A new approach for optimal offline time-series segmentation with error bound guarantee

Ángel Carmona-Poyato[a,*], Nicolás Luis Fernández-Garcia[a], Francisco José Madrid-Cuevas[a], Antonio Manuel Durán-Rosal[b]

[a]*Department of Computing and Numerical Analysis, Maimonides Institute for Biomedical Research (IMIBIC), Rabanales Campus, Albert Einstein building, Third floor, South wing University of Córdoba, Córdoba, 14071, Spain*
[b]*Department of Quantitative Methods, Universidad Loyola Andalucía, c/ Escritor Aguayo, 4, Córdoba, Spain*

Abstract

Piecewise Linear Approximation is one of the most commonly used strategies to represent time series effectively and approximately. This approximation divides the time series into non-overlapping segments and approximates each segment with a straight line. Many suboptimal methods were proposed for this purpose. This paper proposes a new optimal approach, called OSFS, based on feasible space (FS) [1], that minimizes the number of segments of the approximation and guarantees the error bound using the $L\infty$-norm. On the other hand, a new performance measure combined with the OSFS method has been used to evaluate the performance of some suboptimal methods and that of the optimal method that minimizes the holistic approximation error (L^2-norm). The results have shown that the OSFS method is optimal and demonstrates the advantages of $L\infty$-norm over L^2-norm.

Keywords: , Data representation, Optimal Time Series Segmentation, Error Bound Guarantee, $L\infty$-norm

*Corresponding author: Tel.: +34-957-21-21-89; fax: +34 958 -21-83-60;
Email address: ma1capoa@uco.es (Ángel Carmona-Poyato)

1. Introduction

A time-series T_s is a sequence of data points ordered in time such that $T_s = (t_1, t_2, \ldots, t_m)$ where $t_1, t_2, \ldots, t_m$ are individual observations and m is the number of observations in a time series.

Time series have been applied in many areas such as medicine [2], economic [3], telecommunications [4] and online signature verification [5]. Due to the large amount of data used in most application areas, many methods have been proposed to reduce its size without losing relevant information [6].

In the context of this work, segmentation consists of dividing the time series into relevant points, *cut points (CP)*, to reduce its dimensionality by means of a new representation space [7]. The problem of representation is a core issue in pattern recognition, and can dramatically impact the classification performance as well as the computational resources required to solve a particular problem, and the interpretability of the solutions found [8].

For this purpose, one of the most used techniques is called *Piecewise Linear Approximation (PLA)*. PLA divides a time series into segments and uses a linear function to approximate each segment. There are two types of linear approximation [1]: *linear interpolation* that uses the straight line connecting the two endpoints of one segment to represent the data points in the segment and generates continuous piecewise lines; and *linear regression* that uses the regression line to approximate a segment and produces a set of disjointed lines.

Linear interpolation produces a smooth approximation and has low computational complexity because the number of straight lines is much smaller than the number of points of the time series.

The error of the approximation can be calculated by using the L^2-norm and the $L\infty$-norm. The first one is computed from the sum of the squared vertical differences, between the straight line obtained in the segmentation and the real data points, squared. This value is called the *integral squared error* (ISE). The second one is computed from the maximal vertical difference between all the straight lines obtained in the segmentation and the real data points. This value

is called *maximum error* (e_{max}). ISE and e_{max} are calculated as:

$$\text{ISE} = \sum_{i=1}^{n} e_i^2 \qquad\qquad e_{max} = \max_{1 \le i \le n} e_i$$

where e_i is the vertical distance from the real data P_i to its corresponding straight line and n is the number of points of the time series.

Depending on whether the objective is to minimize the error or the amount of information [9], the problem can be addressed by obtaining the best segmentation of a time series using K segments (the holistic approximation error ISE or e_{max} is minimized), or by obtaining the best segmentation of time series such that the maximum error for any segment (e_{max}) or the accumulated error of all segments (ISE) is less than some prefixed threshold (the amount of information used to represent the time series is minimized).

Many works have been proposed to obtain the best segmentation using K segments, where L^2-norm is mostly used. However, there are two main drawbacks when L^2-norm is applied [10]: firstly, this constraint can not generate error-guaranteed representations for streaming data since the stream is naturally unbounded in size; secondly, the L^2-norm is not able to control the approximation error on individual stream data items. To avoid these drawbacks, other methods based on $L\infty$-norm were proposed.

Taking into account how the cut points are obtained, segmentation methods can be categorized into three major groups of approaches [9]: *Sliding Windows*, where the segment increases until a preset error is exceeded and the new segment starts from the last point that does not exceed the preset error; *Top-Down*, where the time series is divided into smaller and smaller segments until a predetermined error is reached; and *Bottom-Up*, where, starting from the largest possible number of segments, these are merged until a predetermined error is exceeded.

Other techniques based on metaheuristics (Genetic Algorithms and particle swarm optimisation) were proposed [11], [12].

Depending on whether the time series is fixed in size or grows dynamically, the methods are classified as *offline* or *online* respectively. Offline methods

segment the complete time series and online methods obtain segments based on the data seen so far. Sarker [13] compared the static with the dynamic segmentation considering if the number of segments is prefixed or not.

All the methods mentioned are suboptimal. Due to its high computational complexity, optimal methods can hardly be used in real-time applications. However, they can be used in order to evaluate the performance of suboptimal methods. An optimal offline method (OSTS) was proposed in [14] which is based on the A^* algorithm, L^2-norm and linear interpolation. Its computational time was greatly reduced by using pruning algorithms. This method was used to obtain the performance of some suboptimal methods based in L^2-norm.

Xie et al. [10] proposed an optimal online method, based on $L\infty$-norm and linear regression. A linear interpolation-based method was also proposed by Xie but is not guaranteed to be optimal.

In this work, we propose a new optimal offline method, called OSFS, based on feasible space (FS) [1] that uses $L\infty$-norm and linear interpolation.

As it mentioned before, Xie [10] showed the main drawbacks of the L^2-norm versus the $L\infty$-norm. Since our proposal is optimal taking into account the $L\infty$-norm, it will be compared to the optimal method (OSTS) proposed in [14], which is based on the L^2-norm.

The present paper is arranged as follows. Section 2 describes the methods that were compared with the proposed method and whose performances were evaluated. Section 3 explains the new proposal. The experiments and results are detailed in Section 4. Finally, the main conclusions are summarized in Section 5.

2. Related work

In this section, some suboptimal time series segmentation methods will be described and their performances will be compared using the OSFS method and the new performance measure. The first group of methods are heuristic and the second one are metaheuristic.

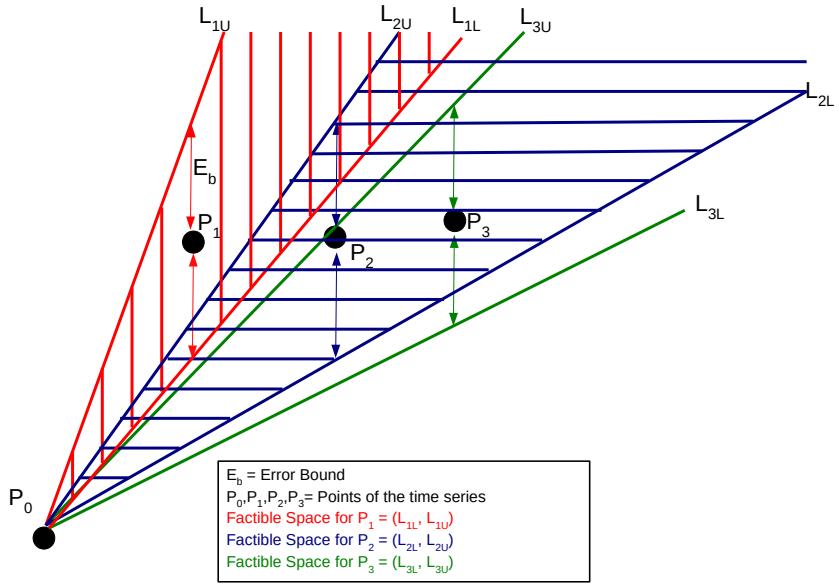

Figure 1: Graphical example of the FS method

2.1. Heuristic methods

Top Down method: this method [15] is an offline method and firstly every possible partitioning of the complete times series is evaluated, and the time series is split at the best location. Then, the approximation errors corresponding to the two resultant parts are calculated. If this error is greater than the user-specified threshold in either part, the algorithm recursively continues to split the subsequences until all the segments have approximation errors below the threshold. In this case, the $L\infty$-norm has been used. In order to select the best location, the one that minimizes the sum of the approximation error of the two resultant parts, was selected. Keogh [9] demonstrated that its computational complexity is $O(n^2)$, where n is the number of points of the time series.

