

## NRC Publications Archive Archives des publications du CNRC

### An Optimal Linear Time Algorithm for Quasi-Monotonic Segmentation Lemire, Daniel; Brooks, Martin; Yan, Y.

This publication could be one of several versions: author's original, accepted manuscript or the publisher's version.  
/ La version de cette publication peut être l'une des suivantes : la version prépublication de l'auteur, la version acceptée du manuscrit ou la version de l'éditeur.

#### **Publisher's version / Version de l'éditeur:**

*IEEE International Conference on Data Mining (ICDM) [Proceedings], 2005*

**NRC Publications Archive Record / Notice des Archives des publications du CNRC :**  
<https://nrc-publications.canada.ca/eng/view/object/?id=67ba23e4-431c-4a75-9ad7-475f28c1d1a>  
<https://publications-cnrc.canada.ca/fra/voir/objet/?id=67ba23e4-431c-4a75-9ad7-475f28c1d1a7>

Access and use of this website and the material on it are subject to the Terms and Conditions set forth at  
<https://nrc-publications.canada.ca/eng/copyright>

READ THESE TERMS AND CONDITIONS CAREFULLY BEFORE USING THIS WEBSITE.

L'accès à ce site Web et l'utilisation de son contenu sont assujettis aux conditions présentées dans le site  
<https://publications-cnrc.canada.ca/fra/droits>

LISEZ CES CONDITIONS ATTENTIVEMENT AVANT D'UTILISER CE SITE WEB.

**Questions?** Contact the NRC Publications Archive team at  
PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca. If you wish to email the authors directly, please see the first page of the publication for their contact information.

**Vous avez des questions?** Nous pouvons vous aider. Pour communiquer directement avec un auteur, consultez la première page de la revue dans laquelle son article a été publié afin de trouver ses coordonnées. Si vous n'arrivez pas à les repérer, communiquez avec nous à PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca.



National Research  
Council Canada

Conseil national  
de recherches Canada

Institute for  
Information Technology

Institut de technologie  
de l'information

# **NRC - CNRC**

---

## *An Optimal Linear Time Algorithm for Quasi-Monotonic Segmentation \**

Lemire, D., Brooks, M., and Yan, Y.  
November 2005

\* published at the IEEE International Conference on Data Mining (ICDM).  
New Orleans, Louisiana, USA. November 27-30, 2005. NRC 48277.

Copyright 2005 by  
National Research Council of Canada

Permission is granted to quote short excerpts and to reproduce figures and tables  
from this report, provided that the source of such material is fully acknowledged.

## An Optimal Linear Time Algorithm for Quasi-Monotonic Segmentation

Daniel Lemire\*<sup>†</sup>, Martin Brooks, Yuhong Yan<sup>‡</sup>

<sup>†</sup>University of Quebec at Montreal (UQAM), 100 Sherbrooke West, Montréal, Qc, H2X 3P2, Canada

<sup>‡</sup> National Research Council of Canada, 1200 Montreal Road, Ottawa, ON, Canada, K1A 0R6

(Received 00 Month 200x; revised 00 Month 200x; in final form 00 Month 200x)

Monotonicity is a simple yet significant qualitative characteristic. We consider the problem of segmenting a sequence in up to  $K$  segments. We want segments to be as monotonic as possible and to alternate signs. We propose a quality metric for this problem using the  $l_\infty$  norm, and we present an optimal linear time algorithm based on novel formalism. Moreover, given a precomputation in time  $O(n \log n)$  consisting of a labeling of all extrema, we compute any optimal segmentation in constant time. We compare experimentally its performance to two piecewise linear segmentation heuristics (top-down and bottom-up). We show that our algorithm is faster and more accurate. Applications include pattern recognition and qualitative modeling.

*Keywords:* Time Series, Segmentation, Monotonicity, Design of Algorithms

*AMS Subject Classifications:* H.2.8

### 1. Introduction

Monotonicity is one of the most natural and important qualitative properties for sequences of data points. It is easy to determine where the values are strictly going up or down, but we only want to identify significant monotonicity. For example, the drop from 2 to 1.9 in the array 0, 1, 2, 1.9, 3, 4 might not be significant and might even be noise-related. The quasi-monotonic segmentation problem is to determine where the data is approximately increasing or decreasing.

---

\*Corresponding author. Email: lemire@acm.org

This is an expanded version of a conference paper [1].

In practical applications, sequences of values can be quite large: it is not uncommon to have sensors record data at 10 kHz or more, thus generating terabytes of data and billions of data points. As a dimensionality reduction step [2], segmentation divides the data into intervals having homogeneous characteristics (flatness, constant slope [3], unimodality [4], monotonicity [5, 6], step, ramp or impulse [7], and so on). The segmentation points can also be used as markers to indicate a qualitative change in the data. Other applications include frequent pattern mining [8] and time series classification [9]. For qualitative reasoning [10], piecewise monotonic segmentation is especially important as it provides a symbolic model describing system behavior in terms of increasing and decreasing relations between variables.

There is a trade-off between the number of segments and the approximation error. Some segmentation algorithms [5] give a segmentation having no more than  $K$  segments while attempting to minimize the error  $\epsilon$ ; other algorithms [6] attempt to minimize the number of segments ( $K$ ) given an upper bound on the error  $\epsilon$ . We are concerned with the first type of algorithm in this paper.

Using dynamic programming or other approaches, most segmentation problems can be solved in time  $O(n^2)$ . Other solutions to this problem, using machine learning to classify the pairs of data points [10], are even less favorable since they have higher complexity. However, it is common for sequence of data points to be massive and segmentation algorithms have to have complexity close to  $O(n)$  to be competitive. While approximate linear regression segmentation algorithms can be  $O(n)$ , we show that using a linear regression error to segment according to monotonicity is not an ideal solution.

