Pattern Recognition Letters
journal homepage: www.elsevier.com

A new approach for optimal time-series segmentation

Ángel Carmona-Poyato[a,**], Nicolás Luis Fernández-García[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 Andalucia, c/ Escritor Aguayo, 4, Córdoba, Spain*

ABSTRACT

Emerging technologies have led to the creation of huge databases that require reducing their high dimensionality to be analysed. Many suboptimal methods have been proposed for this purpose. On the other hand, few efficient optimal methods have been proposed due to their high computational complexity. However, these methods are necessary to evaluate the performance of suboptimal methods. This paper proposes a new optimal approach, called *OSTS*, to improve the segmentation of time series. The proposed method is based on A^* algorithm and it uses an improved version of the well–known Salotti method for obtaining optimal polygonal approximations. Firstly, a suboptimal method for time-series segmentation is applied to obtain pruning values. In this case, a suboptimal method based on Bottom-Up technique is selected. Then, the results of the suboptimal method are used as pruning values to reduce the computational time of the proposed method. The proposal has been compared to other suboptimal methods and the results have shown that the method is optimal, and, in some cases, the computational time is similar to other suboptimal methods.

1. Introduction

Nowadays, the high dimensionality of data and the large size of databases has originated the development of methods to reduce its dimensionality with minimum information loss (Kamalzadeh et al., 2017). These methods are applied in time series data mining, an important area of research in many and different fields of science. Due to this, the problem of time series segmentation is an important task in data mining.

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 are applied to different problems, such as clustering (Ferreira and Zhao, 2016) and classification (Zhao and Itti, 2016), among others (Duran-Rosal et al., 2018a).

Segmentation can be considered as a discretization problem and aims to accurately approximate time series. It is used to reduce the dimensionality of the time series while preserving essential features and characteristics of the original time series (Sangeeta and Geeta, 2012).

In the literature, time series segmentation techniques are also called *time series representation procedures*. Bajcsy et al. (1990) even make the point that both steps should not be handled separately. A new symbolic representation of time series and a hierarchical classification of all the various time series representations were proposed by Lin et al. (2007).

Segmentation consists of cutting the time series in some specific points, called *cut points* (*CP*), in order to achieve two important goals: 1) the search of useful patterns or segments in time series (Nikolaou et al., 2015) and 2) the reduction of the dimensionality or the number of points of time series by transforming it into a new representation space (Aghabozorgi and Wah, 2015)

This work can be included in the second objective. In this case, the general approach is to segment a time series into subsequences (windows), and, then, primitive shape patterns are selected to improve the representation of the original time series. The simplest and most intuitive technique is called *Piecewise Linear Representation (PLR)*. This technique approximates the time series, of length n, with K straight lines. There are two methods for obtaining the straight lines, by using linear

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

interpolation or by using linear regression (Keogh et al., 2004).

The proposed method uses PLR with linear interpolation due to its low computational complexity. Usually, K is much smaller than n to reduce the number of points of the time series and to make its computation more efficient.

The problem can be addressed in different ways (Keogh et al., 2004).

- Given a time series, obtain the best segmentation using K segments.
- Given a time series, obtain the best segmentation such that the maximum error for any segment does not exceed a prefixed threshold.
- Given a time series, obtain the best segmentation such that the accumulated error of all segments is less than some prefixed threshold.

The error is calculated from the vertical differences between the straight line obtained in the segmentation and the real data points, squaring them and then summing them together (i.e. the L^2-norm between the line and the data).

Another commonly used measure of goodness of fit is the distance between the best fit line and the data point furthest away in the vertical direction (i.e. the $L\infty$-norm between the line and the data).

This work can be included in the first item and uses L^2-norm.

Considering how to get the cut points, segmentation methods can be classified into three three categories (Keogh et al., 2004).

- *Sliding Windows*: A segment is grown until it exceeds some error bound. The process repeats with the next data point not included in the newly approximated segment.
- *Top-Down*: The time series is recursively partitioned until some stopping criteria is met.
- *Bottom-Up*: Starting from the finest possible approximation, segments are merged until some stopping criteria is met.