Bottom-Up method: it is the natural complement to the Top-Down method. Firstly, the finest possible approximation of the time series is produced, so that $n-1$ segments are used to approximate the n points of the time

series. In this case, all the approximation errors associated with these segments are zero. Next, the obtained approximation error of merging each pair of adjacent segments is calculated, and, in the next iterations, the method begins to merge the lowest approximation error pair until a stopping criteria is met. In this case, the $L\infty$-norm was used to obtain the approximation error. Keogh [9] demonstrated that its computational complexity is $O(Ln)$, where n is the number of points of the time series and L is the average length of the resulting segments.

Adapted Sarker method: Sarker [16] proposed an improved version of the Bottom-Up method. In this work, this method was adapted from the Bottom-Up method. In order to identify the optimal contiguous segments, the Bottom-up method was adapted by using $n/2, n/3, n/4, n/5 \ldots$ segments to initially approximate the n points of the time series. The Bottom-up method was applied for each initial number of segments and the one that generated the approximation with the fewest points was selected. Since the solution is obtained in a few iterations, it can be considered that its computational complexity is similar to the complexity of the Bottom-Up method.

FSW method: Liu [1] proposed the feasible space window (FSW) online method: the longest segment with an error bound guarantee on each data is obtained. For this purpose is used the feasible space (FS). FS is an area in the data value space of a time series where any straight line in this area can approximate each data point of the corresponding segment within a given error bound. When new data points are read, the FS is incrementally updated and is smaller with the new data points since it is obtained by intersecting areas. When the FS becomes empty, the new segment is generated and the last points that produced a non-empty FS will be the starting point of the next segment. Figure 1 shows an example of updating FS when a data point is read. E_b is the given error bound, and P_0, P_1, P_2, P_3 are points of the time series.

1. When P_1 is read, the FS is the area bounded by lines L_{1L} and L_{1U} (marked with vertical red lines). Since FS is not empty, P_1 is a cut point candidate.

2. When P_2 is read, the area bounded by lines L_{2L} and L_{2U} is obtained (marked with horizontal blue lines). The new FS is obtained by intersecting this area and the current FS (marked with vertical red and horizontal blue lines). Since FS is not empty, P_2 is a cut point candidate.
3. When P_3 is read, the area bounded by lines L_{3L} and L_{3U} (green lines) is obtained. In this case, the intersection of this area and the current FS is empty; therefore, P_3 is not a cut point candidate, and P_2 will be the cut point and the starting point of the next segment.

This method has been adapted as an offline method. Due to this, the algorithm finishes when the last point of time series is included in the FS of a previous point. Its computational complexity is $O(Kn)$, where n is the number of points of the time series and K is the number of segments.

SFSW method: Liu [1] proposed the stepwise feasible space window (SFSW) online method. This method is an improvement of the FSW method. The SFSW method involves backward segmenting one segment to refine the previously obtained endpoint every time we obtain two forward segments using the FSW method. When the backward segmenting point is different to the forward segmenting point, the optimal segmenting point is located between them. In this case, the point between them that produces the minimum error is selected as the real segmenting point. Once the real segmenting point is found, the stepwise process starts from that point. This method has been adapted in a similar way to the FSW method. Its computational complexity is $O(Kn)$, where n is the number of points of the time series and K is the number of segments.

OSTS method: Carmona [14] proposed an optimal offline method (OSTS), by using L^2-norm. This method is based on an improved version of Salotti method [17, 18]. The optimal segmentation is obtained from the search of the shortest path in a graph. For this purpose the A^* algorithm is used. In order to reduce the computational time, the results obtained by the suboptimal Pikaz method [19] were used as pruning values. Its computational complexity is $O(Kn^2)$, where K is the number of segments and n is the number of points

of the time series.

2.2. Metaheuristic methods

In this work, we also compare our method against three evolutionary algorithms which show a good performance in the segmentation of time series [11].

Genetic algorithms (GA or GAs): They are one of the most extended bioinspired metaheuristics. They consist of different processes that simulate the evolution of the species.

Firstly, GA initialises a population of solutions known as chromosomes. Then, GA simulates the evolution for some generations until a stop condition is full-filed: 1) a set of parents is selected to be crossed under a crossover probability in order to generate new solutions (exploitation of the search space), 2) the new solutions are mutated under a mutation probability by performing changes in them (exploration), 3) a fitness function is calculated for the offspring which is related to the problem optimisation, and 4) the competition between parents and offspring to survive for the next generation is made by a replacement operator. Finally, the best solution (chromosome) is returned.

Coral reef optimisation algorithms (CRO or CROs): They are a novel bioinspired strategy to search and optimisation, which simulates the processes occurring in a real coral reef.

Firstly, CRO initialises the coral reef using a set of corals (solutions in our problem), but, in this case, the algorithm maintains some unfilled positions (these positions can be filled during the evolutionary process). Then, CRO simulates the processes of reproduction and reef formation, using different operators: 1) the asexual reproduction takes a percentage of the healthiest corals (with best fitness function) and mutates them under a probability to promote diversity, 2) a fraction of corals are combined between them (external) and the rest are mutated (internal) to mimic the two processes of sexual reproduction, 3) a fitness function is calculated to the new larvae, 4) the new larvae try to settle in the reef competing with the already settled corals, 5) depredation is

performed to delete those less healthy corals under a given probability. Finally, the healthiest coral is returned.

Statistically-driven CRO algorithms (SCRO or SCROs): Statistically-driven CRO algorithms (SCROs) are a modification of the standard CROs where the big amount of parameters needed in them are deleted and dynamically calculated during the evolution. For that, the algorithm performs operations using the fitness distribution of the whole coral reef.

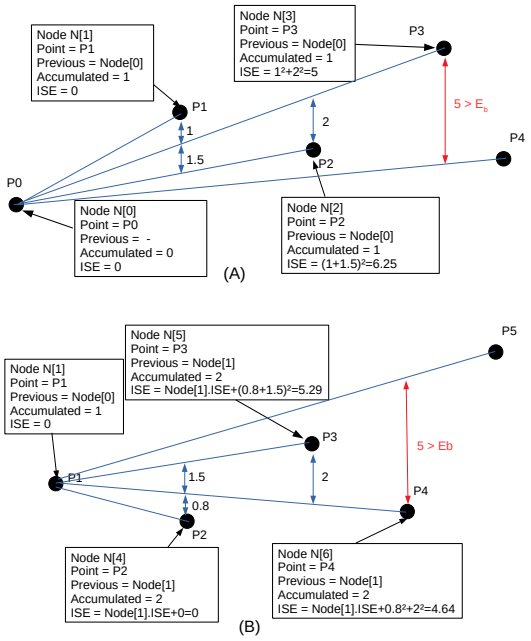

Figure 2: Example of the proposed method (I). (A)Obtaining possible successors to point P0 and updating the priority queue. (B) Obtaining possible successors to point P1 and updating the priority queue.