We present a metric for the quasi-monotonic segmentation problem called the Optimal Monotonic Approximation Function Error (OMAFE); this metric differs from previously introduced OPMAFE metric [5] since it applies to all segmentations and not just “extremal” segmentations. We formalize the novel concept of a maximal \*-pair and shows that it can be used to define a unique labeling of the extrema leading to an optimal segmentation algorithm. We also present an optimal linear time algorithm to solve the quasi-monotonic segmentation problem given a segment budget together with an experimental comparison to quantify the benefits of our algorithm.

## 2. Monotonicity Error Metric (OMAFE)

Finding the best piecewise monotonic approximation can be viewed as a classical functional approximation problem [11], but we are concerned only with discrete sequences.

Suppose  $n$  samples noted  $F : D = \{x_1, \dots, x_n\} \rightarrow \mathbb{R}$  with  $x_1 < x_2 < \dots < x_n$ . We define,  $F|_{[a,b]}$  as the restriction of  $F$  over  $D \cap [a, b]$ . We seek the best monotonic (increasing or decreasing) function  $f : \mathbb{R} \rightarrow \mathbb{R}$  approximating  $F$ . Let  $\Omega_\uparrow$  (resp.  $\Omega_\downarrow$ ) be the set of all monotonic increasing (resp. decreasing) functions. The **Optimal Monotonic Approximation Function Error (OMAFE)** is  $\min_{f \in \Omega} \max_{x \in D} |f(x) - F(x)|$  where  $\Omega$  is either  $\Omega_\uparrow$  or  $\Omega_\downarrow$ .

The segmentation of a set  $D$  is a sequence  $S = X_1, \dots, X_K$  of intervals in  $\mathbb{R}$  with  $[\min D, \max D] = \bigcup_i X_i$  such that  $\max X_i = \min X_{i+1} \in D$  and  $X_i \cap X_j = \emptyset$  for  $j \neq i+1, i, i-1$ . Alternatively, we can define a segmentation from the set of points  $X_i \cap X_{i+1} = \{y_{i+1}\}$ ,  $y_1 = \min X_1$ , and  $y_{K+1} = \max X_K$ . Given  $F : \{x_1, \dots, x_n\} \rightarrow \mathbb{R}$  and a segmentation, the Optimal Monotonic Approximation Function Error (OMAFE) of the segmentation is  $\max_i \text{OMAFE}(F|_{X_i})$  where the monotonicity type (increasing or decreasing) of the segment  $X_i$  is determined by the sign of  $F(\max X_i) - F(\min X_i)$ . Whenever  $F(\max X_i) = F(\min X_i)$ , we say the segment has no direction and the best monotonic approximation is just the flat function having value  $(\max F|_{X_i} - \min F|_{X_i})/2$ . The error is computed over each interval independently; optimal monotonic approximation functions are not required to agree at  $\max X_i = \min X_{i+1}$ . Segmentations should alternate between increasing and decreasing, otherwise sequences such as 0, 2, 1, 0, 2 can be segmented as two increasing segments 0, 2, 1 and 1, 0, 2: we consider it is natural to aggregate segments with the same monotonicity.

We solve for the best monotonic function as follows. If we seek the best monotonic increasing function, we first define  $\bar{f}_\uparrow(x) = \max\{F(y) : y \leq x\}$  (the maximum of all previous values) and  $\underline{f}_\uparrow(x) = \min\{F(y) : y \geq x\}$  (the minimum of all values to come). If we seek the best monotonic decreasing function, we define  $\bar{f}_\downarrow(x) = \max\{F(y) : y \geq x\}$  (the maximum of all values to come) and  $\underline{f}_\downarrow(x) = \min\{F(y) : y \leq x\}$  (the minimum of all previous values). These functions, which can be computed in linear time, are all we need to

solve for the best approximation function as shown by the next theorem which is a well-known result [12].

**THEOREM 2.1** *Given  $F : D = \{x_1, \dots, x_n\} \rightarrow \mathbb{R}$ , a best monotonic increasing approximation function to  $F$  is  $f_\uparrow = (\bar{f}_\uparrow + \underline{f}_\uparrow)/2$  and a best monotonic decreasing approximation function is  $f_\downarrow = (\bar{f}_\downarrow + \underline{f}_\downarrow)/2$ . The corresponding error (OMAFE) is  $\max_{x \in D} (|\bar{f}_\uparrow(x) - \underline{f}_\uparrow(x)|)/2$  or  $\max_{x \in D} (|\bar{f}_\downarrow(x) - \underline{f}_\downarrow(x)|)/2$  respectively.*

The implementation of the algorithm suggested by the theorem is straight-forward. Given a segmentation, we can compute the OMAFE in  $O(n)$  time using at most two passes.

### 3. A Scale-Based Algorithm for Quasi-Monotonic Segmentation

We use the following proposition to prove that the segmentations we generate are optimal (see Theorem 3.8).

**PROPOSITION 3.1** *A segmentation  $y_1, \dots, y_{K+1}$  of  $F : D = \{x_1, \dots, x_n\} \rightarrow \mathbb{R}$  with alternating monotonicity has a minimal OMAFE  $\epsilon$  for a number of alternating segments  $K$  if*

(i)  $F(y_i) = \max F([y_{i-1}, y_{i+1}])$  or  $F(y_i) = \min F([y_{i-1}, y_{i+1}])$  for  $i = 2, \dots, K$ ;

(ii) in all intervals  $[y_i, y_{i+1}]$  for  $i = 1, \dots, K$ , there exists  $z_1, z_2$  such that  $|F(z_2) - F(z_1)| > 2\epsilon$ .

*Proof* Let the original segmentation be the intervals  $S_1, \dots, S_K$  and consider a new segmentation with intervals  $T_1, \dots, T_K$ . Assume that the new segmentation has lower error (as given by OMAFE). Let  $S_i = [y_i, y_{i+1}]$  and  $T_i = [y'_i, y'_{i+1}]$ .

