Knowledge and Information Systems (2024) 66:2417–2438
https://doi.org/10.1007/s10115-023-02029-8

REGULAR PAPER

Optimal online time-series segmentation

Ángel Carmona-Poyato[1] · Nicolás-Luis Fernández-García[1] · Francisco-José Madrid-Cuevas[1] · Rafael Muñoz-Salinas[1] · Francisco-José Romero-Ramírez[1]

Received: 22 September 2022 / Revised: 7 July 2023 / Accepted: 22 November 2023 /
Published online: 26 December 2023

Abstract
When time series are processed, the difficulty increases with the size of the series. This fact is aggravated when time series are processed online, since their size increases indefinitely. Therefore, reducing their number of points, without significant loss of information, is an important field of research. This article proposes an optimal online segmentation method, called OSFS-OnL, which guarantees that the number of segments is minimal, that a preset error limit is not exceeded using the $L\infty$-norm, and that for that number of segments the value of the error corresponding to the L^2-norm is minimized. This new proposal has been compared with the optimal OSFS offline segmentation method and has shown better computational performance, regardless of its flexibility to apply it to online or offline segmentation.

Keywords Optimal time series segmentation · Segmentation online versus offline · Error bound guarantee · $L\infty$-norm

Ángel Carmona-Poyato, Nicolás-Luis Fernández-García, Francisco-José Madrid-Cuevas, Rafael Muñoz-Salinas and Francisco-José Romero-Ramírez have contributed equally to this work.

✉ Ángel Carmona-Poyato
ma1capoa@uco.es

Nicolás-Luis Fernández-García
ma1fegan@uco.es

Francisco-José Madrid-Cuevas
ma1macuf@uco.es

Rafael Muñoz-Salinas
in1musar@uco.es

Francisco-José Romero-Ramírez
i32roraf@uco.es

[1] Department of Computing and Numerical Analysis, Maimonides Institute for Biomedical Research (IMIBIC), University of Cordoba, Rabanales Campus, Albert Einstein building, third floor, South wing, 14071 Cordoba, Spain

1 Introduction

Time series T_s are a special kind of data, in which data points are collected chronologically, and ordered in time such that $T_s = (t_0, t_1, \ldots, t_m)$ where $t_0, t_1, \ldots, t_m$ are individual observations and m is the number of observations in a time series.

Time series can be obtained from different areas such as climate [1], hydrology [2], medicine [3], economics [4], online signature verification [5], telecommunications [6], etc. They can be applied to multiple fields depending on the purpose and the area of application [7]. In statistics, econometrics or meteorology, its main task is forecasting [8]. In signal processing, they are used for detection and estimation [9]. In data mining and pattern recognition, they can be used for anomaly detection [10], classification [11], clustering [12] or prediction [13], among others.

In most applications, the amount of data to be processed is huge, raising the need for methods able to reduce the dimensionality of time series without decreasing the amount of information [14]. The application of time series segmentation procedures dividing the time series into relevant points, called *cut points (CP)*, reduces its dimensionality by means of a new representation space. On the other hand, time-series segmentation is also used for discovering useful patterns [15, 16].

To approximate the segments defined by the CP, one of the most common techniques is the *Piecewise Linear Approximation (PLA)*. PLA uses a linear function to approximate each segment, and depending on how to obtain the linear function, there are two methods: (a) *linear interpolation* and (b) *linear regression*. Keogh et al. [17] proposed some methods based on linear interpolations. Fuchs [18] proposed a method to approximate the segments by means of polynomials obtained by least squares.

The norms L^2 and $L\infty$ are commonly used to measure errors in approximate segments. In the first case, the error is obtained by adding the squares of the differences between the points of the time series and the function that approximates it (*integral quadratic error ISE*), and in the second case it is obtained by calculating the maximum difference between the points of the time series and the function that approximates it (*maximum error* e_{max}). In Xie et al. [19] and Carmona-Poyato et al. [20], the advantages of using $L\infty$-norm over L^2-norm were demonstrated.

On the other hand, depending on the static or dynamic nature of the time series, the methods can be *offline* or *online*, respectively. In the first case, the complete time series must be known before applying the method. In the second case, chunk of data are added to the time series as it is segmented. In Sarker [21] are compared the offline and online approaches.

Since the aim is to minimize the amount of information, the problem is to obtain the best time series segmentation such that the error (ISE or e_{max}) is less than some predetermined threshold. In this way, the objective would be to minimize the number of segments or CP, for a predetermined error.

Depending on the optimality of the solution, the methods can be classified as optimal and suboptimal. Since optimal methods are computationally very expensive and they can hardly be used in real-time applications, most of the methods that have been proposed are non-optimal. Xie et al. [19] proposed an optimal online method, based on $L\infty$-norm and linear regression. More recently, two optimal offline methods based on linear interpolation have been proposed [20, 22], which use L^2-norm and $L\infty$-norm, respectively. Despite their difficult application in real time, optimal methods can be used to evaluate the performance of suboptimal ones.

In this work, we propose a new optimal online method, called OSFS-OnL, based on feasible space (FS) [23] that uses $L\infty$-norm and linear interpolation. The OSFS-OnL method outperforms the optimal offline OSFS method, proposed by the authors [20], and its use could even be considered in some real-time applications.

This document is organized as follows. Section 2 describes the methods that were compared with the proposed method. Section 3 explains the new proposal. The experiments and results are detailed in Sect. 4. Finally, the main conclusions are summarized in Sect. 5.

2 Related work

In this section, the following types of methods will be described.

1. Three suboptimal online methods that will be compared with the proposed method.
2. Two optimal offline methods that will be compared with the proposed method.
3. Other relevant suboptimal and optimal methods.

Three suboptimal online methods and two optimal offline methods are described, which will later be compared with the proposed method.

2.1 Suboptimal online methods

Sliding window method. This method [24] obtains a first segment using the first point of the series as the left endpoint, and the following points are used to obtain the right endpoint, until the segment obtained exceeds the maximum allowed error. In this case, the left endpoint of the first segment is the first point in the series, and the right endpoint is the last point that forms a segment with the left endpoint that does not exceed the allowed error. To obtain the second segment, the process is repeated by taking the right endpoint of the first segment as the starting point of the second segment. The process is repeated until the last point of the time series is reached. 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.

FSW method. Liu et al. [23] proposed the feasible space window (FSW) online method, based on the so-called *feasible space FS*. FS is an area that contains consecutive values of a time series, in such a way that any straight line contained in that area could approximate the consecutive values of the time series without exceeding the maximum allowed error. The FSW method starts by treating the first points of the time series and updates the FS corresponding to these points as it treats them. Since the FS is obtained by intersection of areas, it decreases until it is empty. When this happens, a new approximate segment is generated, and the last point that generated a non-empty FS will be the starting point of the next segment. This process is repeated until reaching the last point of the time series, thus obtaining the approximate segments or the CP of the segmentation. For simplicity, from now on, we will call points of the FS of a given point those points of the time series that are included in its FS. 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 et al. [23] proposed the stepwise feasible space window (SFSW) online method. This method is an improvement of the FSW method. The first segment obtained is similar to that of the FSW method, but from the second segment, the authors try to improve the result obtained by the FSW method. To do this, each time a new segment is obtained; a backward segmentation is performed to refine the starting point of that segment. When the

backward segmenting point is different from the start point, the optimal segmenting point is between them. In this case, the point between them that produces the least error is selected as the real segmentation point. Once the actual segmentation point is found, the step-by-step process starts from that point. Its computational complexity is $O(Kn)$, where n is the number of points in the time series and K is the number of segments.