Adaptation to our problem: these three algorithms presented in [11] were developed to minimise the error of the approximation with a predefined number of points/segments. For this reason, some modifications of the operators have been changed to ensure convergence of the algorithm for the problem presented.

1. Initialisation: each solution is initialised using a suboptimal algorithm

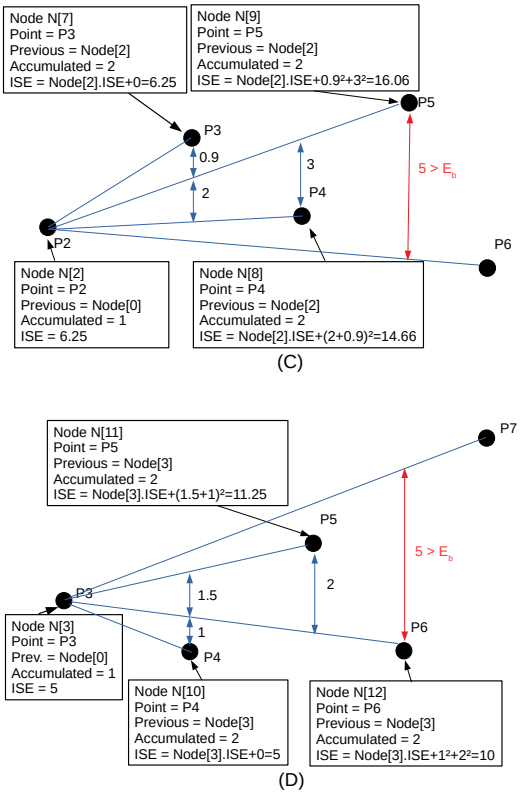

Figure 3: Example of the proposed method (II). (C)Obtaining possible successors to point P2 and updating the priority queue. (D) Obtaining possible successors to point P3 and updating the priority queue.

called *Sliding Window* [20]. This greatly reduces the possibility that initial solutions are not feasible and that the algorithm will only evolve through them.

2. Crossover/External sexual reproduction: given two parents, a single point-crossover is performed to generate two new solutions.
3. Mutation/Internal sexual reproduction: a solution can be mutated adding, deleting, or moving points to the left or the right.
4. Fitness function: given that the work aims to reduce the number of points

in the time series approximation preventing the maximum error ($maxE$) from exceeding a predefined threshold (t), the new fitness function is:

$$f = \begin{cases} 1 - \frac{K}{n} & maxE \leq t \\ \left(1 - \frac{K}{n}\right) * (1 - \alpha) * \left(\frac{1}{1 + \frac{maxE - t}{t}}\right) & maxE > t \end{cases} \quad (1)$$

where K is the number of segments of the approximation, n is the time series length, and α is a penalty parameter which is calculated as the current number of evaluation divided by the maximum number of evaluations (our stop criteria). In this way, non-feasible solutions are allowed at the beginning to explore the search space, but the algorithm tries to exploit the searched feasible solutions in the end. This is in accordance with the philosophy of evolutionary algorithms.

5. Hybridisation: we do not apply any local search to the best solution obtained by the evolutionary algorithm due to the one proposed in the commented work is specifically designed to optimise another fitness function.

3. Our proposal

3.1. Proposed method (OSFS method)

In this work, a new optimal offline segmentation method called *Optimal Segmentation based on Feasible Space* (OSFS) is proposed. This method minimizes the number of segments with error bound guarantee. Therefore, given a maximum error allowed, based on the $L\infty$-norm, the number of cut points (or segments) that approximates the time series must be minimized. Since there can be several solutions, one will be obtained that also minimizes the value of ISE (L^2-norm). The solution to our problem is reduced to finding the shortest path in a directed graph and the feasible space method (FS) proposed by Liu [1] will be used in order to obtain the possible successors of a cut point with error bound guarantee.

Let TS be the time series and e_b the error bound guarantee. Since the shortest path in a graph must be found, it is necessary to define what information

is going to be stored in the nodes of the graph: *Point* depicts a point of the time series; *Previous* depicts the predecessor node of that node in the provisional solution and is necessary in order to obtain the optimal solution; *Accumulated* depicts the number of segments that approximate the series in the provisional solution obtained up to that point; and *ISE* depicts the accumulated value of ISE in the provisional solution obtained up to that point. To refer to each of the items in the node, the operator *dot* will be used. For example, the point corresponding to node N is referred as $N.Point$. The OSFS method is based on a breadth-first search in a directed graph. Candidate nodes are stored in a priority queue (Q_c) The priority queue is sorted according to the value of *Accumulated*. In the case of equality in the value of *Accumulated*, the node with lower value of ISE has priority, thus the ISE value is minimized in the final solution. Table (A_e) will be used to know if the point belonging to a node, with a specific previous point, was already evaluated.

The OSFS method is described in more detail below and its pseudocode is shown in the Algorithm 1.

1. The first node, N_0, of the shortest path will be the node that contains the first point of the time series, P_0 (lines 1 and 2 of the pseudocode). In this case, it has no predecessor and N_0 is used as a predecessor, its accumulated value is 0 and its ISE value is 0.0. $N_0 = (P_0, N_0, 0, 0.0)$ Initially, that will be the only node that is contained in Q_c (line 3 of the pseudocode).
2. Select and extract the node contained in Q_c, in this case the node N_0 (lines 4 and 5 of pseudocode). $minimumN = Q_c.front()$ $Q_c.deque()$ Now an iterative process is followed consisting of the following steps:
    - All the points in the time series between the one corresponding to this node and its previous one will be marked as evaluated and cannot be selected later (line 6 of the pseudocode). So, for each point P_k in the time series, between $minimunN.Previous.Point$ and $minimunN.Point$, $A_e(minimumN.Point, P_k) = true$
    - All the possible successor points of the point corresponding to se-

lected node are obtained by using the feasible space (FS) proposed by Liu [1]. This selection of points works as a pruning procedure and greatly reduces the search space (line 7 of the pseudocode)

$$\{P_1, P_2, \dots P_i, \dots P_n\} = FS(TS, minimumN.Point, e_b)$$

- For each $P_i \in FS$, a new candidate node ($newN$) will be generated, where:

  $newN.Point = P_i$, $newN.Previous = minimumN$

  $newN.Accumulated = 1 + minimumN.Accumulated$

  $newN.ISE = minimumN.ISE + ISE(minimumN.Point, newN.Point)$.

  The last addition represents the ISE value between the two points.

- To treat the new node, two cases arise:

  No node of the graph contains P_i. In this case, this node is inserted in the candidate queue. (line 8 of the pseudocode, $Q_c.enque(newN)$)

  There is a node N of the graph that contains P_i. If the accumulated value is less than it previously had, the node will be updated with the new accumulated value, the new previous and the new ISE is calculated as the ISE value between $N.Point$ and P_i (line 9 of the pseudocode, $Q_c.update(newN)$) In any other case, it is not updated (line 10 of the pseudocode).

- Select and extract the node contained in Q_c with minimum value of $Accumulated$ ($minimumN$) (lines 11 and 12 of the pseudocode). ($minimumN = Q_c.front()$, $Q_c.deque()$) In the event that there are several minimums, the one with the lowest ISE value will be selected.

- If the selected node corresponds to the last point of the time series, the algorithm finishes and the optimal segmentation has been obtained. This is the exit condition of the iterative process.

3. Once the last point in the time series is selected, the iterative process ends. To obtain the cut points, all the previous points are obtained from

the last point until reaching the first of the time series (lines 13 and 14 of the pseudocode).

Figures 2 and 3 shows an example of how the OSFS method works for a time series of 7 points. An error bound guarantee equal to 5 will be considered($e_b = 5$). The double vertical arrows depict the errors of each point and cannot exceed the error bound guarantee. The values of each node are represented in boxes. The four steps represented in the figure can be summarized as follows:

1. First step (A).

   The first node that is initially in the priority queue ($N[0]$), and containing point P_0, is removed from the queue.