If any segment  $T_m$  contains a segment  $S_j$ , then the existence of  $z_1, z_2$  in  $[y_j, y_{j+1}]$  such that  $|F(z_2) - F(z_1)| > 2\epsilon$  and  $\text{OMAFE}(T_m) \leq \epsilon$  implies that  $T_m$  and  $S_j$  have the same monotonicity.

We show that each pair of intervals  $S_i, T_i$  has nonempty intersection. Suppose not, and let  $i$  be the smallest index such that  $S_i \subset T_{i-1}$ . Since  $S_i$  and  $T_{i-1}$  have the same monotonicity, for each  $j < i$ ,  $S_j$  and  $T_j$  have opposite monotonicity. Now consider the  $i-1$  intervals  $T_1, \dots, T_{i-1}$  and the  $i$  points  $y_1, \dots, y_i$ . At least one interval contains two consecutive points; choose the largest  $j < i$  such that  $T_j$  contains  $y_j, y_{j+1}$ . But then  $S_j \subset T_j$ , contradicting at least one of the assumptions  $|F(z_2) - F(z_1)| > 2\epsilon$  for  $z_1, z_2 \in S_i$  and  $\text{OMAFE}(T_j) \leq \epsilon$ .

It now follows that each pair of intervals  $S_i, T_i$  has the same monotonicity.

Since  $\text{OMAFE}(T) < \text{OMAFE}(S)$ , we can choose an index  $j$  such that  $\text{OMAFE}(T_j) < \text{OMAFE}(S_j)$ .

We show that there exists another index  $p$  such that  $\text{OMAFE}(T_p) \geq \text{OMAFE}(S_j)$ , thus contradicting  $\text{OMAFE}(T) < \text{OMAFE}(S)$ . Suppose  $S_j$  is increasing; the proof is similar for the opposite case. Then there exist  $x < z \in S_j$  such that  $F(x) - F(z) = 2 \times \text{OMAFE}(S_j)$ . From  $\text{OMAFE}(T_j) < \text{OMAFE}(S_j)$  it follows that at least one of  $x$  or  $z$  lies in  $S_j - T_j$ , and hence  $F(x) - F(y_j) \geq 2 \times \text{OMAFE}(S_j)$  or  $F(y_{j+1}) - F(z) \geq 2 \times \text{OMAFE}(S_j)$ . Thus  $\text{OMAFE}(T_p) \geq \text{OMAFE}(S_j)$  for either  $p = j - 1$  or  $p = j + 1$ .  $\square$

For simplicity, we assume  $F$  has no consecutive equal values, i.e.  $F(x_i) \neq F(x_{i+1})$  for  $i = 1, \dots, n - 1$ ; our algorithms assume all but one of consecutive equal values values have been removed. We say  $x_i$  is a maximum if  $i \neq 1$  implies  $F(x_i) > F(x_{i-1})$  and if  $i \neq n$  implies  $F(x_i) > F(x_{i+1})$ . Minima are defined similarly.

Our mathematical approach is based on the concept of  $\delta$ -pair [13] (see Fig. 1):

**Definition 3.2** The tuple  $x, y$  ( $x < y \in D$ ) is a  $\delta$ -pair (or a pair of scale  $\delta$ ) for  $F$  if  $|F(y) - F(x)| \geq \delta$  and for all  $z \in D$ ,  $x < z < y$  implies  $|F(z) - F(x)| < \delta$  and  $|F(y) - F(z)| < \delta$ . A  $\delta$ -pair's *direction* is *increasing* or *decreasing* according to whether  $F(y) > F(x)$  or  $F(y) < F(x)$ .

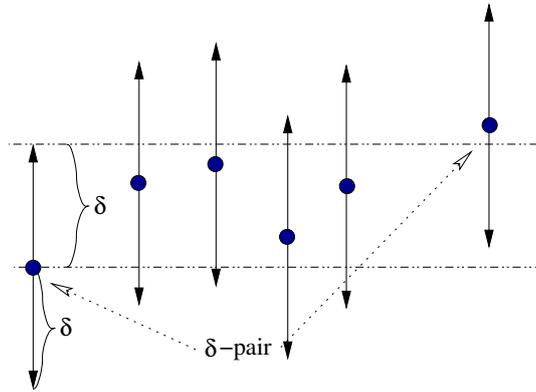
$\delta$ -Pairs having opposite directions cannot overlap but they may share an end point.  $\delta$ -Pairs of the same direction may overlap, but may not be nested. We use the term “\*-pair” to indicate a  $\delta$ -pair having an unspecified  $\delta$ . We say that a \*-pair is significant at scale  $\delta$  if it is of scale  $\delta'$  for  $\delta' \geq \delta$ . From a topological viewpoint, a \*-pair is the pairing of critical points used to determine each extremum's persistence [14].

We define  $\delta$ -monotonicity as follows:

**Definition 3.3** Let  $X$  be an interval,  $F$  is  $\delta$ -monotonic on  $X$  if all  $\delta$ -pairs in  $X$  have the same direction;  $F$  is *strictly*  $\delta$ -monotonic when there exists at least one such  $\delta$ -pair. In this case:

- $F$  is  $\delta$ -increasing on  $X$  if  $X$  contains an increasing  $\delta$ -pair.
- $F$  is  $\delta$ -decreasing on  $X$  if  $X$  contains a decreasing  $\delta$ -pair.

A  $\delta$ -monotonic interval  $X$  satisfies  $\text{OMAFE}(X) < \delta/2$ . We say that a \*-pair  $x, y$  is **maximal** if whenever

Figure 1. A  $\delta$ -pair.

$z_1, z_2$  is a  $*$ -pair of a larger scale in the same direction containing  $x, y$ , then there exists a  $*$ -pair  $w_1, w_2$  of an opposite direction contained in  $z_1, z_2$  and containing  $x, y$ . For example, the sequence 1, 3, 2, 4 has 2 maximal  $*$ -pairs: 1, 4 and 3, 2. Maximal  $*$ -pairs of opposite direction may share a common point, whereas maximal  $*$ -pairs of the same direction may not. Maximal  $*$ -pairs cannot overlap, meaning that it cannot be the case that exactly one end point of a maximal  $*$ -pair lies strictly between the end points of another maximal  $*$ -pair; either neither point lies strictly between or both do. In the case that both do, we say that the one maximal  $*$ -pair properly contains the other. All  $*$ -pairs must be contained in a maximal  $*$ -pair.

