Important Extrema of Time Series

Eugene Fink
Computer Science, Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
e.fink@cs.cmu.edu, www.cs.cmu.edu/∼eugene

Harith Suman Gandhi
11336 Cypress Reserve Drive
Tampa, Florida 33626
suman.reddy@nielsen.com

Abstract – *We describe a technique for fast lossy compression of a time series based on the assignment of importance levels to its minima and maxima.*

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. 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).

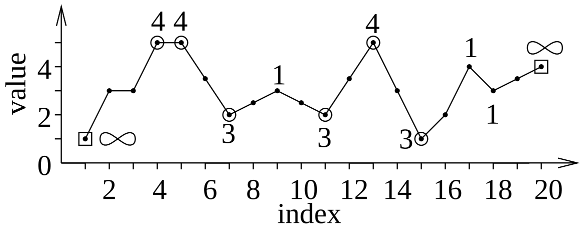


Figure 1: Example of a time series and its extrema. We show the importance of each extremum and mark the extrema with importances greater than 1.

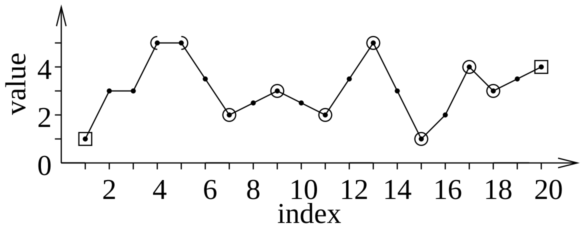
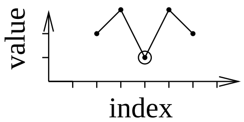


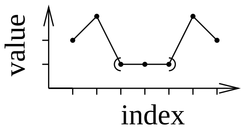
Figure 2: Compression by extracting all extrema. We show strict extrema by circles, left and right extrema by half-circles, and end-points by squares.

2 Extrema

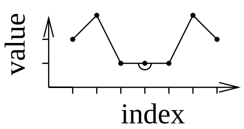
We begin with a simple compression based on the extraction of all extrema (Figure 2). We distinguish four types of extrema, called strict, left, right, and flat (Figure 3). We give the definition of strict, left, right, and flat minima; the definition for maxima is similar.



(a) Strict minimum.



(b) Left and right minima.



(c) Flat minimum.

Figure 3: Four types of minima.

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 $a_{left-1} > a_{left} = ... = a_i = ... = a_{right} < a_{right+1}$.

ALL-EXTREMA — Finding 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*, *type*) — Output extrema
if *left* = *right*
  then output(a_{right}, *right*, *type*, "strict")
  else output(a_{left}, *left*, *type*, "left")
    for *flat* = *left* + 1 **to** *right* − 1 **do**
      output(a_{flat}, *flat*, *type*, "flat")
    output(a_{right}, *right*, *type*, "right")

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

In Figure 4, we give a procedure that identifies all extrema and determines their types, which is an extended version of the compression algorithm developed by Pratt and Fink [8]; it takes linear time and constant memory. Note that it can process new points as they arrive, one by one, without storing the series in memory; 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 the two end-points, and discarding the flat extrema and nonextremal points (Figure 2). We now state an important property of this compression, called monotonicity.

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 $a_{i_c} \leq a_i \leq a_{i_{c+1}}$ or $a_{i_{c+1}} \leq a_i \leq a_{i_c}$.

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

3 Important extrema

We can achieve a higher compression rate by selecting only certain important extrema. We first introduce the notion of distance, 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 $a \leq b \leq c$, then $dist(a, b) \leq dist(a, c)$ and $dist(b, c) \leq dist(a, c)$.

For example, the functions $|a - b|$, $\frac{|a-b|}{|a|+|b|}$, and $\frac{|a-b|}{\max(|a|,|b|)}$ are three different distance functions. We do not assume that distances satisfy the triangle inequality; thus, $dist(a, c)$ may be greater than $dist(a, b) + dist(b, c)$.

We next describe a method for combining distance functions into a more complex distance function.

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.