On the other hand, the methods that segment the complete time series are called *offline methods* and those that produce segments based on the data seen so far are called *online methods*. Recently, other techniques based on metaheuristics (Genetic Algorithms and particle swarm optimisation) have been used (Duran-Rosal et al., 2018a), (Duran-Rosal et al., 2018b). Another recent and interesting work proposes an online segmentation method that adapts unannounced mutations of the data (Gonzalez-Vidal et al., 2018). Sarker (2019) discusses the problem of time-series segmentation by comparing the static with the dynamic segmentation, taking into account if the number of segments is preset or not. Sarker et al. (2019) considers time segments according to recent activities. This work proposes an offline method, called *OSTS*, based on an improved and adapted version (Carmona-Poyato et al., 2017) of the well-known Salotti method (Salotti, 2002) for obtaining optimal polygonal approximations. In order to reduce the computational time, the present work uses pruning values obtained from a suboptimal Bottom-Up method. For this purpose, an adapted version of the suboptimal Pikaz method (Pikaz and Dinstein, 1995), improved by Masood (2008), has been used. The present paper is arranged as follows. Section 2 describes the related and used works to compare the proposed method. 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, the methods used to compare with the proposed method will be briefly described.

Bottom-Up method (Keogh et al., 2004): in this method, all the points are considered as possible cut points. Then, in each iteration, the two adjacent segments that produce a minimum error are merged.

Top-Down method (Keogh et al., 2004) is the complementary algorithm. At the beginning, the segment that joins the first and the last point of the time series is used to segment all the time series. Then, this method recursively divides the segment by selecting the point that produces the maximum error reduction as cut point.

Sarker et al. (2017) proposed an improved version of Bottom-Up method. A dynamic behavior-oriented time segmentation approach for extracting temporal behaviour rules, applied to mine mobile user behavior, was used. This approach dynamically identifies the optimal continuous time segments, each of which is dominated by a particular behavior of the user.

The previous methods are run until some stopping criteria are met (related to the approximation error). In order to use these algorithms, they have been adapted and the stopping criteria is the final number of cut points.

Duran-Rosal et al. (2018b) used Bottom-Up and Top-Down methods as a local search strategy, and then this local search was combined with the metaheuristics GA (Genetic Algorithm), PSO (Particle Swarm Optimisation) and the different PSO variants: at the beginning of the evolution, a 50% of the population is randomly selected, and these individuals are improved by the local search. After that, the metaheuristic is applied to the complete population, including standard random individuals and the ones improved by the local search method. Finally, the best solution obtained by the metaheuristic is also applied to local search. Taking into account this procedure and the method described in (Duran-Rosal et al., 2016), a 40% of the cut points, found by the GA, Barebones particle swarm optimisation algorithm (BBPSO), Exploiting barebones particle swarm optimisation algorithm (BBePSO), and Dynamic exploiting barebones particle swarm optimisation algorithm (DBBePSO) are fine-tuned resulting the hybrid versions of GA, PSO and PSO variants (HGA, HPSO, HBBPSO, HBBePSO, HDBBePSO) (Duran-Rosal et al., 2018b).

For all algorithms, the population size is 100. The number of segments is set to a 2.5% of the total number of points of the time series. The stop criterion of all the algorithms is a maximum number of fitness evaluations, which is established based on the length of each time series, N. These algorithms have been run 30 times with different random seeds.

3. Proposed method

Although the optimal methods have high computational complexity and cannot be used in real-time applications, they can be used to assess the goodness of suboptimal methods. Duncan and Bryant (1996) suggest to use dynamic programming for optimal time series segmentation. However, this method has high computational complexity like all methods based on this technique. On the other hand, it is possible to reduce the computational time of an optimal method using an efficient pruning technique. So, an optimal method could achieve computational times comparable to suboptimal methods. This is the goal of the present work.

The following features of the *OSTS* method can be highlighted:

- To obtain the optimal time series segmentation, a method based on an improved version of Salotti method is proposed (Carmona-Poyato et al., 2017).
- A suboptimal method, in this case Pikaz method, is used to obtain pruning values in order to reduce its computational time (Pikaz and Dinstein, 1995).

The method is detailed in the following subsections.

3.1. Adapted Pikaz suboptimal method

Pikaz and Dinstein (1995) proposed a very efficient suboptimal method based on a greedy iterative algorithm to obtain polygonal approximation of a boundary. This method eliminates the redundant point of the contour with the smallest associated error value. To calculate the associated error with each point P_j, two neighbouring points, P_{j-1} and P_{j+1}, are joined with a straight line. Maximum perpendicular (squared) distance of all boundary points between P_{j-1} and P_{j+1} from the straight line is called as *associated error value* of point P_j. In each iteration, only the point with the smallest associated error value is deleted. If more than one point with the smallest associated error value exists, any of them may be removed because sequence of removal (in case of more than one candidate) will not affect the results. When a point is deleted, only the associated error with its two neighbours must be updated. This method has a low computational complexity ($O(n \log n)$) and can produce polygonal approximation with any pre-set number of final points. This method has been adapted in order to apply it in time series. The associated error is calculated from the vertical differences between each point P_j and the straight line that joins P_{j-1} and P_{j+1} instead of perpendicular distance. In this way, a preset number of cut points of the time series could be obtained. This method can be considered a Bottom-Up method.