LEMMA 3.4 *The smallest maximal  $*$ -pair containing a  $*$ -pair must be of the same direction.*

*Proof* Suppose a  $*$ -pair is immediately contained in a maximal  $*$ -pair  $W$ . Suppose  $W$  is not in the same direction, then within  $W$ , seek the largest  $*$ -pair in the same direction as  $P$  and containing  $P$ , then it must be a maximal  $*$ -pair in  $D$  since maximal  $*$ -pairs of different directions cannot overlap.  $\square$

The first and second point of a maximal  $*$ -pair are extrema and the reverse is true as well as shown by the next lemma.

LEMMA 3.5 *Every extremum is either the first or second point of a maximal  $*$ -pair.*

*Proof* The case  $x = x_1$  or  $x = x_n$  follows by inspection. Otherwise,  $x$  is the end point of a left and a right  $*$ -pair. Each  $*$ -pair must immediately belong to a maximal  $*$ -pair of same direction: a  $*$ -pair  $P$  is contained in a maximal  $*$ -pair  $M$  of same direction and there is no maximal  $*$ -pair  $M'$  of opposite direction such that  $P \subset M' \subset M$ . Let  $M^l$  and  $M^r$  be the maximal  $*$ -pairs immediately containing the left and right

\*-pair of  $x$ . Suppose neither  $M^l$  and  $M^r$  have  $x$  as an end point. Suppose  $M^l \subset M^r$ , then the right \*-pair is not immediately contained in  $M^r$ , a contradiction. The result follows by symmetry.  $\square$

Our approach is to label each extremum in  $F$  with a scale parameter  $\delta$  saying that this extremum is “significant” at scale  $\delta$  and below. Our intuition is that by picking extrema at scale  $\delta$ , we should have a segmentation having error less than  $\delta/2$ .

**Definition 3.6** The scale labeling of an extremum  $x$  is the maximum of the scales of the maximal \*-pairs for which it is an end point.

For example, given the sequence 1, 3, 2, 4 with 2 maximal \*-pairs (1, 4 and 3, 2), we would give the following labels in order 3, 1, 1, 3.

**Definition 3.7** Given  $\delta > 0$ , a *maximal alternating sequence of  $\delta$ -extrema*  $Y = y_1 \dots y_{K+1}$  is a sequence of extrema each having scale label at least  $\delta$ , having alternating types (maximum/minimum), and such that there exists no sequence properly containing  $Y$  having these same properties. From  $Y$  we define a *maximal alternating  $\delta$ -segmentation* of  $D$  by segmenting at the points  $x_1, y_2 \dots y_K, x_n$ .

**THEOREM 3.8** *Given  $\delta > 0$ , let  $P = S_1 \dots S_K$  be a maximal alternating  $\delta$ -segmentation derived from maximal alternating sequence  $y_1 \dots y_{K+1}$  of  $\delta$ -extrema. Then any alternating segmentation  $Q$  having  $OMAFE(Q) < OMAFE(P)$  has at least  $K + 1$  segments.*

*Proof* We show that conditions A and B of Proposition 3.1 are satisfied with  $\epsilon = OMAFE(P)$ .

First we show that each segment  $S_i$  is  $\delta$ -monotone; from this we conclude that  $OMAFE(P) < \delta/2$ . Intervals  $[x_1, y_1]$  and  $[y_K, x_n]$  contain no maximal \*-pairs of scale  $\delta$  or larger, and therefore contain no \*-pairs of scale  $\delta$  or larger. Similarly, no  $[y_i, y_{i+1}]$  contains an opposite-direction significant \*-pair.

Condition A: Follows from  $\delta$ -monotonicity of each  $S_i$  and maximal \*-pairs not overlapping.

Condition B: We show that  $|F(y_{i+1}) - F(y_i)| \geq \delta > 2 \times OMAFE(P)$ . If  $i = 1$ , then  $y_i$  must begin an maximal \*-pair, and the maximal \*-pair must end with  $y_{i+1}$  since maximal \*-pairs cannot overlap. The case  $i + 1 = k$  is similar. Otherwise, since maximal \*-pairs cannot overlap, each  $y_i, y_{i+1}$  is either a maximal \*-pair of scale  $\delta$  or larger or there exist indices  $j$  and  $k$ ,  $j < i$  and  $k > i + 1$  such that

$y_j, y_i$  is a maximal  $*$ -pair of scale at least  $\delta$ , and  $y_{i+1}, y_k$  is a maximal  $*$ -pair of scale at least  $\delta$ . These two maximal  $*$ -pairs have the same direction, and that this is opposite to the direction of  $[y_i, y_{i+1}]$ . Now suppose  $|F(y_i) - F(y_{i+1})| < \delta$ . Then  $y_j, y_k$  is a  $*$ -pair properly containing  $y_j, y_i$  and  $y_{i+1}, y_k$ . But neither  $y_j, y_i$  nor  $y_{i+1}, y_k$  can be properly contained in a  $*$ -pair of opposite direction lying within  $y_j, y_k$ , thus contradicting their maximality and proving the claim.  $\square$

Sequences of extrema labeled at least  $\delta$  are generally not maximal alternating. For example the sequence 0, 10, 9, 10, 0 is scale labeled 10, 10, 1, 10, 10. However, a simple relabeling of certain extrema can make them maximal alternating. Consider two same-sense extrema  $z_1 < z_2$  such that lying between them there exists no extremum having scale at least as large as the minimum of the two extrema's scales. We must have  $F(z_1) = F(z_2)$ , since otherwise the point upon which  $F$  has the lesser value could not be the endpoint of a maximal  $*$ -pair. This is the only situation which causes choice when constructing a maximal alternating sequence of  $\delta$ -extrema. To eliminate this choice, replace the scale label on  $z_1$  with the largest scale of the opposite-sense extrema lying between them. In the next section, Algorithm 1 incorporates this re-labeling making Algorithm 2 simple and efficient.