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

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 the minimal value of some segment $a_{il}, ..., a_{ir}$, and the end-point 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 is a strict important minimum if it is a strict minimum of some segment, and the end-point values of this 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 similar; 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 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 the two end-points, and discarding the other points. In Figure 7, we show the selected extrema for the distance $|a - b|$ with $R = 3$. Note that the resulting 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 end-points, 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 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.

4 Importance levels

We next 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)

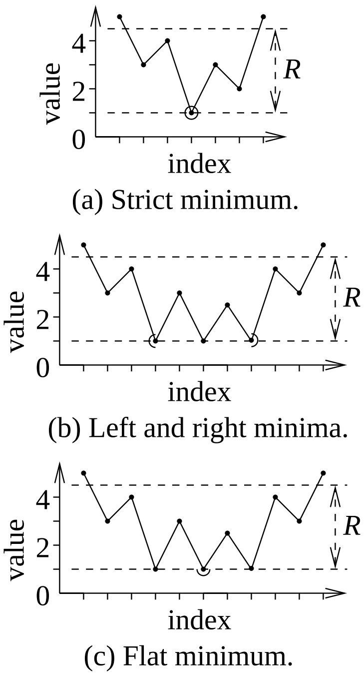


Figure 5: Four types of important minima.

extremum for $R = I$, but not for any $R > I$. Note that this importance value depends on a specific distance function.

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 its 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 end-points as infinity, which means that we always include them into a compressed series.

We now give basic properties of importances.

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; determining the strict importance of a maximum is similar.

IMPORTANT-EXTREMA — Finding 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 min after 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 max after 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*, *type*) — Output extrema
if $left = right$
  then output$(a_{right}$, *right*, *type*, "strict")
  else output$(a_{left}$, *left*, *type*, "left")
    for $flat = left + 1$ **to** $right - 1$ **do**
      if $a_{flat} = a_{left}$
        then output$(a_{flat}$, *flat*, *type*, "flat")
    output$(a_{right}$, *right*, *type*, "right")

Figure 6: Selecting the important extrema.

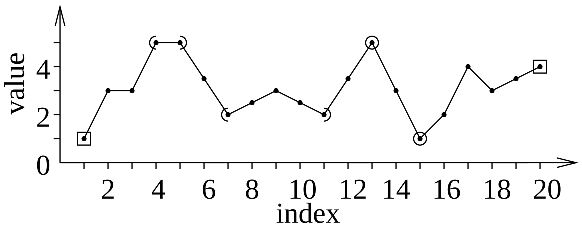
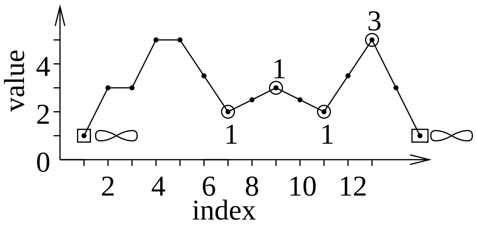
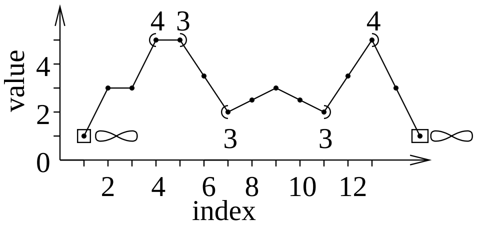


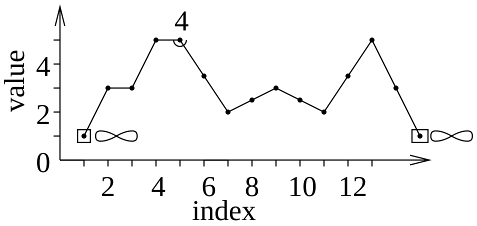
Figure 7: Important extrema for the distance $|a - b|$ with $R = 3$. We show the strict extrema by circles, left and right extrema by half-circles, and end-points by squares.



(a) Strict importances.



(b) Left and right importances.



(c) Flat importances.