   Points P_1, P_2 and P_3, possible successors of P_0, will generate their corresponding nodes that will be inserted in the priority queue. P_4 overpasses the value of e_b and is not inserted into the queue.

   The priority queue would have the following nodes: $Q_c = \{N[1], N[2], N[3]\}$. The accumulated value for these nodes is 1.

2. Second step (B).

   Node $N[1]$, which contains point P_1, is removed from the queue.

   Points P_2, P_3 and P_4, possible successors of P_1, will generate their corresponding nodes ($N[4], N[5], N[6]$). The accumulated value for these nodes is 2. P_5 overpasses the value of e_b and is not inserted into the queue.

   $N[4]$ and $N[5]$ are not inserted in the queue because the accumulated values are higher than that of points P_2 and P_3 from the previous step.

   Node $N[6]$, which contains the point P_4, is inserted as a new node in the queue. The accumulated value for this node is 2.

   The priority queue would have the following nodes: $Q_c = \{N[2], N[3], N[6]\}$.

Algorithm 1 Pseudocode for the OSFS method

Input: Time series (TS). Error bound guarantee (e_b).

Output: List of Cut Points of the optimal segmentation (L_{cp})

Local Data Structures: Priority queue of candidate nodes (Q_c). Two-dimensional array of evaluated nodes (A_e). Current Point (P). Initial Node (N_0).

Begin-Pseudocode

1. $P_0 = TS.FirstPoint()$
2. $N_0 = (P_0, N_0, 0, 0.0)$
3. $Q_c.enque(N_0)$
4. $minimumN = Q_c.front()$
5. $Q_c.deque()$

repeat

  for $P_j = minimumN.Previous.Point, \ldots, minimumN.Point - 1$ **do**

    6. $A_e(minimumN.Point, P_j) = true$ {minimumN.Point will no longer be selected}

  end for

  $P = minimumN.Point$

  for P_i in $FS(TS, P, e_b)$ and $A_e(P_i, minimumN.Point) \neq true$ **do**

    {For each $P_i \in FS$ not previously selected, a new node N_i is created}

    $7. N_i = (P_i, 1 + minimumN.Accumulated, minimumN, minimumN.ISE + ISE(P, P_i))$

    if $\forall N \in Q_c \Rightarrow P_i \neq N.Point$ {P_i is not contained in any node of Q_c} **then**

      8. $Q_c.enque(N_i)$

    else if $N_i.Accumulated \leq N_i.oldAccumulated$ {old value of Accumulated} **then**

      9. $Q_c.update(N_i)$ {N_i is updated with new Accumulated and Previous}

    else

      10. {N_i is not updated}

    end if

  end for

  11. $minimumN \leftarrow Q_c.front()()$
  12. $Q_c.deque()$

until $minimumN.Point = TS.LastPoint()$

{Now, the cut points are inserted in L_{cp}}

while $minimumN.Point \neq P_0$ **do**

  13. $L_{cp}.insert(minimumN.Point)$
  14. $minimumN \leftarrow minimumN.Previous$

end while

End-Pseudocode

3. Third step (C).

Node $N[2]$, which contains point P_2, is removed from the queue.

Points P_3, P_4 and P_5, possible successors of P_2, will generate their corresponding nodes ($N[7], N[8], N[9]$). The accumulated value for these nodes is 2. P_6 overpasses the value of e_b and is not inserted into the queue.

$N[7]$, which contains the point P_3, is not inserted in the queue because the accumulated value is higher than that of point P_3 from the first step (Node $N[3]$).

Node $N[8]$, which contains the point P_4, has a accumulated value equal to that of node $N[6]$, which also contains point P_4, but the ISE value for node $N[6]$ is lower. Therefore, node $N[8]$ will not be queued.

Node $N[9]$, which contains the point P_5, is inserted as a new node in the queue. The accumulated value for this node is 2.

The priority queue would have the following nodes: $Q_c = \{N[3], N[6], N[9]\}$.

4. Fourth step (D).

Node $N[3]$, which contains point P_3, is removed from the queue.

Points P_4, P_5 and P_6, possible successors of P_3, will generate their corresponding nodes ($N[10], N[11], N[12]$). The accumulated value for these nodes is 2. P_7 overpasses the value of e_b and is not inserted into the queue.

Node $N[10]$, which contains the point P_4, has a accumulated value equal to that of node $N[6]$, which also contains point P_4, but the ISE value for node $N[6]$ is lower. Therefore, node $N[10]$ will not be queued.

Node $N[11]$, which contains the point P_5, has a accumulated value equal to that of node $N[9]$, which also contains point P_5, but the

ISE value for node $N[11]$ is lower. Therefore, node $N[11]$ will replace node $N[9$ in the queue.

Node $N[12]$, which contains the point P_6, is inserted as new node in the queue. The accumulated value for this node is 2.

The priority queue would have the following nodes: $Q_c = \{N[6], N[11], N[12]\}$.

5. Finally, the next node to come out of the queue will be $N[6]$, which contains point $P4$. Points P_5, P_6 and P_7, possible successors of P_3 will generate their corresponding nodes. In this case, when the last point of the time series (P7) is selected, the algorithm ends. The solution will be obtained from the previous value of the Node that contains $P7$ and so on. Therefore, the optimal cut points will be $P7$, $P4$, $P1$ and $P0$.

3.2. Proposed performance measure

Any segmentation method tries to minimize the number of segments (amount of information) and the error made in it. These objectives are opposed and when they try to improve one of them, the other method is worsened. To avoid this drawback, Rosin [21] proposed a balanced measure to compare polygonal approximations of closed contours. This measure had two main advantages: 1) it can be used to compare polygonal approximations with different numbers of points; 2) it allows to compare the polygonal approximation obtained by a given method with the optimal polygonal approximation.

This measure used two components: fidelity and efficiency. Fidelity measures how well the polygonal approximation obtained by the method to be tested fits the optimal polygonal segmentation in terms of the approximation error. Efficiency measures how compact the polygonal approximation obtained by the method to be tested is, relative to the optimal polygonal approximation which incurs the same error. They are defined as

$$Fidelity = \frac{E_{opt}}{E_{approx}} * 100 \qquad Efficiency = \frac{N_{opt}}{N_{approx}} * 100$$

where E_{approx} is the error that the tested method would require to produce the same number of cut points as the optimal method, and E_{opt} is the error

incurred by the optimal method. N_{approx} is the number of cut points in the segmentation produced by the method to be tested, and N_{opt} is the number of cut points that the optimal method would require to produce the same error as the tested method. From these two measures and combining the properties of both, Rosin proposed the measure of Merit. $Merit = \sqrt{Fidelity \times Efficiency}$

In this work, the Rosin measure has been adapted to assess the merit of any segmentation method. In this case, the Rosin measure has been used because it assesses the performance of each of the two optimal methods (OSTS and OSFS) using the recommended measure of merit for the other method. Thus, when the performance of one method is evaluated, the measure that optimizes the other method is taken into account. On the other hand, it should be noted that this measure also evaluates the performance of the suboptimal methods.

To obtain these measurements in a suboptimal method it is required to calculate the number of cut points (N_{approx}) and the maximum error (E_{approx}) for that method, how many points the optimal method would require to produce the same maximum error as the suboptimal method N_{opt}, and the maximum error (E_{opt}) that the optimal method would obtain if it had the same cut points as the suboptimal method (N_{approx}).

In our case, the error incurred is the one corresponding to the $L\infty$-norm.

3.3. Complexity Analysis

Since the OSFS method is based on the FSW method, the computational complexity of this method is analyzed. Liu [1] simplified this analysis by assuming a fixed number of segments K and n points. In the worst case, to obtain the feasible space (FS), every time the next cut point is searched, the last data point of time series is reached, which implies that the complexity of finding one cut point is $O(n)$. Since it is assumed that there are K segments, the computational complexity of the FSW algorithm is $O(Kn)$ in the worst case. However, in the average case every segment contains n/K points. In order to obtain the feasible space (FS), n/K points are processed until the next cut point is obtained. Therefore, the computational complexity of the FSW algorithm is $O(n)$ in the

average case.