### 3.1. Computing a Scale Labeling Efficiently

Algorithm 1 (next page) produces a scale labeling in linear time. Extrema from the original data are visited in order, and they alternate (maxima/minima) since we only pick one of the values when there are repeated values (such as 1, 1, 1).

The algorithm has a main loop (lines 5 to 12) where it labels extrema as it identifies extremal  $*$ -pairs, and stack the extrema it cannot immediately label. At all times, the stack (line 3) contains minima and maxima in **strictly** increasing and decreasing order respectively. Also at all times, the last two extrema at the bottom of the stack are the absolute maximum and absolute minimum (found so far). Observe that we can only label an extrema as we find new extremal  $*$ -pairs (lines 7, 10, and 14).

- If the stack is empty or contains only one extremum, we simply add the new extremum (line 12).
- If there are only 2 extrema  $z_1, z_2$  in the stack and we found either a new absolute maximum or new

absolute minimum ( $z_3$ ), we can pop and label the oldest one ( $z_1$ ) (lines 9, 10, and 11) because the old pair ( $z_1, z_2$ ) forms a maximal \*-pair and thus must be bounded by extrema having at least the same scale while the oldest value ( $z_1$ ) does not belong to a larger maximal \*-pair. Otherwise, if there are only 2 extrema  $z_1, z_2$  in the stack and the new extrema  $z_3$  satisfies  $z_3 \in (\min(z_1, z_2), \max(z_1, z_2))$ , then we add it to the stack since no labeling is possible yet.

- While the stack contains more than 2 extrema (lines 6, 7 and 8), we consider the last three points on the stack ( $s_3, s_2, s_1$ ) where  $s_1$  is the last point added. Let  $z$  be the value of the new extrema. If  $z \in (\min(s_1, s_2), \max(s_1, s_2))$ , then it is simply added to the stack since we cannot yet label any of these points; we exit the while loop. Otherwise, we have a new maximum (resp. minimum) exceeding (resp. lower) or matching the previous one on stack, and hence  $s_1, s_2$  is a maximal \*-pair. If  $z \neq s_2$ , then  $s_3, z$  is a maximal \*-pair and thus,  $s_2$  cannot be the end of a maximal \*-pair and  $s_1$  cannot be the beginning of one, hence both  $s_2$  and  $s_1$  are labeled. If  $z = s_2$  then we have successive maxima or minima and the same labeling as  $z \neq s_2$  applies.

During the “unstacking” (lines 13 and following), we visit a sequence of minima and maxima forming increasingly larger maximal \*-pairs.

The algorithm runs in time  $O(n)$  (independent of  $K$ ). Indeed, for any index of an extremum, the condition at line 3.1 will evaluate once to false; moreover the condition at line 3.1 cannot evaluate to true more than  $O(n)$  times.

Once the labeling is complete, we find  $K + 2$  extrema having largest scale in time  $O(nK)$  using  $O(K)$  memory, then we remove all extrema having the same scale as the smallest scale in these  $K + 2$  extrema (removing at least one), we replace the first and the last extrema by 0 and  $n - 1$  respectively (see Algorithm 2). The result is an optimal segmentation having at most  $K$  segments.

Alternatively, if we plan to resegment the time series several times with different values of  $K$ , we can sort all extrema by their label in time  $O(n \log n)$ , and compute in time  $O(n)$  an auxiliary structure on the sorted set so that when selecting the  $i$ th item in the sorted list ( $d_i$ ), we obtain the index  $j$  of the earliest occurrence of this scale in the list ( $\text{scale}(d_j) = \text{scale}(d_i)$  and  $\text{scale}(d_j) < \text{scale}(d_{j-1})$  if  $j > 0$ ) in constant

---

**Algorithm 1** Algorithm to compute the scale labeling in  $O(n)$  time.

---

```

1: INPUT: an array  $d$  containing the  $y$  values indexed from 0 to  $n - 1$ , repeated consecutive values have
   been removed
2: OUTPUT: a scale labeling for all extrema
3:  $S \leftarrow$  empty stack,  $\text{First}(S)$  is the value on top,  $\text{Second}(S)$  is the second value
4: define  $\delta(d, S) = |d_{\text{First}(S)} - d_{\text{Second}(S)}|$ 
5: for  $e$  index of an extremum in  $d$ ,  $e$ 's are visited in increasing order do
6:   while  $\text{length}(S) > 2$  and ( $e$  is a minimum such that  $d_e \leq \text{Second}(S)$  or  $e$  is a maximum such that
    $d_e \geq \text{Second}(S)$ ) do
7:     label  $\text{First}(S)$  and  $\text{Second}(S)$  with  $\delta(d, S)$ 
8:     pop stack  $S$  twice
9:   end while
10:  if  $\text{length}(S)$  is 2 and ( $e$  is a minimum such that  $d_e \leq \text{Second}(S)$  or  $e$  is a maximum such that
    $d_e \geq \text{Second}(S)$ ) then
11:    label  $\text{Second}(S)$  with  $\delta(d, S)$ 
12:    remove  $\text{Second}(S)$  from stack  $S$ 
13:  end if
14:  stack  $e$  to  $S$ 
15: end for
16: while length of  $S > 2$  do
17:   label  $\text{First}(S)$  with  $\delta(d, S)$ 
18:   pop stack  $S$ 
19: end while
20: label  $\text{First}(S)$  and  $\text{Second}(S)$  with  $\delta(d, S)$ 

```

---

**Algorithm 2** Given the scale labeling, this algorithm will return a segmentation using at most  $K$  segments. It is assumed that there are at least  $K + 1$  extrema to begin with.