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

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 let 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 the minima and maxima. We state the results for the left and flat importances of minima; the other procedures are similar.

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 seg-

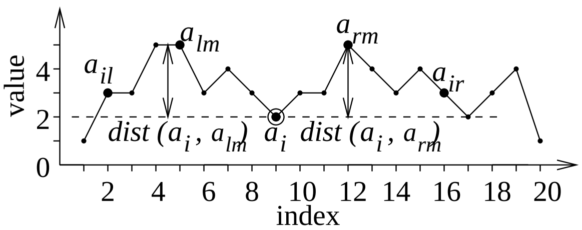


Figure 9: Computation of the strict importance of a minimum. To determine the importance of a_i, we identify the maximal segments $a_{il}, ..., a_{i-1}$ and $a_{i+1}, ..., a_{ir}$ whose values are strictly greater than a_i, find the maximal values a_{lm} and a_{rm} in these segments, and compute the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

ment 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, then 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 similar. 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.

The procedure 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-imp*$_i$, left importance by *left-imp*$_i$, right importance by *right-imp*$_i$, and flat importance by *flat-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 compression rate is 90%, then we select the ten most important extrema. As another example, the compression rate in Figure 1 is 60% since we have selected eight out of twenty points.

We assume that the 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, the compression rate must be at least 80%. If the series includes n points, the number of selected extrema must be $s = \lfloor n \cdot (1 - rate) \rfloor$.

We give 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 algorithm is slower, but it can process new points as they arrive, without storing the entire series in memory.

ALL-IMPORTANCES — Finding 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
$i = 1$; $left = 1$; $max\text{-}value = a_1$
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($left$, $max\text{-}value$)
  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** $i = i + 1$
  $max\text{-}value = a_i$; $i = i + 1$; $left = i$
  if S_1 is not empty **then** TOP(S_3) = $max\text{-}value$
while S_1 is not empty **do** $max\text{-}value$ = POP-MIN($max\text{-}value$)
for every i in the list L **do**
  output($a_i, i, strict\text{-}imp_i, left\text{-}imp_i, right\text{-}imp_i, flat\text{-}imp_i$)

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

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

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.

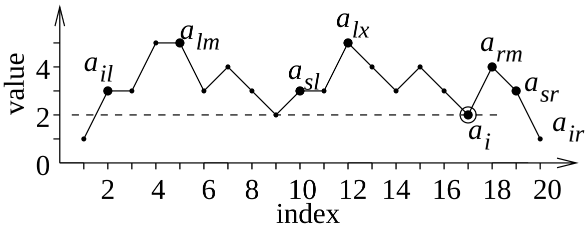


Figure 11: Illustration of the local maxima surrounding a minimum a_i, which are used in computing the importances of a_i; the notation is the same as in Figure 9. The point a_{lx} is the maximum of the segment $a_{sl}, ..., a_{i-1}$, whose values are strictly greater than a_i; the algorithm in Figure 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.

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

$s = \lfloor n \cdot (1 - rate) \rfloor$
Calculate the importances of all extrema (Figure 10)
Find the sth greatest importance
Output the extrema whose importances are no smaller
  than the sth greatest importance

Figure 12: Three-pass compression at a given rate, which requires storing an entire series in memory.

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 extrema. Second, it finds the sth order statistic among the importance values, that is, the sth greatest value. Third, it selects all extrema with importances no smaller than the sth greatest value. For an n-point series with m extrema, it runs in $O(n)$ time and takes $O(m)$ memory.

Note that 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 sth largest value, then the procedure selects all these extrema, which means that the actual compression rate may be lower than the given rate. For example, suppose that we compress the series in Figure 1 and the given rate is 70%. Then, $s = \lfloor 20 \cdot (1 - 0.7) \rfloor = 6$, the sth 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 compression rate is 60%.

One-pass algorithm: The one-pass 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 the tree includes s extrema and they all are more important than a_i, then the procedure does not change the tree. Finally, if the tree 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 $O(m)$.

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 index 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 [2], which is a continuation of the work by Pratt and Fink on the indexing of time series [1, 8].

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] 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.

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