2.2 Optimal methods

OptimalPLR method. Xie et al. [19] proposed an optimal online method (OptimalPLR), by using $L\infty$-norm. This method obtains segments of maximum length, guaranteeing the fixed error, from two convex hulls that are updated each time a new point of the time series is processed. However, the segments obtained are disconnected and their ends do not have to be points of the time series. The authors propose a modification to obtain connected segments, but it does not guarantee its optimality or that the cutoff points are points of the time series. Its computational complexity is $O(n)$, where n is the number of points of the time series.

OSFS method. Carmona et al. [20] proposed an optimal offline method (OSFS), by using $L\infty$-norm. This method minimizes the number of segments (CP) without exceeding a preset maximum error. Since it is possible that several solutions can be obtained, the one that minimizes the value of ISE (L^2-norm) is obtained. To solve the problem, it was reduced to finding the shortest path in a directed graph. To limit the search space, the FS proposed by Liu [23] was used, in such a way that only those points that guaranteed that the preset error was not exceeded, could be considered ends of a segment. This reduction of the search space allowed to greatly reduce the computational time of the method.

The computational complexity of the OSFS method is $O(n^2 log(n/K))$, where n is the number of points of the time series and K is the number of segments.

2.3 Other relevant methods

PAA method. Keogh et al. [25] proposed segmenting the time series using segments of equal size, obtained by averaging the values of the series points corresponding to these segments. Therefore, the segments are horizontal, and the errors committed are greater than in the PLA methods. For a more reduced representation, for each segment, it stores the average value obtained in each of them. Its computational complexity is $O(Nm)$, where N is the number of segments and m is the number of windows used.

Sax method. Lin et al. [26] proposed an extension of PAA method. First, they used the representation of the PAA method and then used this representation to obtain a discrete string. The main advantage is that they obtained a symbolic distance measure that lower limits the PAA distance measure.

SWAB method. Keogh et al. [27] proposed an online method based on the SW and Bottom-Up methods [17, 24], taking advantage of both. In this case, the number of points that are added is variable and depends on the structure of the incoming data. Its computational complexity is similar to that of Bottom-Up algorithm, that is, $O(Ln)$, where n is the number of points of the time series and L is the average length of the resulting segments.

OSTS method. Carmona et al. [22] proposed an optimal offline method (OSTS), by using L^2-norm, based on an improved version of Salotti method [28]. The optimal segmentation is obtained from the search of the shortest path in a graph. To reduce the computational time,

the suboptimal segmentation obtained by Pikaz method [29] was used as pruning value. 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.

3 Our proposal

3.1 Proposed method (OSFS-OnL method)

In order to describe our proposal, some preliminary considerations are raised in Sect. 3.1.1 and the structure of the graph nodes used in the method is described in Sect. 3.1.2. The description of the algorithm is developed in Sect. 3.1.3. Finally, Sect. 3.1.4 shows an example of the application.

3.1.1 General considerations

The proposed method, called *Optimal On Line Segmentation based on Feasible Space* (OSFS-OnL), tries to solve the drawback of offline optimal methods in applications that use large time series and new chunks of data arrive continuously. In this case, to obtain this optimal result on the fly, the optimal offline method would have to be executed on the complete time series, each time new chunk of data are added to the time series, and therefore the computational cost would be infeasible in time and memory.

The new method minimizes the number of segments, guaranteeing that the error limit is not exceeded, and adapting the optimal solution each time new chunk of data are added to the time series. As the problem is stated, it is possible that there are several solutions. In our proposal, the one that also minimizes the value of ISE (L^2-norm) will be obtained.

The solution to our problem can be reduced to finding the shortest path in a directed graph. In order to reduce the number of successors of each graph node, and therefore the size of the graph, the FS method, proposed by Liu [23], will be used. The use of the FS allows us to analyze as successors of a possible cut point, only those that meet the preset error, which will significantly reduce the size of the graph.

On the other hand, being an online method, it must be taken into account that new chunks of data are continuously added to the time series. Each time this occurs, the new optimal solution obtained will affect the previously calculated optimal solution, since unfortunately the CP of the previous solution may change. This fact implies that the optimal solution at each instant cannot be obtained as the sum of the optimal solutions of each of the chunks of data that have been previously added to the time series.

3.1.2 Structure of the graph nodes

The following information will be stored in each node of the graph:

- *Point* represents a point of the time series.
- *Previous* represents the predecessor node of that node in the optimal solution up to that point and is needed to reconstruct the optimal solution at each instant.
- *Accumulated* represents the number of segments that approximate the series in the provisional solution obtained up to that point. It is the value to be minimized
- *ISE* represents the accumulated value of ISE in the provisional solution obtained up to that point.

To refer to each of the elements of the node, the notation *object.atribute* will be used. For example, the point corresponding to node N is named $N.Point$.

Algorithm 1 Pseudocode for the OSFS-OnL method

Input:
Chunk of data ($TS = \{t_{i1}, t_{i2}, \ldots, t_{in}\}$)
Previous Nodes ($prevNode_{k1}, prevNode_{k2}, \ldots, prevNode_{kn}$)
Error bound guarantee (e_b)
Previous solution graph ($previousCP$).
Output: Updated solution graph ($previousCP$)
Local Data Structures:
Priority queue of candidate nodes (Q_c).
Optimized ($Optimized$) vector indicating whether the shortest path to a node has been optimized. It is initialized to false for all nodes.
Begin-Pseudocode
for $(i = k1, k2, \ldots, kn)$ **do**
    1. $Q_c.enqueue(PrevNode_i)$
end for
repeat
    2. $minN \leftarrow Q_c.front()$
    3. $P \leftarrow minN.Point$
    4. $Q_c.deque()$
    5. $Optimized(P) = true$
    6. $previousCP(P) = minN.Previous$
    for $P_i \in FS(TS, P, e_b)$ and $Optimized(P_i) == false$ **do**
        For each P_i non-optimized $\in FS$ a new node N_i is created
        $7.N_i \leftarrow Node(P_i, 1 + minN.Accumul, minN, minN.ISE + ISE(P, P_i))$
        if $(N_i \notin Q_c)$ **then**
            8. $Q_c.enqueue(N_i)$
        else if $(N_i.Accumul \leq N_i.oldAccumul)$ **then**
            9. $Q_c.update(N_i)$
            N_i is updated in the queue with new Accumulated, Previous and ISE
        else
            10. *N_i is not updated*
        end if
    end for
until $Q_c.isEmpty()$

3.1.3 Description of the OSFS-OnL method

The OSFS-OnL method is described in more detail below, and its pseudo-code is shown in Algorithm 1.

- The initial point of the time series will be part of the optimal solution and, therefore, will form the initial node of the graph.
- The OSFS offline method is applied to the first chunk of data, obtaining the solution graph for this chunk. This graph will contain the optimal solution if only the first chunk of data are considered. All the nodes that contain points whose FS ends at the last point of the first chunk of data, must be stored, since its FS may increase when the next chunk of data are added. This particularity is very important and is the reason why the global optimum cannot be obtained from the optimum of each of the chunk of data that are added to the time series. Therefore, it must be considered each time a chunk of data are added.
- In this case, the solution graph is stored in an array of predecessors, in such a way that the ith element of the array stores the cut point prior to the ith point of the time series, in the optimal solution. Thus, by going back in the array of predecessors, the optimal solution can be obtained up to any point in the time series.

