003.

[Cryer and Chan, 2009] Jonathan D. Cryer and Kung-Sik Chan. Time Series Analysis: With Applications in R. Springer, Berlin, Germany, second edition, 2009.

[Das et al., 1998] Gautam Das, King-In 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.

[Di et al., 2007] Sheng Di, Hai Jin, Shengli Li, Jing Tie, and Ling Chen. Efficient time series data classification and compression in distributed monitoring. In Takashi Washio, Zhi-Hua Zhou, Joshua Zhexue Huang, Xiaohua Hu, Jinyan Li, Chao Xie, Jieyue He, Deqing Zou, Kuan-Ching Li, and Mário M. Freire, editors, Emerging Technologies in Knowledge Discovery and Data Mining, pages 389–400. Springer, Berlin, Germany, 2007.

[Dorr and Denton, 2009] Dietmar H. Dorr and Anne M. Denton. Establishing relationships among patterns in stock market data. Data and Knowledge Engineering, 68(3), pages 318–337, 2009.

[Eruhimov et al., 2006] Victor Eruhimov, Vladimir Martyanov, Peter Raulefs, and Eugene Tuv. Combining unsupervised and supervised approaches to feature selection

for multivariate signal compression. In Emilio Corchado, Hujun Yin, Vicente Botti, and Colin Fyfe, editors, Intelligent Data Engineering and Automated Learning, pages 480–487. Springer, Berlin, Germany, 2006.

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

[Fu et al., 2008] Tak-chung Fu, Fu-lai Chung, Robert Luk, and Chak-man Ng. Representing financial time series based on data point importance. Engineering Applications of Artificial Intelligence, 21(2), pages 277–300, 2008.

[Gandhi, 2004] Harith Suman Gandhi. Important extrema of time series: Theory and applications. Master's thesis, Department of Computer Science and Engineering, University of South Florida, 2004.

[Graps, 1995] Amara L. Graps. An introduction to wavelets. IEEE Computational Science and Engineering, 2(2), pages 50–61, 1995.

[Guralnik and Srivastava, 1999] Valery Guralnik and Jaideep Srivastava. Event detection from time series data. In Proceedings of the Fifth ACM SIGMOD International Conference on Knowledge Discovery and Data Mining, pages 33–42, 1999.

[Guralnik et al., 1998] Valery Guralnik, Duminda Wijesekera, and Jaideep Srivastava. Pattern-directed mining of sequence data. In Proceedings of the Fourth ACM International Conference on Knowledge Discovery and Data Mining, pages 51–57, 1998.

[Gusfield, 1997] Dan Gusfield. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, Cambridge, United Kingdom, 1997.

[Huang and Yu, 1999] 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.

[Ikeda et al., 1999] 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.

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

[Keogh and Pazzani, 1998] 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.

[Kriegel et al., 2008] Hans-Peter Kriegel, Peer Kröger, Alexey Pryakhin, and Matthias Renz. Analysis of time series using compact model-based descriptions. In Jayant R. Haritsa, Ramamohanarao Kotagiri, and Vikram Pudi, editors, Database Systems for Advanced Applications, pages 698–701. Springer, Berlin, Germany, 2008.

[Lam and Wong, 1998] Sze Kin Lam and Man Hon Wong. A fast projection algorithm for sequence data searching. Data and Knowledge Engineering, 28(3), pages 321–339, 1998.

[Last et al., 2001] Mark Last, Yaron Klein, and Abraham Kandel. Knowledge discovery in time series databases. IEEE Transactions on Systems, Man, and Cybernetics, Part B, 31(1), pages 160–169, 2001.

[Lin and Risch, 1998] 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.

[Montgomery et al., 2008] Douglas C. Montgomery, Cheryl L. Jennings, and Murat Kulahci. Introduction to Time Series Analysis and Forecasting. John Wiley and Sons, Hoboken, NJ, 2008.

[Park et al., 1999] 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.

[Perng et al., 2000] 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.

[Pratt, 2001] Kevin B. Pratt. Locating patterns in discrete time series. Master's thesis, Computer Science and Engineering, University of South Florida, 2001.

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

[Qu et al., 1998] 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.

[Sheikholeslami et al., 1998] 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.

[Singh and McAtackney, 1998] Sameer Singh and Paul McAtackney. Dynamic time-series forecasting using local approximation. In Proceedings of the Tenth IEEE International Conference on Tools with Artificial Intelligence, pages 392–399, 1998.

[Stoffer, 1999] David S. Stoffer. Detecting common signals in multiple time series using the spectral envelope. Journal of the American Statistical Association, 94, pages 1341–1356, 1999.

[Yi et al., 2000] 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.