So that the results of this method can be applied as pruning values for the optimal method, the accumulated error values for each point in the series are calculated. This value is called the integral squared error (ISE). The value of ISE_n for each point P_n of the time series is calculated as:

$$\mathrm{ISE}_n = \sum_{i=1}^{n} e_i^2 \tag{1}$$

where e_i is the vertical distance from P_i to its corresponding approximated segment.

3.2. Optimal Salotti method

Salotti (Salotti, 2002) proposed a method based on the search of the shortest path in a graph using A^* algorithm, to obtain optimal polygonal approximations of a boundary, minimizing the global ISE of the approximation.

If the A^* algorithm is applied directly to solve this problem, its computational time is similar to algorithm based on dynamic programming due to the cost of the management of the graph and node sorting. In order to reduce the search, Salotti (Salotti, 2002) proposed two improvements:

- To obtain a first rough polygonal approximation to estimate the value of a threshold on the maximum global error. Thus, nodes which cannot lead to optimal solutions are pruned. This rough polygonal approximation is obtained by using a suboptimal method with low computational complexity.
- To stop the exploration of successors of the shortest path in the graph as soon as possible. For this reason, Salotti proposed a simple solution to stop the exploration using a lower-bound calculated from the linear regressions y/x and x/y to estimate least-square errors. When this lower bound is greater than the maximum global error, this node cannot lead to optimal solution and it is pruned, and another node is tested. When the node that depicts the last point of the time series is selected, the algorithm finishes.

Using these improvements, Salotti managed to reduce the time complexity of the A^* algorithm. In this case the computational complexity is $O(MN^2)$, where M is the number points of the polygonal approximation and N is the the number of points of the boundary. Since M is much smaller than N, the computational complexity is close to $O(N^2)$.

3.3. Improved optimal Salotti method

Carmona-Poyato et al. (2017) improved Salotti method by calculating the lower-bound using the minimum error of the best line segment approximating a set S of consecutive points $(P_i, \ldots, P_j)$ instead of using the linear regressions y/x and x/y to estimate least-square errors. This method is known as *total least squares* or *orthogonal regression*. By using this method, the time taken to calculate the lower-bound is halved. This improvement reduces the computational time of the original Salotti method about 16%. This method was adapted in (Duran-Rosal et al., 2018b) for time series segmentation, obtaining optimal segmentations in all cases. However, its computational time is high compared to other methods based on particle swarm optimisation and used in (Duran-Rosal et al., 2018b).

3.4. Our proposal (OSTS method)

Our proposal can be explained in the following steps:

1. Adapted Pikaz method (Pikaz and Dinstein, 1995) is applied. Thus, the accumulated values of ISE for each point of the time series are calculated. These values are used as pruning values in next step. The computational complexity of Pikaz method is $O(n \log n)$, thus, for all time series used in the experiments, the computational time obtained is in the order of milliseconds.

2. An improved version of (Carmona-Poyato et al., 2017) is used. This improvement consists of:
    - As in (Carmona-Poyato et al., 2017), the proposal is based on the search of the shortest path in a graph using A^* algorithm.
    - In this graph, each node contains information about a point in the time series (index), its possible position in the final segmentation (position), its predecessor node (previous) and the ISE value accumulated at that point.
    - When the node of the graph of minimum accumulated error, not yet evaluated, is selected, it is compared to the pruning value corresponding to this point, obtained in the first step. If the ISE value for this node is greater than the pruning value, the node is pruned because it cannot obtain an optimal solution. Otherwise, the node is a candidate to be part of the optimal solution (cut point). It should be noted that the error corresponding to that point of the node and in that position is optimal because it has been obtained by using A^* algorithm.
    - The algorithm finishes when the candidate node to be part of the optimal solution contains the last point of the time series.

The pseudocode and a graphical example of the proposed method are shown summarized in Table 1 and Figure 1, respectively.