- Once the solution for the first chunk of data is obtained, the OSFS-OnL method is applied for each of the chunks of data, using as input parameters the solution graph of the previous chunks of data, and the previous nodes whose FS ends at the last point of the previous chunk of data ($prevNode$ in Algorithm 1). For each of the chunks of data added to the time series, the method is applied as follows.
  1. Insert in a priority queue the nodes corresponding stored in $prevNode$ (line 1 in Algorithm 1). This queue is ordered according to the value of *Accumulated*. In the case of equality in the value of *Accumulated*, the node with the lowest value of ISE has priority, so the value of ISE is minimized in the final solution (line 1 in Algorithm 1)
  2. Now an iterative process is followed consisting of the following steps:
  3. Repeat until priority queue is empty or the node containing the end point of the chunk of data has been selected from the priority queue.
     - Get the highest priority node from the queue and its corresponding point. Remove the node from the queue, mark it as optimized and store its previous cut point in the solution graph. In this case, the minimum path to that node can already be obtained (lines 2, 3, 4, 5, 6 in Algorithm 1). It should be noted that this node would meet the necessary condition to be part of the final optimal solution.
     - All the possible successor points of the point, corresponding to selected node, are obtained by using the FS (in for scheme, after line 6). This selection of points works as a pruning procedure and greatly reduces the search space.
     - Each of these possible successors could generate a candidate node, where its previous node would be the node just removed from the priority queue, its Accumulated value would be one unit higher than that of its previous node, and its ISE value would be the value of ISE of its previous node plus the value of ISE between both points (line 7 in Algorithm 1).
     - For each of the possible candidate nodes, three cases arise:
       - The node is not contained in the priority queue. In this case, this node is created and inserted in the priority queue (line 8 in Algorithm 1).
       - 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 both points. Finally, the new value of N_i is updated in the priority queue (line 9 in Algorithm 1)
       - In any other case, it is not updated (comment on line 10 in Algorithm 1).
  4. Once all chunks of data in the time series have been processed, and the priority queue is empty, the iterative process ends. To obtain the cut points, the graph solution (array $previousCP$) is used. To do this, it works backward from the last point of the array, using the stored values which contain the previous cut point at any point, until reaching the starting point of the time series.

3.1.4 Example of application of the method

Figures 1, 2 and 3 show an example of how the OSFS-OnL method works for a time series of 7 points. The example time series has 8 points ($P_0, P_1, \ldots, P_7$) and is assumed to have two chunks of data: $\{P_0, P_1, P_2, P_3\}$ and $\{P_4, P_5, P_6, P_7\}$. An error bound guarantee equal to 4 will be considered($e_b = 4$). 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:

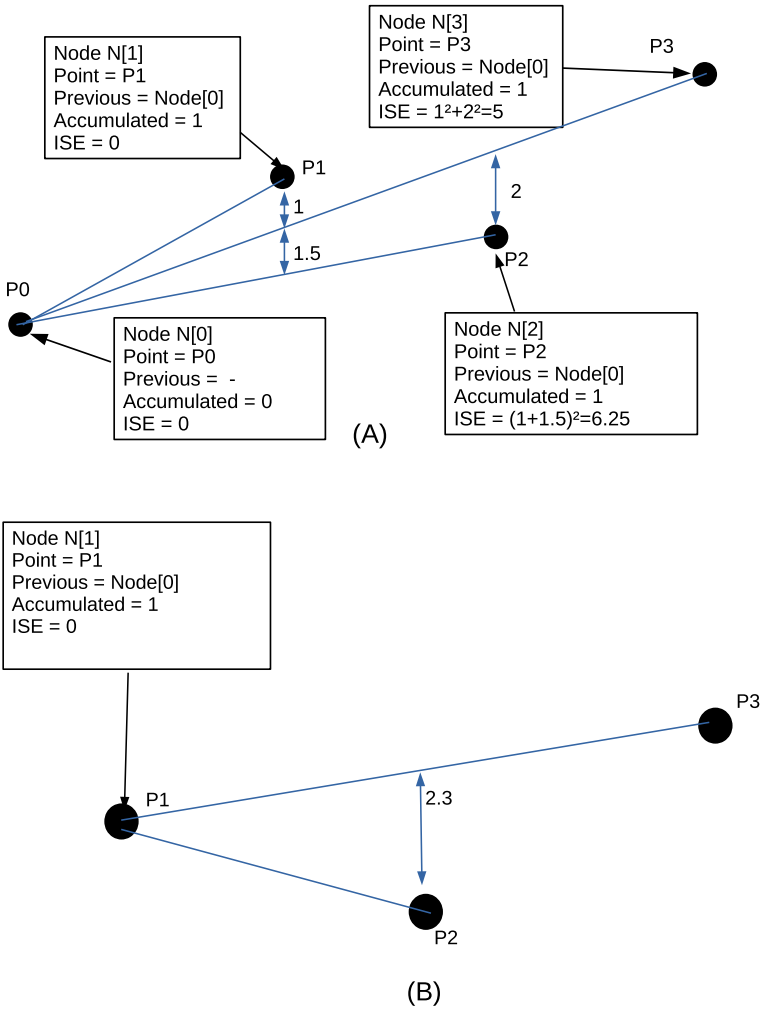
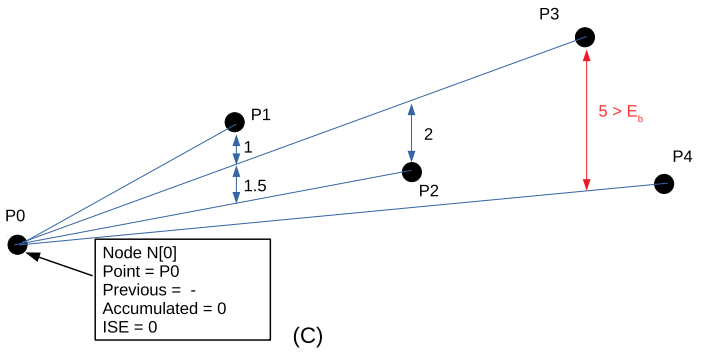
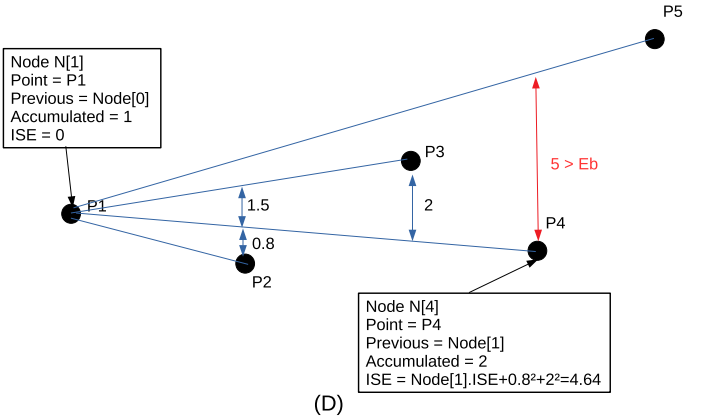


Fig. 1 Example of the proposed method (first part). Application of the method to the first chunk of data. **A** Points P1, P2 and P3 are included as possible successors of P0 in the optimal solution. **B** Points P2 and P3 cannot be successors of P1 in the optimal solution

First chunk of data (P_0, P_1, P_2, P_3)

- 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 are possible successors of P_0 (Fig. 1A). They will generate their corresponding nodes that will be inserted in the priority 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.
- Node $N[1]$, which contains point P_1, is removed from the queue.
- Points P_2, P_3, are possible successors of P_1 (Fig. 1B). The node $N[2]$ already contained point P_2, and it is not updated because the value of Accumulated would increase. The node