---

```

INPUT: an array  $d$  containing the  $y$  values indexed from 1 to  $n$ 
INPUT:  $K$  a bound on the number of segments desired
OUTPUT: unsorted segmentation points (a  $\delta$ -segmentation)
 $L \leftarrow$  empty array (capacity  $K + 3$ )
for  $e$  is index of an extremum in  $d$  having scale  $\delta$ ,  $e$  are visited in increasing order do
  insert  $(e, \delta)$  in  $L$  so that  $L$  is sorted by scale in decreasing order (sort on  $\delta$ ) using binary search
  if length of  $L$  is  $K + 3$  then
    pop last( $L$ )
  end if
end for
remove all elements of  $L$  having the scale of last( $L$ )
RETURN: the indexes in  $L$  replacing first one by 1 and last one by  $n$ 

```

---

time. Hence, we can segment any time series optimally in constant time given this precomputation in time  $O(n \log n)$ .

**LEMMA 3.9** *Given a precomputation in time  $O(n \log n)$  using  $O(n)$  storage, for any desired upper bound on the number of segments  $K$ , we can compute the segmentation points of an optimal OMAFE, and the corresponding OMAFE value, in constant time.*

Hence, we can compute an OMAFE versus  $K$  plot in  $O(n \log n)$  time.

#### 4. Experimental Results and Comparison to Piecewise Linear Segmentation Heuristics

We compare our optimal  $O(nK)$  algorithm with our implementations of two piecewise linear segmentation heuristics [3]: top-down, which runs in  $O(nK)$  time (see Algorithm 3), and bottom-up which runs in  $O(n(n - K))$  time (see Algorithm 4). The top-down heuristic successively segments the data starting with only one segment, each time picking the segment with the worse linear regression error and finding the best segmentation point; the linear regression is not continuous from one segment to the other. The regression error can be computed in constant time if one has precomputed the range moments [15, 16]. The bottom-up heuristic starts with intervals containing only one data point and successively merge them, each time choosing the least expensive merge. By maintaining the segments in a doubly-linked list coupled with a heap or tree, it is possible to obtain a bottom-up heuristic with  $O((n - K) \log n)$  complexity, but it then uses much more memory and it is more difficult to implement.

---

##### Algorithm 3 Piecewise Linear Top-Down Segmentation Heuristic.

---

**INPUT:** Time Series  $(x_i, y_i)$  of length  $n$   
**INPUT:** Desired number of segments  $K$   
**INPUT:** Function  $E(p, q)$  computing linear fit error in range  $[x_p, x_q]$   
 $S \leftarrow (1, n, E(0, n))$   
**while**  $|S| < K$  **do**  
  find tuple  $(i, j, \epsilon)$  in  $S$  with maximum last entry  
  find minimum of  $E(i, l) + E(l + 1, j)$  for  $l = i, \dots, j - 1$   
  remove tuple  $(i, j, \epsilon)$  from  $S$   
  insert tuples  $(i, l, E(i, l))$  and  $(l, j, E(l + 1, j))$  in  $S$   
**end while**  
 $S$  contains the segmentation

---



---

##### Algorithm 4 Piecewise Linear Bottom-Up Segmentation Heuristic.

---

**INPUT:** Time Series  $(x_i, y_i)$  of length  $n$   
**INPUT:** Desired number of segments  $K$   
**INPUT:** Function  $E(p, q)$  computing linear fit error in range  $[x_p, x_q]$   
 $S \leftarrow [0, 0], [1, 1], [2, 2], \dots, [n, n]$   
**while**  $|S| > K$  **do**  
  find consecutive intervals in  $S$ ,  $[p_1, p_2]$  and  $[p_2 + 1, p_3]$ , having minimal value  $E(p_1, p_3) - E(p_1, p_2) - E(p_2 + 1, p_3)$   
  merge the two consecutive intervals  
**end while**  
 $S$  contains the segmentation

---

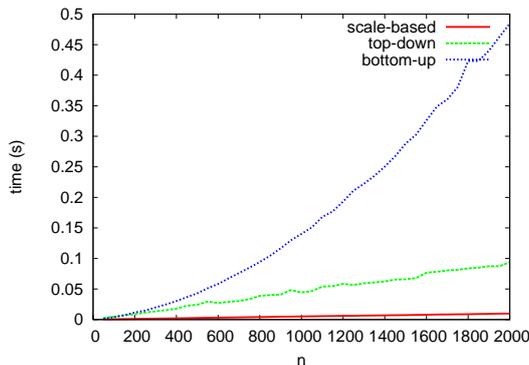


Figure 2. Time to segment a time series of length  $n$  in  $K = 20$  segments.

Once the piecewise linear segmentation is completed, we run through the segments and aggregate consecutive segments having the same sign where the sign of a segment  $[y_k, y_{k+1}]$  is defined by  $F(y_{k+1}) - F(y_k)$ , setting 0 to be a positive sign (increasing monotonicity).

We implemented all algorithms in Python (version 2.5) and ran the experiments on a 2.16 GHz Intel Core 2 Duo processor with sufficient RAM (1 GB). Fig. 2 presents the relative speed of the various segmentation algorithms on time series of various lengths for a fixed number of segments (using randomly generated data). The timings reported include all pre-processing.

#### 4.1. *Electrocardiograms (ECG)*

ECGs have a well known monotonicity structure with 5 commonly identifiable extrema per pulse (reference points P, Q, R, S, and T) (see Fig. 3) though not all points can be easily identified on all pulses and the exact morphology can vary. We used freely available samples from the MIT-BIH Arrhythmia Database [17]. We only present our results over one sample (labeled “100.dat”) since we found that results did not vary much between data samples. These ECG recordings used a sampling rate of 360 Hz per channel with 11-bit resolution (see Fig. 4(a)). We keep the first 4000 samples (11 seconds) and about 14 pulses, and we do no preprocessing such as baseline correction. We can estimate that a typical pulse has about 5 “easily” identifiable monotonic segments. Hence, out of 14 pulses, we can estimate that there are about 70 significant monotonic segments, some of which match the domain-specific markers (reference points P, Q, R, S, and T). A qualitative description of such data is useful for pattern matching applications.