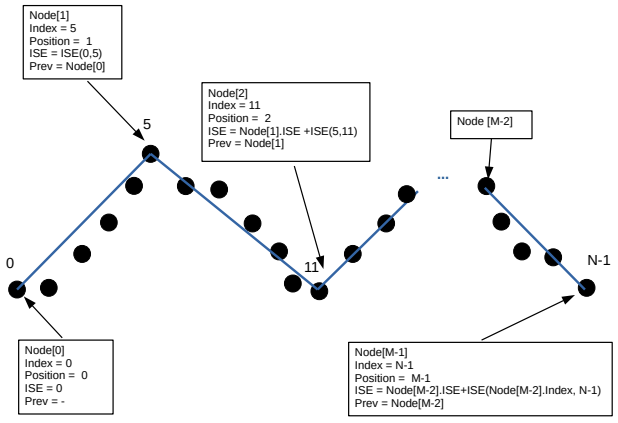

Fig. 1. Graphical example of the proposed method. In this case, the Cut Points are $0, 5, 11, \ldots, N-1$. M is the number of Cut Points and N is the number of points of the time series

In this case the computational complexity is $O(MN^2)$, where M is the number of cut points and N is the the number of points of the time series. Since M is much smaller than N, the computational complexity is close to $O(N^2)$.

The tests carried out have shown that, with this improvement, the computational time is reduced between 65% and 97% compared to the version used in (Duran-Rosal et al., 2018b).

Table 1. Pseudocode for the proposed method

OSTS method
INPUT: Time series (TS), Accumulated ISE for all points of TS obtained using Pikaz method ($AccumulatedIse$)
OUTPUT: List of Cut Points of the optimal segmentation (L_{cp})
Local Data Structures: List of candidate nodes (L_c), List of evaluated nodes (L_e), Initial Node (N_0).
Pseudo-code
Initialise N_0 using the first point of the time series
$L_c.insert(N_0)$
$minimum = L_c.getNodeMinimumIse()$
Repeat
  $L_e.insertNode(minimum)$
  For each Node N_i possible candidate successor of minimum
    $N_i.index \leftarrow i$ //*i is the original position of N in the time series*
    $N_i.position \leftarrow minimum.position + 1$
    $N_i.ISE \leftarrow minimum.ISE + ISE(minimun.index, N_i.index)$
    $N_i.previous \leftarrow minimum$
    If $N_i.ISE > AccumulatedIse(N_i.index)$
      continue //N_i *is pruned*
    End-If
    If $N_i \notin L_c$
      $L_c.insert(N_i)$
    Else if $N_i.ISE \leq N_i.oldISE$ // *old value of ISE for the i-th node*
      $L_c.update(N_i)$
    Else
      //N_i *is not updated*
    End-If
  End-For
  $minimum \leftarrow L_c.getNodeMinimumISE()$
Until $minimum.index = LastPointTimeSeries$
While $minimum.index \neq 0$
  $L_{cp}.insert(minimum.index)$
  $minimum \leftarrow minimum.previous$
End-While
End-Pseudo-code

4. Experiments and results

This section shows the time series considered to evaluate the different methods, the experimental setting and the results obtained.

4.1. Datasets used

The performance of the *OSTS* method has been evaluated in several synthetic and real-world time series collected from public repositories, to test its robustness in different scopes of application. The time series used are the following:

- UCR time series: four datasets from the UCR Time Series Classification Archive has been selected (Dau et al., 2018). The time series selected are Hand Outlines, with a total of 8127 points, Mallat, and StarLightCurves, all of them with 8192 observations.
- Donoho-Johnstone time series (Donoho and Johnstone, 1994, 1995). The Donoho-Johnstone benchmarks are 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.
- Stock prices time series from financial applications: five different indexes have been selected. IBEX35, 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.

- Wave height time series (Hs): four time series of significant wave height collected from buoys of the National Data Buoy Center of the USA have been used (NOAA, 2015). 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).

- Arrhythmia dataset contains cardiology data which belongs to the PhysioBank ATM of the MIT BIH Arrhythmia dataset (Moody and Mark, 2001). 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 2.

4.2. Error measures

To measure the effectiveness of the *OSTS* method, the following error measures have been used: 1) the root mean square of ISE divided by the number of points of the time series (n), called *RSME*. This is the main measure since it has been used as objective function to optimize the error. 2) Average error of the average errors within each segment ($error_{AA}$). 3)Average error of the maximum errors within each segment ($error_{AM}$). 4) Maximum error of the average errors within each segment ($error_{MA}$). 5) Maximum error of the maximum errors within each segment ($error_{MM}$). The last four measures were used in (Fuchs et al., 2010).