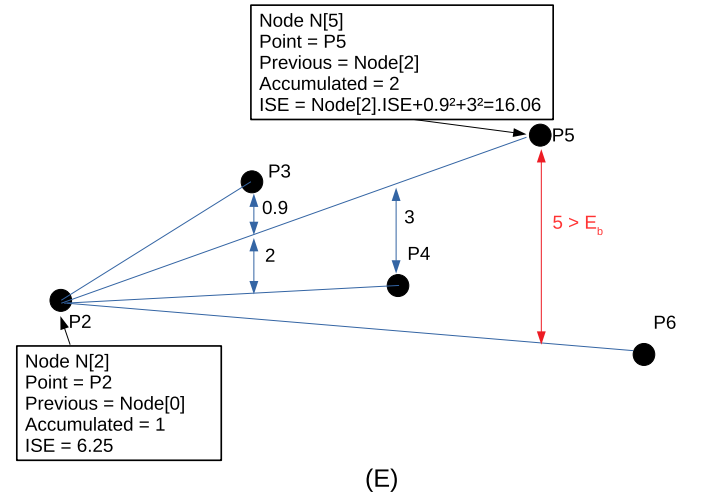


Fig. 2 Example of the proposed method (second part). Application of the method to the second chunk of data. **C** The feasible space of P0 ends in P2p; therefore, it is not taken into account for the second chunk of data. **D** The feasible space of P1 ends in P4; therefore, it is taken into account for the second chunk of data and includes node N[4] in the queue. **E** The feasible space of P2 ends in P5; therefore, it is taken into account for the second chunk of data and includes node N[5] in the queue

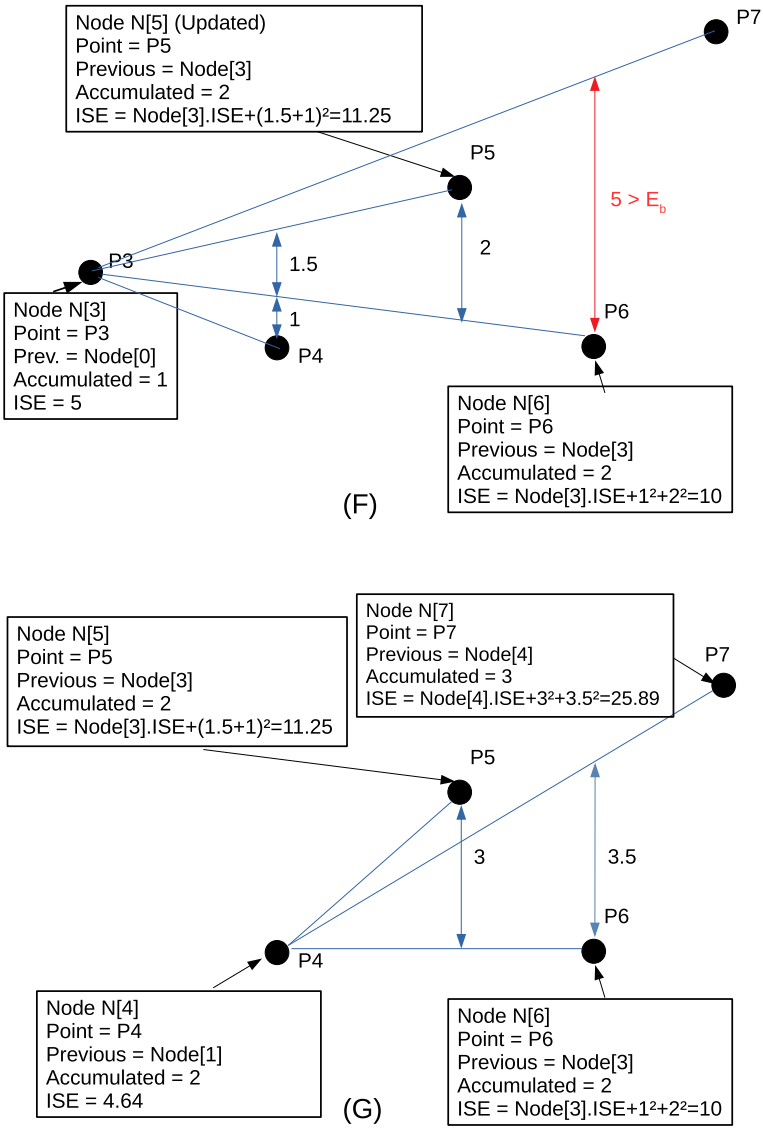


Fig. 3 Example of the proposed method (third part). Application of the method to the second chunk of data. **F** The feasible space of P3 ends in P6; therefore, it is taken into account for the second chunk of data, updates node N[5] and includes node N[6] in the queue. **G** The feasible space of P4 ends in P7, does not update nodes N[5] and N[6] and includes node N[7] in the queue

$N[3]$ already contained point P_3, and it is not updated because the value of Accumulated would increase.

- The priority queue would have the following nodes: $Q_c = \{N[2], N[3]\}$.
- Node $N[2]$, which contains point P_2, is removed from the queue.
- Point P_3 is a possible successor of P_2. The node $N[3]$ already contains point P_3, and it is not updated because the value of Accumulated would increase.

- The priority queue only contains node $N[3]$: $Q_c = \{N[3]\}$.
- Finally, the node $N[3]$, corresponding to the last point of the first group, is removed from the queue, and the queue is left empty. Thus, the method terminates for the first group of points in the time series, since the priority queue is empty, and the minimum path to all nodes in the first group has been optimized.

Since the predecessor of the last point of the first group is the point P_0, the CP of the solution for this first group of points would be P_0 and P_3. This partial solution could change when the points of the next group are processed.

Last chunk of data (P_4, P_5, P_6, P_7)

- As the last point of the FS of each one of the points of the first chunk of data is the point P_3 (last point of the previous group), the FS of these points can change when considering the new chunk of data. Therefore, the nodes corresponding to those points will be inserted in the priority queue. In this way, the priority queue will be: $Q_c = \{N[0], N[1], N[2], N[3]\}$
- Node $N[0]$, which contains point P_0, is removed from the queue. In this case, as can be seen in Fig. 2C, the points of the new group do not belong to the FS of point P_0 and therefore do not provide any possible successor.
- Node $N[1]$, which contains point P_1, is removed from the queue. In this case, as can be seen in Fig. 2D, only point P_4 is a possible successor to P_1 and will generate its corresponding node $N[4]$. Its Accumulated value is 2. In this way the priority queue will be: $Q_c = \{N[2], N[3], N[4]\}$.
- Node $N[2]$, which contains point P_2, is removed from the queue. In this case, as can be seen in Fig. 2E, points P_4 and P_5 are possible successors to P_2. The node $N[4]$ already contains point P_4, and it is not updated because the value of Accumulated would remain the same and the value of ISE increases. However, point P_5 generates a new node ($N[5]$) whose predecessor is $N[2]$ and its Accumulated value is 2. In this way the priority queue will be: $Q_c = \{N[3], N[4], N[5]\}$.
- Node $N[3]$, which contains point P_3, is removed from the queue. In this case, as can be seen in Fig. 3F, points P_4, P_5 and P_6 are possible successors to P_3. The node $N[4]$ already contains point P_4, and it is not updated because the value of Accumulated would remain the same and the value of ISE increases. The node $N[5]$ already contains point P_5 and it is updated because the value of Accumulated would remain the same and the value of ISE decreases. However, point P_6 generates a new node ($N[6]$) whose predecessor is $N[3]$ and its Accumulated value is 2. In this way, the priority queue will be: $Q_c = \{N[4], N[5], N[6]\}$.
- Node $N[4]$, which contains point P_4, is removed from the queue. In this case, as can be seen in Fig. 3G, points P_5, P_6 and P_7 are possible successors to P_4. The node $N[5]$ already contains point P_5, and it is not updated because the value of Accumulated would increase. The node $N[6]$ already contains point P_6, and it is not updated because the value of Accumulated would increase. However, point P_7 generates a new node ($N[7]$) whose predecessor is $N[4]$ and its Accumulated value is 3. In this way the priority queue will be: $Q_c = \{N[5], N[6], N[7]\}$.
- The next nodes to be removed from the queue would be $N[5]$, $N[6]$, $N[7]$ which do not improve the accumulated values for the remaining nodes, and therefore no node would be updated, and the queue would be empty.