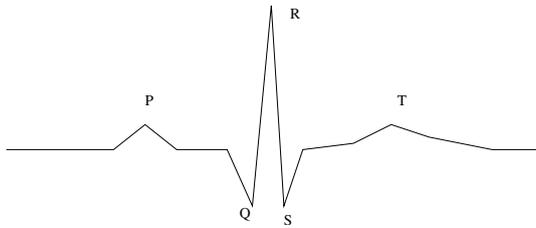


Figure 3. Schema of an ECG pulse with commonly identified reference points (PQRST).

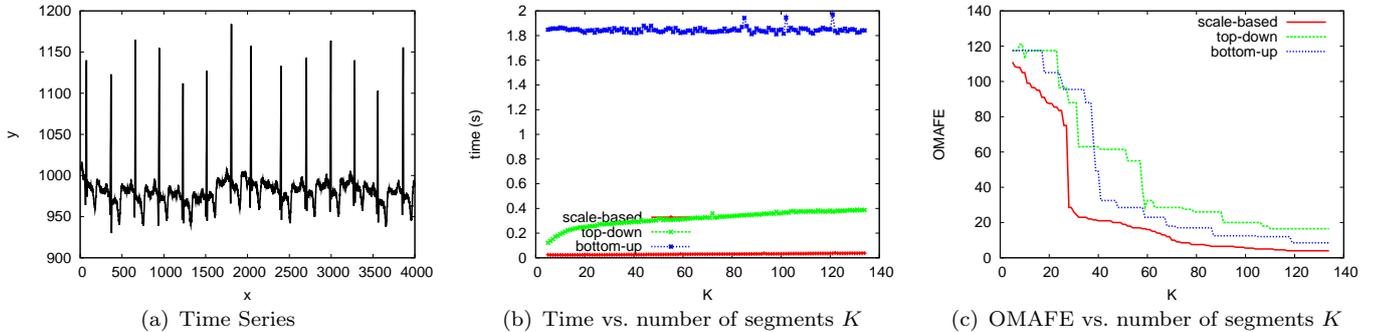


Figure 4. Results of experiments over ECG data.

The running time as a function of  $K$  is presented in Fig. 4(b). The scale-based segmentation implementation is faster than our implementations of the piecewise linear heuristics. On such a long time series (4000 samples), our implementation of the bottom-up heuristic is much slower than the alternatives.

We want to determine how well the piecewise linear segmentation heuristics do comparatively. OMAFE is an absolute and not relative error measure, but because the range of the ECGs under consideration is roughly between 950 and 1150, we expect the OMAFE to never exceed 100 by much. The OMAFE with respect to the maximal number of segments ( $K$ ) is given in Fig. 4(c): it is a “monotonicity spectrum.” By counting on about 5 monotonic segments per pulse with a total of 14 pulses, there should about 70 monotonic segments in the 4000 samples under consideration. We see that the decrease in OMAFE with the addition of new segments starts to level off between 50 and 70 segments as predicted. The addition of new segments past 70 ( $K > 70$ ) has little impact. The scale-based algorithm is optimal, but also at least 3 times more accurate than the top-down algorithm for larger  $K$  and this is consistent over other data sets. In fact, the OMAFE becomes practically zero for  $K > 80$  whereas the OMAFE of the top-down linear regression algorithm remains at around 20, which is still significant. The bottom-up heuristic is more accurate than the top-down heuristic, but it still has about twice the OMAFE for large  $K$ . OMAFE of

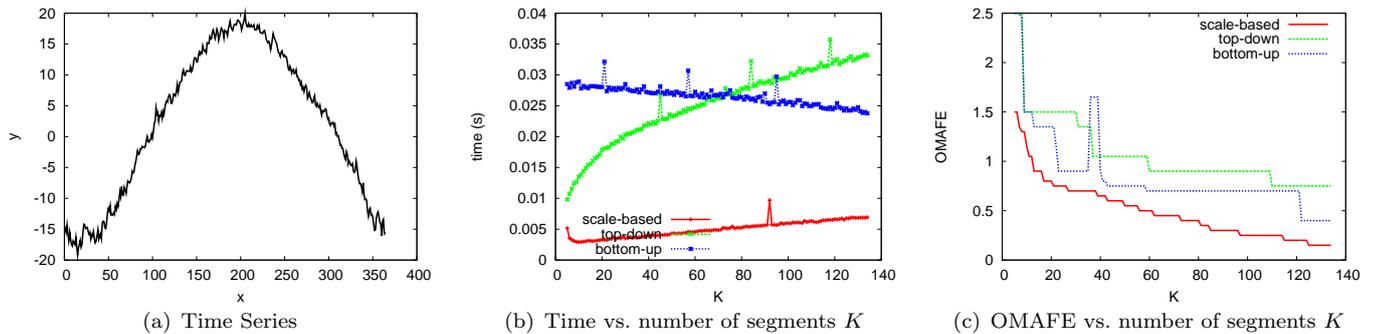


Figure 5. Results of experiments over daily temperature data.

the scale-based algorithm is a non increasing function of  $K$ , a consequence of optimality.

#### 4.2. Temperature Recordings

We consider the daily temperature recordings of the first of 35 weather stations in the MD\*Base Daily temperature data set [18]<sup>1</sup>. Since we only have one year of recordings, only 365 data points are used (see Fig. 5(a)). We also give the running times (see Fig. 5(b)) and the accuracy (see Fig. 5(c)). Our implementation of the bottom-up heuristic is now much faster due to small size of the times series, but the OMAFE, while superior to the top-down heuristic, exhibits a spurious spike near  $K = 40$ , showing the danger of relying on a piecewise linear heuristic to study the monotonicity of a data set. Considering the OMAFE of our scale-based algorithm, we notice that the accuracy increases slowly after  $K = 10$ .