To analyze the computational complexity of the OSFS method, the average case will be assumed, which is much more realistic than the worst case. Thus, the following considerations will be taken into account:

1. The feasible space is calculated for all the points of the time series, which implies an order of complexity $O(n^2)$.
2. All the points that are processed to obtain the feasible spaces create nodes that go in and out of the priority queue as the algorithm selects the cut points of the optimal segmentation.
3. Since the algorithm performs a breadth-first search on a directed graph and since its nodes enter and exit in the priority queue, only nodes corresponding to two consecutive levels of depth can belong to the priority queue in each step of the algorithm. Thus, for the previous level of depth, the priority queue will contain on average n/K nodes, and for each of those nodes it will contain n/K nodes at the next level of depth. Therefore, on average, the number of nodes stored in the priority queue will be $(n/K)^2$.
4. On the other hand, and taking into account that the priority queue was implemented as a heap, the operation to obtain the minimum is of constant complexity and the operations to insert, delete or update are of logarithmic complexity. Therefore, the computational complexity of these operations would be $O(log((n/K)^2) \equiv O(log(n/K))$

Finally, taking into account the order of complexity of obtaining the feasible spaces and the processing of the nodes in the priority queue, the computational complexity of the OSFS method is $O(n^2 log(n/K))$.

4. Experiments and results

This section shows the time series considered to evaluate the different methods, the experimental setting and the results obtained. The OSFS method has been implemented in C++ and all the experiments were run using an Intel(R)

Core(TM) i7-870 K CPU at 3.70GHz with 64GB of RAM. The code and the time series used are available at https://github.com/ma1capoa/OSFS_Method

4.1. Datasets used

The performance of the OSFS method has been evaluated and compared to other methods whose performance has been evaluated as well. For this purpose, several synthetic and real-world time series collected from public repositories have been used. So, its robustness in different scopes of application has been tested: (1) three datasets from the *UCR Time Series Classification Archive* were selected [22]: Hand Outlines, with a total of 8127 points, Mallat, and StarLightCurves, all of them with 8192 observations; (2) *Donoho-Johnstone time series* [23, 24], formed by four functions to which random noise can be added to produce an infinite number of datasets. In this work, we have considered the function Blocks with medium noise, producing a total of 2048 observations; (3) *Stock prices time series from financial applications* including five different indexes: BBVA, Deutsche Bank, Intesa San Paolo, and Société Genéralé. These four series have a length of 4174 points, considering daily values from 1 January 1999 to 9 February 2015; (4) *Wave height time series* (Hs) including four time series of significant wave height collected from buoys of the National Data Buoy Center of the USA have been used [25]. Two buoys are collecting data in the Gulf of Alaska (with registration number 46001 and 46075), and the rest are from Puerto Rico (41043 and 41044). One value every six hours from 1st January 2008 to 31st December 2013 is considered for buoy 46001 (8767 observations), while data from 1st January 2011 to 31st December 2015 are considered for the rest of buoys (7303 observations for each one); (5) *Arrhythmia dataset* contains cardiology data which belongs to the PhysioBank ATM of the MIT BIH Arrhythmia dataset [26]. We used the MLII signal of the record 108 (9000 observations) to test the algorithm in this dataset.

Some representative series are shown in Figure 4.

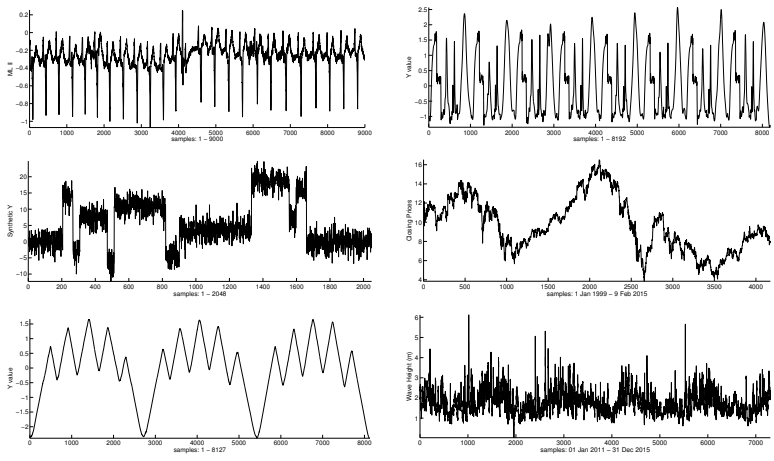

Figure 4: Some representatives time series. From left to right and from top to bottom: Arrhythmia, Mallat, Donoho-Johnstone, BBVA, HandOutlines and buoy-41043

4.2. Performance measures

In order to assess the effectiveness of the OSFS method, three measures were used: (1) the number of cut points or segments obtained in the segmentation of the time series; (2) the root mean square of ISE value divided by the number of points of the time series, called *RSME*, calculated as: $\text{RSME} = \sqrt{\frac{\text{ISE}}{n}}$; and (3) the measure of merit cited in subsection 3.2. The number of cut points is the main measure since it has been used as an objective function to optimize the method. On the other hand, *RSME* is the error considered in the L^2-norm.

4.3. First experiment. Comparison of optimal methods

In this subsection, the OSFS method is compared to the optimal method proposed in [14]. Since the two methods are optimal, when the Rosin measurement is calculated for each of them, taking into account their respective optimization criteria, a value of 100 is obtained for both. Due to this, when comparing these methods, the Rosin measure of merit was calculated using, for each of the methods, the optimization criterion of the other one. Thus, the measures of merit for the OSTS method and for the other one were calculated

using the RSME value (L^2-norm), and the number of cut points ($L\infty$-norm), respectively.

To test both methods under similar conditions, the following steps were followed:

1. By using OSTS method, and for all series, the optimal segmentation with $0.025n$ cut points were calculated, where n is the number of points in the original series, just like in the OSTS method [14].
2. For each of the optimal segmentations obtained in the previous step, the maximum error value was calculated ($L\infty$-norm).
3. Using the maximum error values obtained as the error bound guarantee in each of the time series, the OSFS method was tested.

These maximum error values will be used as error bounds guarantee in the next experiments as well.

Tables 1 shows the computational time, merit, number of cup points and RSME values for the two optimal methods. The best results for each time series are highlighted in bold.

Table 1: Values of error bound guarantee (e_b), time (ms), Merit, number of cut points (N_{cp} and RMSE for the two optimal methods.

	error bound	OSFS method				OSTS method			
Time series	e_b	time(ms)	Merit	N_{cp}	RSME	time(ms)	Merit	N_{cp}	RSME
Hand Outlines	0.018	**3946.4**	**79.1**	**126**	0.007	12146.9	60.3	203	**0.004**
Mallat	0.259	**3852.5**	**85.7**	**176**	0.105	10489.2	81.2	204	**0.072**
StarLightCurves	0.038	**3882.0**	**91.2**	**166**	0.017	20050.4	75.5	204	**0.011**
Donoho-Johnstone	8.405	**285.5**	**99.8**	**22**	2.651	734.8	56.3	51	**2.218**
BBVA	1.324	**1098.0**	**88.4**	**29**	0.520	3672.9	38.9	104	**0.236**
DEUTSCHE	7.467	**1087.4**	**90.2**	**40**	2.581	2953.8	46.9	104	**1.421**
SAN PAOLO	0.527	**1138.9**	**73.5**	**26**	0.205	3708.7	33.6	104	**0.080**
SO Generálé	8.830	**1101.7**	**88.6**	**33**	3.034	3870.0	41.3	104	**1.598**
B46001	4.914	**5233.2**	**24.5**	**4**	2.452	64818.1	9.3	219	**0.799**
B46075	3.951	**3404.3**	20.9	**31**	1.705	30415.3	**32.3**	182	**0.822**
B41043	1.752	**3347.6**	**82.4**	**17**	0.742	29851.7	21.6	182	**0.295**
B41044	1.465	**3207.0**	**69.2**	**40**	0.526	23399.0	36.4	182	**0.292**
Arrhytmia	0.115	**4699.0**	**79.5**	**180**	0.038	15938.7	75.6	225	**0.022**
Mean Values	3.005	**2791.0**	**74.8**	**68.46**	1.122	17080.7	46.9	159.08	**0.605**

Obviously, the results show that each one of them obtains better results in the parameter they optimize (N_{cp} and RSME). However, the computational time of the OSFS method is approximately one sixth of that of the OSTS method. On the other hand, the average value of the measure of merit obtained in the OSTS method is 74.8 while that of the OSTS method is 46.9. Only in the case of the time series corresponding to buoy-B46075, the measure of merit is better for the OSTS method; in the rest of cases, the proposed method is better.