Finally, the solution will be obtained from the previous value of the Node that contains $P7$ and so on. Therefore, the optimal CP will be $P7$, $P4$, $P1$ and $P0$. As can be seen, the partial optimal solution for the first group of points does not coincide with the final solution.

3.2 Complexity analysis

To study the computational complexity of the OSFS-OnL method, we will rely on the computational complexity of the OSFS method [20]. In this case, the complexity could be obtained on the basis that the OSFS-OnL method could be considered as the application of the OSFS method in each of the chunks of data into which the original time series is divided. It should also be taken into account that when the method is applied to a chunk of data, it is possible that the successors of some of the points of the previous chunk will have to be revised and, thus, the computational cost would increase. However, practice has shown that this additional computational cost is negligible compared to the cost of applying the method to a chunk of data. Therefore, the computational complexity of the OSFS-OnL method would be obtained by multiplying the number of chunk of data groups by the complexity of each chunk of data. Let n_{CD} be the number of chunks of data, n the total number of points in the series, n/n_{CD} the number of points in chunk of data and K the average number of CP in each chunk of data group, the computational complexity would be:

$$O\left(n_{CD}\left(\frac{n}{n_{CD}}\right)^2 log\left(\frac{n}{n_{CD}K}\right)\right) = O\left(\left(\frac{n^2}{n_{CD}}\right) log\left(\frac{n}{n_{CD}K}\right)\right) \quad (1)$$

If the computational complexities of both methods are compared, it can be seen that in the case of the OSFS-OnL method, the quadratic term is divided by the number of chunks of data (n_{CD}), as occurs in the logarithmic term. In this way, it can be concluded that the computational complexity of the OSFS-OnL method is lower than that of the OSFS method, as will also be shown in the experiments carried out.

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-OnL method has been implemented in C++, and all the experiments were run using an Intel(R) Core(TM) i7-10700 CPU at 2.90 GHzx8 with 16 GB of RAM.

To evaluate the performance of the OSFS-OnL method, three experiments with several synthetic and real-world time series collected from public repositories have carried out. The next time series have been used:

- *Stock prices time series from financial applications* including five different indexes: BBVA, Deutsche Bank, Intesa San Paolo, and Société Genéralé. These four series contain daily values from January 1, 1999, to February 9, 2015.
- *Wave height time series* (Hs) including three time series of significant wave height collected from buoys of the National Data Buoy Center of the USA have been used [30].
- Three datasets from the *UCR Time Series Classification Archive* were selected [31]: Hand Outlines, Mallat, and StarLightCurves.
- *Donoho–Johnstone time series* [32, 33], 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.
- *Arrhythmia dataset* contains cardiology data which belongs to the PhysioBank ATM of the MIT BIH Arrhythmia dataset [34].
- *Amazon closing prices time series* contains log returns of daily Amazon (AMZN) closing prices from January 2, 2004, to May 19, 2017 [35].

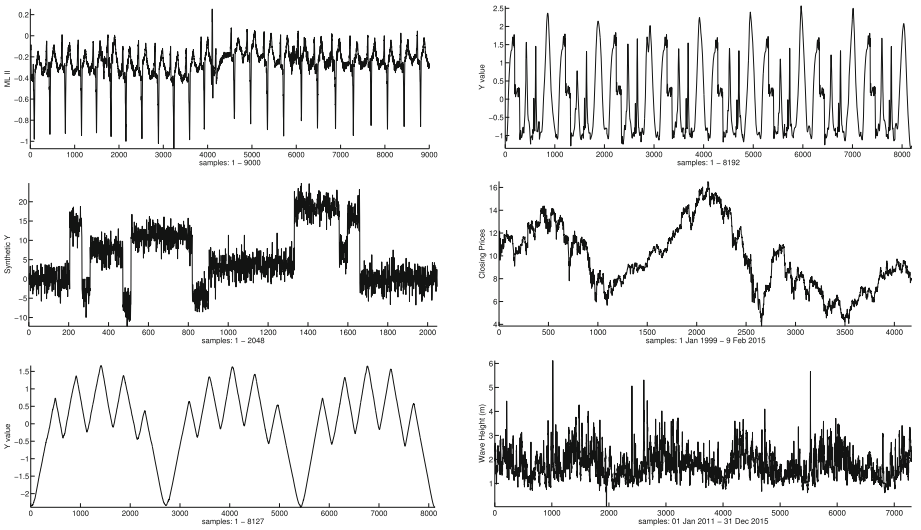


Fig. 4 Some representatives time series. From left to right and from top to bottom: Arrhythmia, Mallat, Donoho–Johnstone, BBVA, HandOutlines and buoy-41043

- *Daily gold fixing price* in London Bullion Market in US dollars per Troy ounce from January 3, 1995, to November 10, 2016 [35].
- *Wind speed time series*, includes observations of the maximum wind speed (m/s) over 10-min intervals recorded at a height of 40 m above ground level at a particular location [35].
- *Time series of the meteorological station of Córdoba, Spain*, contains average daily data on evaporation, humidity and temperature from 2000 to 2020 [36].
- *The Bonn EEG time series*, includes four time series (F001,N001,O001, Z001) each containing 100 single-channel EEG segments of 23.6-sec duration [37].

Some representative series are shown in Fig. 4.

4.1 Performance measures

To measure the effectiveness of the *OSTS* method, four measures were used:

- N_{cp}: number of CP or segments obtained in the segmentation of the time series. This is the main measure since it has been used as objective function to optimize.
- *RMSE*: root-mean-square error, which is obtained by dividing the square root of ISE by the number of points in the time series (n).

On the other hand, the following times were also measured:

- t_T: Segmentation time of the complete time series.
- t_g: Average segmentation time obtained by adding a chunk of data into which each time series has been divided.

4.2 First experiment: comparison of optimal methods

In this experiment, the OSFS-OnL method has been compared with two variants of the optimal offline OSFS method and with OptimalPLR method. In the first variant, the OSFS method is applied independently to each chunk of data into which the time series is divided. In this case, in order not to force the last point of each group to be a cut point, the penultimate cut point of each chunk of data has been used as the initial cut point to calculate the cut points of the next chunk of data. This variant will be referenced as *partial OSFS-OnL*. In the second variant, it is applied as proposed in Carmona-Poyato et al.[20], with which the optimal offline for the complete series is obtained.

In the tests of the OSFS-OnL method and the partial OSFS-OnL variant, each of the complete time series was divided into 20 groups of points of the same size. In the case of the OptimalPLR method, it has been applied to the complete time series.

As already mentioned above, the OptimalPLR method generates disconnected segments, and therefore, the number of CP is twice the number of segments obtained in the segmentation of the time series. However, in the rest of the methods used, the segments are connected and their ends are points of the time series. In this way, the number of segments is equal to the number of CP minus one.

In Carmona-Poyato et al. [20], the OSTS and OSFS methods were used to compare the L^2-norm with the $L\infty$-norm. The first one [22] obtains the optimal segmentation by prefixing the CP number and minimizing the RSME value (L^2-norm), and the second one obtains the optimal segmentation by fixing the maximum permissible error ($L\infty$-norm). To perform this comparison, the OSTS method was applied, setting the number of CP as 2.5% of the points of the time series, and the maximum error obtained ($L\infty$-norm) was used as input to the OSFS method. These maximum errors have also been used as error bound guarantee (e_b) in these experiments.

Table 1 shows the error bound guarantee (e_b), the computational times (t_T, t_g), number of CP (N_{cp}) and RMSE values for the OSFS-OnL method, for the partial OSFS-Onl method, for the OSFS method and for the OptimalPLR method. The values of N_{cp} and RMSE corresponding to the optimal offline OSFS method are similar to those of the OSFS-OnL method, so they are not shown in the table. The best results are marked in bold.