#### 4.3. Synthetic Random Walk Data

Random walks are often used as models for common time series such as stock prices. We generated a random walk  $(i, y_i)_{i=1, \dots, 4000}$  using the formula  $y_{i+1} = y_i + \epsilon$  where  $\epsilon \sim N(0, 1)$  (see Fig. 6(a)). The running times are nearly identical to the ECG case, as is expected since the time series have the same length. However, the OMAFE differs (see Fig. 6(c)): using our optimal algorithm, the curve is smooth with no sharp drop. Meanwhile, the bottom-up heuristic exhibits another spurious spike in the OMAFE (around  $K = 20$ ) while it provides the optimal segmentation at  $K = 5$ .

<sup>1</sup>the data is attributed to Ramsay and Silverman [19]

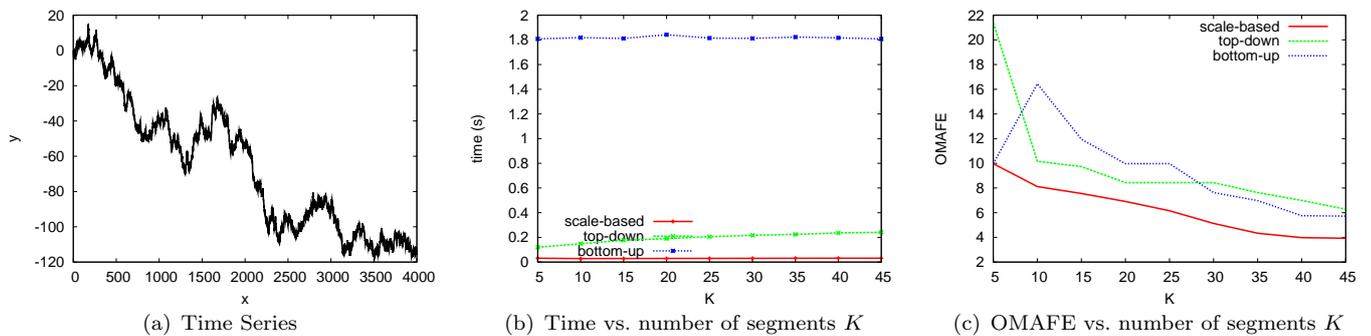


Figure 6. Results of experiments over random walk.

## 5. Conclusion and Future Work

We presented optimal and fast algorithms to compute the best piecewise monotonic segmentation in time  $O(n)$  and the complete OMAFE-versus- $K$  spectrum in time  $O(n \log n)$ . Our experimental results suggest that one should be careful when deriving monotonicity information from piecewise linear segmentation heuristics. Future work will focus on choosing the optimal number of segments for given applications. We also plan to investigate the applications of the monotonicity spectrum as a robust analysis. Further work to integrate flat segments is needed [5, 16].

## References

- [1] D. Lemire, M. Brooks, and Y. Yan, “An optimal linear time algorithm for quasi-monotonic segmentation,” in *ICDM 2005*, 2005.
- [2] E. Bingham, A. Gionis, N. Haiminen, H. Hiisilä, H. Mannila, and E. Terzi, “Segmentation and dimensionality reduction,” in *SDM 2006*, 2006.
- [3] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani, “An online algorithm for segmenting time series,” in *ICDM 2001*, pp. 289–296, 2001.
- [4] N. Haiminen and A. Gionis, “Unimodal segmentation of sequences,” in *ICDM 2004*, 2004.
- [5] M. Brooks, Y. Yan, and D. Lemire, “Scale-based monotonicity analysis in qualitative modelling with flat segments,” in *IJCAI 2005*, 2005.
- [6] W. Fitzgerald, D. Lemire, and M. Brooks, “Quasi-monotonic segmentation of state variable behavior for reactive control,” in *AAAI 2005*, 2005.
- [7] D. G. Galati and M. A. Simaan, “Automatic decomposition of time series into step, ramp, and impulse primitives,” *Pattern Recognition*, vol. 39, pp. 2166–2174, November 2006.
- [8] J. Han, W. Gong, and Y. Yin, “Mining segment-wise periodic patterns in time-related databases,” in *KDD 1998*, 1998.
- [9] E. J. Keogh and M. J. Pazzani, “An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback,” in *KDD 1998*, pp. 239–243, 1998.

- [10] D. Šuc and I. Bratko, “Induction of qualitative tree,” in *ECML 2001*, pp. 442–453, Springer, 2001.
- [11] V. A. Ubhaya, S. E. Weinstein, and Y. Xu, “Best piecewise monotone uniform approximation,” *Approx. Theory*, vol. 63, pp. 375–383, December 1990.
- [12] V. A. Ubhaya, “Isotone optimization I,” *Approx. Theory*, vol. 12, pp. 146–159, 1974.
- [13] M. Brooks, “Approximation complexity for piecewise monotone functions and real data,” *Computers and Mathematics with Applications*, vol. 27, no. 8, 1994.
- [14] H. Edelsbrunner, D. Letscher, and A. Zomorodian, “Topological persistence and simplification,” *Discrete Comp. Geo.*, vol. 28, pp. 511–533, 2002.
- [15] D. Lemire, “Wavelet-based relative prefix sum methods for range sum queries in data cubes,” in *CASCON 2002*, IBM, 2002.
- [16] D. Lemire, “A better alternative to piecewise linear time series segmentation,” in *SDM 2007*, 2007.
- [17] A. L. Goldberger *et al.*, “PhysioBank, PhysioToolkit, and PhysioNet,” *Circulation*, vol. 101, no. 23, pp. 215–220, 2000. <http://www.physionet.org/physiobank/database/mitdb/> – last checked in April 2007.
- [18] Institute for Statistics and Econometrics, “MD\*Base Online,” 2007. <http://www.quantlet.org/mdbase/> – last checked in April 2007.
- [19] J. O. Ramsay and B. W. Silverman, *The Analysis of Functional Data*. Springer, 1997.