4.4. Second experiment. Comparison with heuristic methods and assessment of their performance

In this subsection, the OSFS method is compared to the heuristic methods described in section 2 and the performance of these methods is also evaluated.

In this case, the same values of error bound guarantee (e_b) of the previous experiment were used. Table 2 shows the computational time and merit for the heuristic methods. The best results for each time series are highlighted in bold (the optimal method was not taken into account for the merit measure).

Table 2: Values of time (ms) and merit for the heuristic methods. $L\infty$-norm has been used. Error bound guarantee values are the same as in Table 1

	OSFS		Top Down		Bottom Up		Sarker		FSW		SFSW	
Time series	time(ms)	Merit	time(ms)	Merit	time(ms)	Merit	time(ms)	Merit	time(ms)	Merit	time(ms)	Merit
Hand Outlines	3946.43	100.0	2969.9	56.1	1180.1	62.6	1223.3	62.6	**1.4**	94.5	5.6	**95.4**
Mallat	3852.45	100.0	8110.5	35.5	1230.4	72.3	1929.2	72.3	**1.4**	**97.9**	5.1	97.9
StarLightCurves	3881.97	100.0	4744.6	40.3	1024.6	71.3	1040.0	71.3	**2.4**	95.0	7.2	**95.5**
Donoho-Johnstone	285.50	100.0	108.5	36.9	39.4	22.7	121.2	22.7	**0.4**	**95.3**	1.9	95.2
BBVA	1098.04	100.0	449.8	51.6	412.6	54.7	876.2	57.1	**0.8**	**90.9**	4.0	87.6
DEUTSCHE	1087.36	100.0	531.5	45.8	370.2	43.8	736.1	59.3	**0.8**	91.5	6.5	**94.8**
SAN PAOLO	1138.87	100.0	650.3	43.8	434.4	57.6	1410.6	64.4	**0.7**	82.3	10.2	**86.9**
SO Genéralé	1101.74	100.0	421.9	49.6	336.1	27.6	927.3	54.1	**0.7**	**87.1**	6.2	87.1
B46001	5233.24	100.0	2986.0	18.5	1973.3	21.2	4828.6	27.4	**1.4**	71.1	18.3	**93.6**
B46075	3404.27	100.0	2550.3	32.8	1162.6	32.6	1280.9	32.6	**1.3**	**83.1**	25.4	61.3
B41043	3347.55	100.0	2543.8	32.7	1311.5	40.2	2206.4	41.0	**1.2**	**90.0**	26.1	74.6
B41044	3207.00	100.0	2972.4	40.3	1156.4	41.7	1144.2	41.7	**1.4**	**91.6**	15.0	79.7
Arrhytmia	4698.98	100.0	4058.1	41.3	1391.7	67.5	2699.0	73.7	**1.7**	93.8	8.5	**96.2**
Mean Values	2791.03	100.00	2545.96	40.40	924.86	47.37	1571.00	52.32	**1.19**	**89.53**	10.76	88.14

From the point of view of computational time, the FSW method is the best

and the SFSW method is the second best. Its difference from the other methods is significant. As seen in section 2, the low values of computational time are due to the fact that the computational complexity of these methods is linear. The worst one is the OSFS method, however, its time is not much higher than some of the proven heuristic methods (for example: it is only 10% higher than the top down method). It cannot be forgotten that the OSFS method is optimal.

From the point of view of the merit measure, without considering the OSFS method that is optimal and therefore obtains a merit equal to 100, the best two are the FSW and SFSW methods with no significant difference between them. The worst of the methods by far is the Top Down method.

Taking into account their low computational complexity and that they obtain a high value of the measure of merit, the FSW or SFSW methods would be the two recommended methods for any real-time application.

Table 3 shows the number of cut points (N_{cp}) and RSME values for the heuristic methods (for example: it is only 10% higher than the top down method). The best results for each time series are highlighted in bold (the OSFS method has not been taken into account for the N_{cp} values).

Table 3: Values of number of cut points(N_{cp}) and RSME for the heuristic methods.$L\infty$-norm was used. Error bound guarantee values are the same as in Table 1

	OSFS		Top Down		Bottom Up		Sarker		FSW		SFSW	
Time series	N_{cp}	RSME	N_{cp}	RSME	N_{cp}	RSME	N_{cp}	RSME	N_{cp}	RSME	N_{cp}	RSME
Hand Outlines	126	0.007	224	0.006	198	**0.004**	198	0.004	133	0.009	**132**	0.007
Mallat	176	0.105	440	0.084	233	**0.076**	233	0.076	**181**	0.145	181	0.110
StarLightCurves	166	0.017	332	0.014	237	**0.010**	237	0.010	172	0.023	**171**	0.018
Donoho-Johnstone	22	2.651	107	**2.524**	118	3.293	118	3.293	**26**	2.753	26	2.589
BBVA	29	0.520	79	0.434	79	**0.319**	69	0.348	**40**	0.579	42	0.483
DEUTSCHE	40	2.581	110	2.532	132	**1.510**	79	1.786	48	3.291	**45**	2.647
SAN PAOLO	26	0.205	78	0.169	57	0.135	54	**0.131**	37	0.237	**34**	0.201
SO Genéralé	33	3.034	85	2.882	233	**1.273**	78	1.953	**41**	3.730	41	3.119
B46001	4	2.452	121	1.136	99	**1.123**	64	1.252	13	2.613	**8**	2.441
B46075	31	1.705	192	1.069	198	0.889	198	**0.889**	**47**	2.046	75	1.694
B41043	17	0.742	126	0.459	92	0.440	89	**0.428**	**27**	0.736	35	0.618
B41044	40	0.526	186	0.415	176	**0.349**	176	0.349	**56**	0.647	69	0.536
Arrhytmia	180	0.038	433	0.034	254	**0.024**	233	0.024	190	0.053	**187**	0.040
Mean Values	68.462	1.122	193.308	0.904	162.000	**0.727**	140.462	0.811	**77.769**	1.297	80.462	1.116

Considering the number of cut points, the FSW method is the best and the SFSW method is the second best. Its difference from the other methods is significant. On average, its values are 13% and 17% higher than the optimal method, respectively. The worst one is the Top Down method.

Taking into account the RSME values, the the Bottom Up method is the best and the second best is the adapted Sarker method, the OSFS method is in an intermediate position despite having the least number of cut points and the worst of the methods by far is the FSW method.

4.5. Third experiment. Comparison with metaheuristic methods and assessment of their performance

In this subsection, the OSFS method is compared to the metaheuristic methods described in section 2 and the performance of these methods is also evaluated. In this case, the same values of error bound guarantee of the previous experiments were used. These algorithms were run 30 times with different random seeds, and Tables show the obtained mean values.