From the results obtained, the following can be highlighted:

- From the point of view of computational time, the OSFS-OnL method has much better performance than the offline OSFS method since the values of t_T of the online method are smaller in most cases. In some cases, the total time is reduced to more than a tenth of the time of the offline method. This fact corroborates the results of the computational complexity analysis. This drastic time reduction, and in view of the values of t_g, would allow the proposed method to be used in applications where the processing times required for the chunk of data added to the time series are in the order of tens of milliseconds. However, the OptimalPLR method has much better performance than the rest of the optimal methods, except in the case of the MALLAT series.
- Regarding the number of CP obtained, in the case of the partial OSFS-OnL method is somewhat higher than that of the OSFS-OnL method. This worsening is compensated by the computational time, which is somewhat lower than in the OSFS-OnL method. This small reduction is due, as mentioned above, to the fact that in the OSFS-OnL method, when a new chunk of data are added, the points of the previous chunks can influence the obtaining of the new optimal segmentation. Except in four cases, the performance of the optimalPLR method is worse than that of all variants of the OSFS method.

Table 1 Values of error bound guarantee (e_b), computational times (ms), number of CP (N_{cp}) and RMSE for the three optimal methods

Time series	Error	OSFS-OnL				Partial OSFS-OnL				OSFS (2)	OptimalPLR		
	e_b	t_T (ms)	t_g (ms)	N_{cp}	RMSE	t_T (ms)	t_g (ms)	N_{cp}	RMSE	t_T	t_T (ms)	N_{cp}	RMSE
HandOutlines(8127)	0.018	289.7	13.7	**126**	**0.006**	286.6	**13.6**	129	**0.006**	2723.6	**104.8**	158	0.009
MALLAT(8192)	0.258	188.9	8.9	**176**	**0.104**	**181.4**	**8.6**	177	**0.104**	2785.7	224.0	270	0.155
StarLightCurves(8192)	0.038	273.9	13.0	**166**	**0.016**	245.2	**11.6**	168	**0.016**	2808.9	**210.1**	234	0.023
Donoho–Johnstone(2048)	8.405	176.6	8.4	**22**	**2.651**	125.7	**5.9**	23	2.688	212.7	**18.9**	22	2.779
BBVA(4174)	1.323	461.9	21.9	**29**	0.520	342.8	**16.3**	34	**0.512**	815.6	**44.9**	36	0.617
DEUTSCHE(4174)	7.466	340.6	16.2	**40**	2.581	284.9	**13.5**	42	**2.500**	788.7	**47.3**	48	3.540
SO-GENERAL(4174)	8.829	510.3	24.3	**33**	3.033	346.8	**16.5**	36	**3.007**	820.9	**41.0**	38	4.076
b46075f(7303)	3.950	1815.8	86.4	**31**	1.704	883.8	**42.0**	36	**1.608**	2442.9	**59.4**	30	2.205
b41043f(7303)	1.751	3094.5	147.3	**17**	0.741	1899.0	**90.4**	22	**0.678**	2527.1	**62.2**	22	0.822
b41044f(7303)	1.464	1142.9	54.4	**40**	**0.526**	728.9	**34.7**	44	0.533	2382.5	**65.1**	42	0.674
Arrhythmia(9000)	0.114	299.3	14.9	**180**	0.038	285.1	**14.2**	181	**0.037**	3334.9	**110.9**	258	0.060
Amazon Prices(3367)	0.132	1052.8	50.131	23	0.027	433.4	**20.637**	28	**0.026**	545.6	**27.1**	**20**	0.046
Gold Prices(5526)	54.840	2359.6	112.362	**64**	20.397	1428.4	**68.017**	66	**20.153**	1511.5	**55.0**	82	25.322
F001(4097)	28.523	28.3	1.349	**65**	10.908	**23.9**	**1.137**	69	**10.903**	135.3	48.3	150	14.756
N001(4097)	81.436	174.3	8.300	**61**	**34.605**	140.1	**6.672**	63	35.615	723.5	**48.7**	70	42.522
O001(4097)	123.330	535.8	25.512	**26**	**46.514**	254.9	**12.138**	29	48.1469	771.3	**34.7**	30	49.889
Z001(4097)	94.258	246.9	11.761	41	37.705	177.2	**8.436**	45	**37.453**	732.1	**36.9**	**40**	40.736
Wind(1008)	5.863	123.9	5.901	**8**	2.443	51.8	**2.468**	13	**1.966**	58.232	**8.6**	**8**	2.616
Evaporation(7155)	3.490	623.1	29.969	**40**	**0.977**	473.7	**22.557**	41	1.016	2185.5	**61.9**	44	2.421
Humidity(7188)	37.262	1786.3	85.061	15	17.064	705.9	**33.618**	23	**16.742**	2307.9	**53.2**	**14**	18.302
Temperature(7182)	7.916	380.3	18.112	**42**	**2.562**	351.2	**16.726**	**42**	2.633	2189.3	**69.2**	78	3.579

In parentheses, in the name of each time series, the number of points in the time series is shown

- Regarding the RMSE values, the results are very similar and there are no significant differences in the different variants of the OSFS method. In the case of the OptimalPLR method, the values are worse in all time series.

4.3 Second experiment: comparison with heuristic methods

In this subsection, the OSFS-OnL method is compared to the suboptimal online methods described in Sect. 2.

Table 2 shows the computational times (t_T, t_g), number of CP (N_{cp}) and RMSE values for the SW, FSW and SFSW methods; and the number of CP (N_{cp}) and RMSE values for the proposed method. In this case, the values of t_T and t_g for the OSFS-OnL, which were shown in Table 1, were not included in order to reduce its size. The error bound guarantee (e_b) values are the same as in the first experiment. The best results are marked in bold.

From the results obtained, the following can be highlighted:

- From a computational time point of view, the OSFS-OnL method, as expected, performs much worse than the suboptimal methods. The best by far is the FSW method.
- The values of Ncp are smaller in the proposed method. In the case of the SW method, the differences are appreciable; however, in the case of the FSW and SFSW methods, the differences are smaller. The difference between the SW method and the FSW and SFSW methods is that the former is much more restrictive when calculating the lengths of the segments (distances between CP). Of the suboptimal methods, the SFSW method is the one that obtains the best results, but at the cost of a worse computational performance.
- Regarding the RMSE values, the results of the OSFS-OnL method are similar to those of the SFSW method and better than those of the SW and FSW methods.

4.4 Third experiment: computational time versus size of chunk of data

In this experiment we have analyzed how the size of the chunk of data influences computational time in the proposed method. For this, the average time obtained when processing each chunk of data has been calculated. Tests have been made with different sizes of the chunk of data.

Table 3 shows the times obtained. In each column, the size of the chunk is expressed as a percentage of the total size of the time series. As can be seen, the computational times increase as the size of the chunk of data increases. Therefore, the smaller the chunk of data, the better the performance of the method.

5 Conclusions and future improvements

The following conclusions can be drawn from this work.

- An optimal online time series segmentation method has been proposed, which guarantees a preset error limit ($L\infty$-norm) and minimizes the number of segments obtained.
- The method selects the solution that minimizes the value of RMSE, among all the optimal solutions.
- To reduce the search space of possible solutions, it has been used the FS method [23].

Table 2 Values of number of CP (N_{cp}) and RMSE for the proposed method, and values of computational times (ms), number of CP (N_{cp}) and RMSE for the three suboptimal methods

Time series	OSFS-OnL		SW method				FSW method				SFSW method			
	N_{cp}	RMSE	t_T (ms)	t_g (ms)	N_{cp}	RMSE	t_T (ms)	t_g (ms)	N_{cp}	RSME	t_T (ms)	t_g (ms)	N_{cp}	RMSE
HandOutlines	**126**	**0.006**	18.3	0.87	169	0.009	**6.5**	**0.31**	134	0.009	50.1	2.38	131	0.007
MALLAT	**176**	**0.104**	11.8	0.56	214	0.146	**6.9**	**0.33**	182	0.145	24.1	1.15	181	0.111
StarLightC	**166**	**0.016**	15.3	0.73	186	0.023	**7.2**	**0.34**	173	0.022	44.0	2.09	170	0.018
Donoho-J	**22**	**2.651**	2.4	0.11	87	2.904	**2.2**	**0.10**	27	2.753	10.1	0.48	25	2.728
BBVA	**29**	0.520	11.6	0.55	65	0.599	**4.8**	**0.22**	41	0.579	35.5	1.69	39	**0.497**
DEUTSCHE	**40**	**2.581**	13.7	0.65	72	3.166	**4.4**	**0.21**	49	3.290	29.7	1.41	43	2.749
SO-GENERAL	**33**	**3.033**	16.3	0.77	61	4.012	**4.7**	**0.22**	42	3.730	51.1	2.43	40	3.227
b46075f	**31**	**1.704**	35.3	1.68	121	1.926	**8.4**	**0.40**	48	2.046	137.4	6.54	37	1.788
b41044f	**17**	0.741	51.7	2.46	73	0.764	**7.4**	**0.35**	28	0.735	73.0	3.48	22	**0.739**
b41044f	**40**	**0.526**	32.1	1.52	120	0.675	**6.7**	**0.31**	57	0.647	101.1	4.81	51	0.572
Arrhythmia	**180**	**0.038**	16.2	0.81	214	0.054	**7.7**	**0.38**	191	0.053	46.2	2.31	185	0.040
Amazon Prices	**23**	**0.027**	18.1	0.860	77	0.033	**7.3**	**0.349**	28	0.0291	25.7	1.223	26	0.027
Gold Prices	**64**	20.396	77.2	3.674	122	24.832	**8.1**	**0.386**	79	23.210	50.4	2.404	75	**20.265**
F001	**65**	10.908	**1.4**	**0.067**	95	14.096	1.8	0.085	75	13.651	6.4	0.304	71	**11.084**
N001	**61**	34.604	7.3	0.347	104	42.573	**4.4**	**0.208**	70	40.594	27.3	1.303	69	**36.406**
O001	**26**	46.514	9.2	0.440	87	55.031	**4.8**	**0.230**	39	52.325	46.9	2.236	32	**48.241**
Z001	**41**	37.704	7.2	0.340	117	43.739	**4.1**	**0.192**	54	40.646	70.494	3.357	52	**38.377**
Wind	**8**	2.443	2.5	0.118	21	2.611	**1.8**	**0.083**	9	2.576	5.0	0.240	**8**	**2.546**
Evaporation	**40**	0.976	26.1	1.245	113	1.631	**6.6**	**0.312**	49	1.483	126.8	6.038	52	1.018
Humidity	**15**	17.063	19.3	0.920	110	16.380	**7.6**	**0.364**	34	16.076	206.5	9.836	21	**15.820**
Temperature	**42**	2.561	15.4	0.732	144	3.234	**6.31**	**0.300**	62	3.107	40.784	1.9421	52	**2.694**

Table 3 Values of average segmentation time (t_g) by adding a chunk of data for different sizes

Time series	Size chunk (% size time series)							
	2.0	2.5	4.0	5.0	8.0	10.0	15.0	20.0
HandOutlines	7.459	7.46	12.502	13.796	23.186	25.186	39.629	44.724
MALLAT	4.251	4.225	7.563	8.995	14.242	16.607	25.661	30.006
StarLightC	6.223	6.236	10.349	13.043	19.154	23.287	34.030	40.272
Donoho-J	5.229	5.175	6.900	8.412	10.451	12.096	14.608	19.827
BBVA	16.201	16.183	21.653	21.998	31.562	32.596	47.078	55.546
DEUTSCHE	10.116	9.986	14.564	16.220	21.574	26.463	34.686	45.061
SO-GENERAL	17.442	17.437	23.317	24.303	34.352	37.653	50.662	58.657
b46075f	67.624	67.784	83.647	86.467	124.352	113.449	161.026	173.894
b41044f	112.494	112.589	137.573	147.358	184.371	204.236	252.215	286.289
b41044f	38.255	38.113	52.516	54.427	76.011	88.318	114.244	140.057
Arrhythmia	6.926	6.911	11.706	14.969	21.417	28.599	38.883	53.462
Amazon	38.237	37.648	45.070	50.131	54.225	68.848	74.118	102.736
Gold	86.547	86.297	95.678	112.362	138.742	153.293	194.595	223.897
F001	0.716	0.720	1.194	1.348	2.078	2.402	3.489	4.637
N001	4.703	4.697	7.078	8.300	12.045	14.063	20.358	24.093
O001	17.741	17.504	23.112	25.512	34.242	37.253	50.669	57.402
Z001	7.103	7.011	9.882	11.761	16.306	18.288	28.541	30.417
Wind	5.022	5.023	5.812	5.900	7.491	7.749	9.244	10.843
Evaporation	15.337	15.335	23.794	29.669	40.607	48.914	72.135	96.066
Humidity	62.017	62.302	77.680	85.061	105.873	118.924	148.518	178.163
Temperature	9.734	9.743	16.135	18.111	28.751	32.188	49.955	56.593

Sizes are represented as a percentage of the size of the time series

- Its computational complexity is lower than that of the optimal offline method (OSFS) proposed in Carmona-Poyato et al. [20], as also shown by the experiments carried out, which in some cases reduce the computational time to tenth part.
- The times obtained in the experiments show that the processing times of the new chunk of data added are of the order of tens of milliseconds.
- From the point of view of computational cost, the suboptimal methods with which it has been compared have better performance, although in terms of error and information reduction, the proposed method greatly improves the suboptimal methods.

The main drawback of the method is its computational complexity, since for example the FSW or SFSW method offers very good results in terms of information reduction, using much less time. However, as it is an online method, it greatly reduces computation times compared to the optimal offline one.

The main advantages are:

- It is not necessary to have the complete time series to segment it.
- Its usefulness for evaluating the performance of suboptimal online methods
- The possibility of applying the method in some real-time applications, as long as the required response times are not less than tens of milliseconds.

Regarding future improvements, we could try to reduce the computation time at the level of suboptimal methods by adding some probabilistic component, at the cost of guaranteeing the optimality of the method with a predetermined probability.

Acknowledgements This work has been developed with the support of the Research Project PID2019-103871GB-I00 of Spanish Ministry of Economy, Industry and Competitiveness.

Data availability The datasets supporting Fig. 4, as well as those that have been used in the experiments, are publicly available in https://github.com/ma1capoa/OSFS_Method/tree/main/TimeSeriesFiles. No datasets were generated during the current work.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

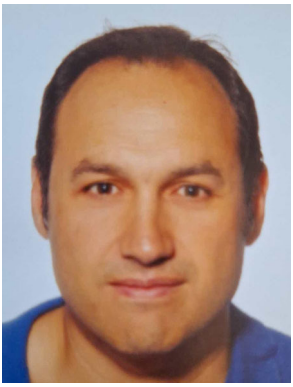
References

1. Pérez-Ortiz M, Durán-Rosal A, Gutiérrez P, Sánchez-Monedero J, Nikolaou A, Fernández-Navarro F, Hervás-Martínez C (2019) On the use of evolutionary time series analysis for segmenting paleoclimate data. Neurocomputing 326–327:3–14
2. Deng W, Wang G (2017) A novel water quality data analysis framework based on time-series data mining. J Environ Manag 196:365–375
3. Koski A, Juhola M, Meriste M (1995) Syntactic recognition of ECG signals by attributed finite automata. Pattern Recognit 28:1927–1940
4. Lee C-H, Liu A, Chen W-S (2006) Pattern discovery of fuzzy time series for financial prediction. IEEE Trans Knowl Data Eng 18:613–625
5. Okawa M (2021) Time series averaging and local stability weighted dynamic time warping for online signature verification. Pattern Recognit 112:107699
6. Cortes C, Fisher K, Pregibon D, Rogers A, Smith F (2000) Hancock: a language for extracting signatures from data streams. In: Proceeding of the sixth ACM SIGKDD international conference on knowledge discovery and data mining
7. Fu T-C (2011) A review on time series data mining. Eng Appl Artif Intell 24(1):164–181
8. Chatfield C (2000) Time-series forecasting. CRC Press, Boca Raton