Table 4 shows the computational time (ms), the number of cut points (N_{cp}) and Merit values for the metaheuristic methods. The best results for each time series are highlighted in bold (the OSFS method was not taken into account for the N_{cp} and Merit values)

Note that the OSFS method is optimal and therefore obtains a value of 100 in the measure of merit. Considering the computational time, N_{cp}, and Merit values, the best results are obtained by the OSFS method, the second best results are obtained by the GA method, and the results obtained by *CRO* and *SCRO* methods are similar. Moreover, the computational time of the metaheuristic methods is very high.

However, if the metaheuristic methods are compared to heuristic methods, taking into account the N_{cp} and Merit values, they are only overcome by FSW and SFSW methods.

Table 4: Values of time (ms), number of cut points (N_{cp}) and Merit for the metaheuristic methods.$L\infty$-norm has been used. Error bound guarantee values are the same as in Table 1

	OSTS			GA			CRO			SCRO		
Time series	time(ms)	N_{cp}	Merit	time(ms)	N_{cp}	Merit	time(ms)	N_{cp}	Merit	time(ms)	N_{cp}	Merit
Hand Outlines	**3946.4**	126.0	100.0	95725.9	167.0	76.8	99404.9	**165.6**	**78.0**	97310.5	167.5	77.3
Mallat	**3852.5**	176.0	100.0	121383.2	212.9	**80.4**	125034.6	**212.8**	80.4	123601.1	214.0	79.8
StarLightCurves	**3882.0**	166.0	100.0	108864.1	**185.2**	**84.6**	108856.0	185.3	84.2	108151.9	185.9	83.7
Donoho-Johnstone	**285.5**	22.0	100.0	10261.1	72.2	**48.3**	12610.7	**80.7**	45.1	12313.6	78.9	45.2
BBVA	**1098.0**	29.0	100.0	20587.6	**59.9**	**64.2**	18900.3	62.5	61.2	21016.9	63.2	61.6
DEUTSCHE	**1087.4**	40.0	100.0	23623.4	**67.0**	**69.1**	23000.6	70.5	67.3	23263.9	71.7	66.0
SAN PAOLO	**1138.9**	26.0	100.0	14114.0	**42.6**	**70.3**	15804.3	45.2	68.6	17795.7	45.5	68.4
SO Genéralé	**1101.7**	33.0	100.0	19185.0	**53.4**	**70.4**	18027.3	56.1	67.6	18629.0	56.8	66.3
B46001	**5233.2**	4.0	100.0	29780.8	**13.2**	**55.0**	26550.6	15.7	50.4	49449.3	18.0	47.1
B46075	**3404.3**	31.0	100.0	58655.0	**95.2**	**52.1**	63983.1	117.1	46.2	65204.6	114.2	46.7
B41043	**3347.6**	17.0	100.0	35169.6	**45.5**	**57.1**	38512.2	62.0	48.2	45505.3	57.2	49.8
B41044	**3207.0**	40.0	100.0	58525.1	**100.8**	**57.4**	63008.2	115.5	51.9	65543.3	116.7	51.3
Arrhytmia	**4699.0**	180.0	100.0	136224.9	213.2	80.5	135013.9	**212.1**	29.1	137201.2	212.2	**80.7**
Mean values	**2791.03**	68.46	100.0	56315.37	**102.16**	**66.6**	57592.81	107.77	59.9	60383.56	107.82	63.4

As Table 4 shows, one of the main disadvantages of the evolutionary algorithms is the computational time when they are compared against one-solution based algorithms. As we commented previously, an evolutionary algorithm is applied to a population of solutions and over some generations. It causes that the evaluation of the solution, which is the slowest part in the algorithm, is repeat a lot of times causing the increase of this computational time. Also, given that this problem has an enormous search space with non-feasible solutions, the initialisation is made by repeating another algorithm for each of the solutions of the initial population resulting in a set of feasible solutions.

Another drawback is that evolutionary algorithms can reach high quality areas, but they are not good at optimising the solutions of that area. To solve this problem, authors in [11] proposed an interesting hybridisation using a local search, but it cannot be applied given that the fitness function and the problem here is different.

For future improvements of the evolutionary algorithms, it would be interesting to focus on reducing the computational time and the number of points by finding a good local search to be applied as a hybridisation of the algorithm.

Table 5 shows the values of RSME for the metaheuristic methods. The best

results for each time series are highlighted in bold.

Table 5: Values of RSME for the metaheuristic methods.$L\infty$-norm has been used. Error bound guarantee values are the same as in Table 1

Time series	Proposed	GA	CRO	SCRO
Hand Outlines	**0.007**	0.009	0.009	0.009
Mallat	**0.105**	0.145	0.144	0.145
StarLightCurves	**0.017**	0.023	0.023	0.023
Donoho-Johnstone	**2.651**	2.844	2.853	2.878
BBVA	**0.520**	0.576	0.590	0.591
DEUTSCHE	**2.581**	3.113	3.130	3.146
SAN PAOLO	**0.205**	0.232	0.236	0.235
SO Genéralé	**3.034**	3.845	3.971	3.981
B46001	2.452	2.429	**2.395**	2.466
B46075	**1.705**	1.813	1.872	1.869
B41043	0.742	**0.730**	0.780	0.767
B41044	**0.526**	0.649	0.669	0.671
Arrhytmia	**0.038**	0.054	0.053	0.053
Mean values	**1.122**	1.266	1.286	1.29

Considering the RSME measure, the best results are obtained by the OSFS method, and the results obtained by the metaheuristic methods are similar.

In general, the results of the metaheuristic methods are worse than those of the heuristic methods, taking into account the RSME measure.

4.6. Fourth experiment. How noise affects the OSFS method

For this purpose, the original time series are considered as noise free signals and white noise has been added to achieve versions with SNR rates with values of 20, 30, 35, 40, 45 and 50 decibels. The method was then run taking into account the same error bound guarantee values as in the previous experiments. Table 6 shows the values of N_{cp} for the different noise values.

Except for the HandOutlines and StarLightCurves series, the noise levels of 50, 45 and 40 decibels hardly affect the solution. For 35 decibels or less, the solutions of the MALLAT, BBVA, DEUTSCHE, SAN PAOLO and SO Generale time series are also affected. From 20 decibels all solutions are affected.

Table 6: Values of N_{cp} for different values of noise. $L\infty$-norm was used. Error bound guarantee values are the same as in Table 1

Time series	original	50db	45db	40db	35db	30db	20db
HandOutlines	126	145	186	927	3425	5276	8150
MALLAT	176	176	177	179	182	192	335
StarLightCurves	166	172	185	217	510	2813	6231
Donoho-Johnstone	22	22	22	23	22	22	28
BBVA	29	29	29	30	44	55	1318
DEUTSCHE	40	41	41	43	48	58	1222
SAN PAOLO	26	25	26	27	32	38	723
SO Genéralé	33	31	32	33	38	52	1022
B46001	4	4	4	4	4	4	6
B46075	31	31	31	31	33	33	40
B41043	17	18	18	19	19	20	26
B41044	40	40	39	39	39	41	48
Arrhythmia	180	180	181	191	182	185	207

5. Conclusions and future improvements

The conclusions of this work can be summarized as follows. The present work proposes an optimal time-series segmentation with error bound guarantee ($L\infty$-norm). Taking into account that several optimal solutions are possible, the one that minimizes the RSME value (L^2-norm) is obtained. In order to reduce the computational time by pruning suboptimal solutions, the feasible space method (FS) proposed by Liu [1] is used to obtain the possible successors of a cut point with error bound guarantee. On the other hand, a new performance measure has also been proposed. Thus the OSFS method could be used in any application scenario and its main utility would be the evaluation of the performance of any suboptimal method. Moreover, the experiments carried out have made it possible to compare the norm of $L\infty$-norm and L^2-norm, by using the OSFS method and the one proposed in [14]. Finally, the OSFS method can be used as a benchmark of the performance of some heuristic and metaheuristic methods.