9. Van Trees HL (2004) Detection, estimation, and modulation theory, part I: detection, estimation, and linear modulation theory. Wiley, Hoboken
10. Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv 41:1–58
11. Fawaz HI, Forestier G, Weber J, Idoumghar L, Muller P-A (2019) Deep learning for time series classification: a review. Data Min Knowl Disc 33:917–963
12. Aghabozorgi S, Shirkhorshidi A, Wah T (2015) Time-series clustering-a decade review. Inf Syst 53:16–38
13. Weigend A (1994) Time series prediction: forecasting the future and understanding the past, 1st edn. Routledge, London
14. Kamalzadeh H, Ahmadi A, Mansour S (2017) A shape-based adaptive segmentation of time-series using particle swarm optimization. Inf Syst 67:1–18
15. Tseng V, Chen C-H, Huang P-C, Hong T-P (2009) Cluster-based genetic segmentation of time series with DWT. Pattern Recognit Lett 30:1190–1197
16. Fuchs E, Gruber T, Nitschke J, Sick B (2009) On-line motif detection in time series with swift motif. Pattern Recognit 42:3015–3031
17. Keogh E, Chu S, Hart D, Pazzani M (2004) Segmenting time series: a survey and novel approach. In: Data mining in time series databases, pp 1–22
18. Fuchs E, Gruber T, Nitschke J, Sick B (2010) Online segmentation of time series based on polynomial least-squares approximations. IEEE Trans Pattern Anal Mach Intell 32(12):2232–2245
19. Xie Q, Pang C, Zhou X, Zhang X, Deng K (2014) Maximum error-bounded piecewise linear representation for online stream approximation. VLDB J 23:915–937
20. Carmona-Poyato A, Fernández-Garcia NL, Madrid-Cuevas FJ, Durán-Rosal AM (2021) A new approach for optimal offline time-series segmentation with error bound guarantee. Pattern Recognit 115:107917
21. Sarker IH (2019) Context-aware rule learning from smartphone data: survey, challenges and future directions. J Big Data 6:95
22. Carmona-Poyato A, Fernandez-Garcia NL, Madrid-Cuevas F, Duran-Rosal A (2020) A new approach for optimal time-series segmentation. Pattern Recognit Lett 135:153–159
23. Liu X, Lin Z, Wang H (2008) Novel online methods for time series segmentation. IEEE Trans Knowl Data Eng 20(12):1616–1626
24. Chu C (1995) Time series segmentation: a sliding window approach. Inf Sci 85(1):147–173
25. Keogh E, Chakrabarti K, Pazzani M, Mehrotr S (2001) Dimensionality reduction for fast similarity search in large time series databases. Knowl Inf Syst 3(3):263–286
26. Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing SAX: a novel symbolic representation of time series. Data Min Knowl Disc 15:107–144
27. Keogh E, Chu S, Pazzani M, Hart D, Pazzani M (2001) An online algorithm for segmenting time series. In: Proceedings 2001 IEEE international conference on data mining, pp 289–296
28. Salotti M (2002) Optimal polygonal approximation of digitized curves using the sum of square deviations criterion. Pattern Recognit 35:435–443
29. Pikaz A, Dinstein I (1995) Optimal polygonal approximation of digital curves. Pattern Recognit 28:373–379
30. NOAA (2015) National Buoy Data Center. http://www.ndbc.noaa.gov/
31. Dau HA, Keogh E, Kamgar K, Yeh C (2018) The UCR time series classification archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ (October)
32. Donoho DL, Johnstone IM (1994) Ideal spatial adaptation by wavelet shrinkage. Biometrika 81(3):425–455
33. Donoho DL, Johnstone IM (1995) Adapting to unknown smoothness via wavelet shrinkage. J Am Stat Assoc 90(432):1200–1224
34. Moody GB, Mark RG (2001) The impact of the MIT-BIH arrhythmia database. IEEE Eng Med Biol Mag 20(3):45–50
35. Tsay RS (2010) Analysis of financial time series, 3rd edn. Wiley, Hoboken
36. IFAPA (2023) Red de Información Agroclimática de Andalucía. https://www.juntadeandalucia.es/agriculturaypesca/ifapa/riaweb/web/estacion/14/6
37. Universidad Pompeu Fabra (2023) The Bonn EEG time series. https://www.upf.edu/web/ntsa/downloads

Ángel Carmona-Poyato received his title of Agronomic Engineering and Ph.D. degree from the University of Cordoba (Spain), in 1986 and 1989, respectively. Since 1990, he has been working with the Department of Computing and Numerical Analysis of the University of Cordoba. Currently he is an assistant professor. His research is focused on time series processing, image processing, and 2D object recognition.

Nicolás-Luis Fernández-García received the bachelor degree in Mathematics from Complutense University of Madrid (Spain) in 1988. He received the PhD in Computer Science from the Polytechnic University of Madrid (Spain) in 2002. Since 1990, he has been working with the Department of Computing and Numerical Analysis at Cordoba University. He is a lecturer of Computer Science and Artificial Intelligence at Cordoba University. His research is focused on edge detection, 2D object recognition, evaluation of computer vision algorithms, and time series processing.



Francisco-José Madrid-Cuevas received the B.S. degree in computer science from Malaga University, Spain, and the Ph.D. degree from the Polytechnic University of Madrid, Spain, in 1995 and 2003, respectively. Since 1996, he has been with the Department of Computing and Numerical Analysis of Cordoba University, where he is currently an Assistant Professor. His research is focused mainly on image segmentation, 2D object recognition, evaluation of computer vision algorithms, and time series processing.

Rafael Muñoz-Salinas received his Bachelor's degree in Computer Science from the University of Granada (Spain) in 2003 and his Ph.D. degree from the University of Granada in 2006. Since then, he has been working with the Department of Computing and Numerical Analysis of Cordoba University where he is currently a full professor. His research is focused mainly on computer vision, soft computing techniques applied to robotics, human-robot interaction and time series processing. He has co-authored more than one hundred papers in conferences, books, and top-ranked journals. One of his papers was the most cited of the prestigious journal Pattern Recognition and is considered a highly-cited paper. He has advised seven Ph.D. students. And participated in more than twenty projects, both industrial and scientific. He has been a visiting researcher at the Universities of DeMontfort (U.K), Orebro (Sweden), TUM (Munich), INRIA(France), Groningen (Netherlands). As a teacher, he has been teaching for more than fifteen years, advised more than 30 final degree projects, and taught international courses in the Universities of Groningen, Luxembourg, Malta, Coimbra (Portugal), and Brno. He is also a researcher at the Biomedical Research Institute of Cordoba (IMIBIC).

Francisco-José Romero-Ramírez received a Bachelor's degree in Computer Science in 2013 and a master's degree in Geomatics and Remote Sensing in 2015 from the University of Cordoba (Spain). He obtained a Ph.D. degree in Computer Vision from the University of Córdoba (Spain) in 2021. He has participated in several national and European research projects where his main tasks have been mainly focused on the processing and analysis of multispectral images and LiDAR. Currently, he is working at the Department of Computer Science and Numerical Analysis of the University of Cordoba. His research is focused on computer vision and time series processing.