The results show that the computational time of the proposed method is approximately 16 percent of that of the OSTS method and its measure of merit is almost 50% higher than that of the OSTS method. These results demonstrate that the $L\infty$-norm produces better results than the L^2-norm. In some cases,

the computational time of the OSFS method is not much higher than that of some heuristic and suboptimal methods. Considering computational time, number of cut points, and measure of merit, the FSW and SFSW methods are the best performing heuristic methods. Taking into account the error measure RSME, the Bottom Up and Adaptive Sarker methods are the best performing heuristic methods. The computational time of the metaheuristic methods is very high. The results obtained by the metaheuristic methods are similar. When the metaheuristic methods are compared to heuristic methods, taking into account the number of cut points obtained, they are only overcome by FSW and SFSW methods. Overall, the results of metaheuristic methods are worse than heuristic methods, taking into account the error measure RSME. Tests performed to see how noise affects the method show that the smoother the curve, the more it will be affected by noise.

The main drawback of the OSFS method is its computational time. Although the time is close to the time of some heuristic methods, it would not be possible to use it in real-time applications.

The main advantages of the present work can be summarized as follows. The OSFS method provides the optimal solution in a reasonable time that will allow objective evaluation of the performance of suboptimal methods. The new measure of merit can be used to compare the performance of suboptimal methods, even if their solutions have different numbers of segments; it can be used to evaluate the goodness of the different segmentations that a method can generate; and, finally, it can also be used to fine-tune the free parameters of suboptimal methods.

Regarding future improvements, the following could be considered. Look for new pruning methods to try to reduce the computational time of the method. Propose the online version for OSFS method and OSTS method. Adapt the OSFS method to obtain optimal polygonal approximations of object contours, taking into account the $L\infty$-norm.

Acknowledgement

This work has been developed with the support of the Research Projects TIN2016-75279-P and TIN2017-85887-C2-1-P of Spain Ministry of Economy, Industry, and Competitiveness, and FEDER.

References

[1] X. Liu, Z. Lin, H. Wang, Novel online methods for time series segmentation, IEEE Transactions on Knowledge and Data Engineering 20 (12) (2008) 1616–1626. doi:10.1109/TKDE.2008.29.

[2] A. Koski, J. M., M. M., Syntactic recognition of ecg signals by attributed finite automata, Pattern Recognition 28 (1995) 1927–1940. doi:10.1016/0031-3203(95)00052-6.

[3] C.-H. Lee, A. Liu, W.-S. Chen, Pattern discovery of fuzzy time series for financial prediction, Knowledge and Data Engineering, IEEE Transactions on 18 (2006) 613–625. doi:10.1109/TKDE.2006.80.

[4] C. Cortes, K. Fisher, D. Pregibon, A. Rogers, F. Smith, Hancock: A language for extracting signatures from data streams, Proceeding of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2000) 10–19doi:10.1145/347090.347094.

[5] M. Okawa, Time series averaging and local stability weighted dynamic time warping for online signature verification, Pattern Recognition (in press) (2020) 107699. doi:10.1016/j.patcog.2020.107699.

[6] H. Kamalzadeh, A. A., S. Mansour, A shape-based adaptive segmentation of time-series using particle swarm optimization, Information Systems 67 (2017) 1–18. doi:10.1016/j.is.2017.03.004.

[7] A. Aghabozorgi, S.and Shirkhorshidi, T. Wah, Time-series clustering–a decade review, Information Systems 53 (2015) 16–38. doi:10.1016/j.is.2015.04.007.

[8] S. Mauceri, J. Sweeney, J. McDermott, Dissimilarity-based representations for one class classification on time series, Pattern Recognition 100 (2020) 107122. doi:10.1016/j.patcog.2019.107122.

[9] E. Keogh, S. Chu, D. Hart, M. Pazzani, Segmenting time series: a survey and novel approach, Data Mining in Time Series Databases (2004) 1–22doi:10.1142/9789812565402_0001.

[10] Q. Xie, C. Pang, X. Zhou, X. Zhang, K. Deng, Maximum error-bounded piecewise linear representation for online stream approximation, The VLDB Journal 23 (2014) 915–937. doi:10.1007/s00778-014-0355-0.

[11] A. Duran-Rosal, P. Gutierrez-Pena, S. Salcedo-Sanz, C. Hervas-Martinez, A statistically-driven coral reef optimization algorithm for optimal size reduction of time series, Applied Soft Computing 63 (2018) 139–153. doi:10.1016/j.asoc.2017.11.037.

[12] A. Duran-Rosal, P. A. Gutierrez-Pena, A. Carmona-Poyato, C. Hervas-Martinez, A hybrid dynamic exploitation barebones particle swarm optimisation algorithm for time series segmentation, Neurocomputing 353 (2018) 45–55. doi:10.1016/j.neucom.2018.05.129.

[13] I. H. Sarker, Context-aware rule learning from smartphone data: survey, challenges and future directions, Journal of Big Data 6 (2019) 95. doi:10.1186/s40537-019-0258-4.

[14] A. Carmona-Poyato, N. Fernandez-Garcia, F. Madrid-Cuevas, A. Duran-Rosal, A new approach for optimal time-series segmentation, Pattern Recognition Letters 135 (2020) 153–159. doi:10.1016/j.patrec.2020.04.006.

[15] R. Duda, P. Hart, Pattern classification and scene analysis, The Library Quarterly 44 (3) (1974) 258–259. doi:10.1086/620282.

[16] I. H. Sarker, A. Colman, M. A. Kabir, J. Han, Individualized time-series segmentation for mining mobile phone user behavior, The Computer Journal 61 (3) (2017) 349–368. doi:10.1093/comjnl/bxx082.

[17] M. Salotti, Optimal polygonal approximation of digitized curves using the sum of square deviations criterion, Pattern Recognition 35 (2002) 435–443. doi:10.1016/S0031-3203(01)00051-6.

[18] A. Carmona-Poyato, E. Aguilera-Aguilera, F. Madrid-Cuevas, M. Marin-Jimenez, N. Fernandez-Garcia, New method for obtaining optimal polygonal approximations to solve the min-epsilon problem, Neural Computing and Applications 28 (2017) 2383–2394. doi:10.1007/s00521-016-2198-7.

[19] A. Pikaz, I. Dinstein, Optimal polygonal approximation of digital curves, Pattern Recognition 28 (1995) 373–379. doi:10.1016/0031-3203(94)00108-X.

[20] E. Keogh, S. Chu, D. Hart, M. Pazzani, Segmenting Time Series: A Survey and Novel Approach, in: Data Mining In Time Series Databases, Vol. 57 of Series in Machine Perception and Artificial Intelligence, World Scientific Publishing Company, 2004, Ch. 1, pp. 1–22.

[21] P. Rosin, Techniques for assessing polygonal approximations of curves, Pattern Analysis and Machine Intelligence, IEEE Transactions on 19 (1997) 659–666. doi:10.1109/34.601253.

[22] H. A. Dau, E. Keogh, K. Kamgar, C. Yeh, The ucr time series classification archive, https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ (October 2018).

[23] D. L. Donoho, I. M. Johnstone, Ideal spatial adaptation by wavelet shrinkage, Biometrika 81 (3) (1994) 425–455. doi:10.1093/biomet/81.3.425.

[24] D. L. Donoho, I. M. Johnstone, Adapting to unknown smoothness via wavelet shrinkage, Journal of the American Statistical Association 90 (432) (1995) 1200–1224.

[25] NOAA, National buoy data center, http://www.ndbc.noaa.gov/ (2015).

[26] G. B. Moody, R. G. Mark, The impact of the mit-bih arrhythmia database, IEEE Engineering in Medicine and Biology Magazine 20 (3) (2001) 45–50. doi:10.1109/51.932724.