4.3. Experimental design

The experimental design is presented in this subsection. In order to evaluate the performance of the *OSTS* method, the following methods has been compared:

- An adapted and improved version of the Salotti method (Duran-Rosal et al., 2018b), (Carmona-Poyato et al., 2017).

- An adapted version of the Bottom-Up method (Keogh et al., 2004), described in section 2.

- An adapted version of the Top-Down method (Keogh et al., 2004), described in section 2.

- An adapted and improved version of the Bottom-Up method (Sarker et al., 2017), described in section 2.

- A genetic algorithm (GA) with crossover and mutation probabilities set to $p_c = 0.8$ and $p_m = 0.2$, respectively.

- A basic particle swarm optimisation algorithm (PSO) with the following specific parameters: initial velocities of the particles are set to values close to zero, the inertia coefficient (w) is set to 0.72, and the constant parameters (C_1 and C_2) are fixed to 1.49.

- The exploiter version of the barebones PSO (BBePSO) proposed by Kennedy (2003) has also been tested.

- The Dynamic exploiting barebones particle swarm optimisation algorithm (DBBePSO) proposed in (Duran-Rosal et al., 2018b). The λ parameter is set to 1 at the beginning, and it linearly decreases to 0.1. No other parameters must be set.

- According to Duran-Rosal et al. (2016), a 40% of the cut points, found by the GA, BBePSO, and DBBePSO are fine-tuned according to the method presented in (Duran-Rosal et al., 2018a) resulting in the HGA, HBBePSO, and HDBBePSO methods, respectively.

The number of cut points is set to a 2.5% of the total number of points of the time series.

4.4. Results and Discussion

Tables 2 and 3 show the obtained values of RSME for all methods.

There is a simple result for the first five methods, because they are deterministic. The remaining methods are not deterministic, and the table shows the mean and the standard deviation of the 30 runs using different seeds. The mean ranking of each algorithm is also included, where 1 is the best method and 13 for the worst one. In this case, the best methods are the *OSTS* method and the improved Salotti method (Carmona-Poyato et al., 2017), because they are optimal. In most cases, the error obtained in both is much smaller than the other methods.

The second method is Sarker method and its RSME value is 30 percent higher in most cases.

Table 6 show the mean ranking of all methods considering the values of $error_{AA}, error_{AM}, error_{MA}$ and $error_{MM}$. This table shows that the best method for $error_{AA}, error_{AM}$ and $error_{MA}$ is the *OSTS* method and Sarker method is the second one. However, for $error_{AM}$ Sarker method is the best and the proposed method is the second one.

Figure 3 shows a graphical comparison between the *OSTS* method and the Sarker method, the second best method considering the error measures that have been used. Since the size of the series is very large, only the last 750 points of the IBEX time series have been used. The figure shows how the *OSTS* method produces better segmentation than the Sarker method.

Tables 4 and 5 show run time for the deterministic methods and the mean and the standard deviation values of 30 run times for the remaining methods. The run times have been measured in seconds. All the experiments were run using an Intel(R) Xeon(R) CPU E5-2620 v3 at 2.40 GHz with 32 GB of RAM and 24 cores. The results show that the worst computational times are obtained by the improved Salotti method (Carmona-Poyato et al., 2017) because is optimal. However, the *OSTS* method has greatly reduced the time of this method. It is important to highlight that:

- The *OSTS* method has outperformed to GA method and HGA method in the ranking.

- For some of the time series used, it has improved some of the remaining methods. For example, see Hand Outlines, Mallat, Star Light Curves and Donoho-Johnstone.

Table 2. RMSE values (1/2) obtained by all the algorithms in each time series. Proposed, Improved Salotti, Top-Down, Bottom-Up and Sarker are deterministic and they are run once, while GA, HGA, PSO, HPSO, BbePSO, HBBePSO, DBBePSO, HDBBePSO are run 30 times with different seeds due to their stochastic nature (Mean ± Standard deviation). The mean rankings of all algorithms are also included.

Time series \method	Proposed	Improved Salotti	Bottom-Up	Sarker	Top-Down	GA	HGA
Hand Outlines	**0.0036**	0.0036	0.005	0.0038	0.006	0.030 ± 0.004	0.006 ± 0.000
Mallat	**0.072**	0.072	0.097	0.076	0.502	0.338 ± 0.014	0.167 ± 0.009
StarLightCurves	**0.011**	0.011	0.016	0.012	0.026	0.064 ± 0.005	0.024 ± 0.001
Donoho-Johnstone	**2.218**	2.218	2.639	2.285	3.466	3.119 ± 0.078	2.642 ± 0.062
IBEX	**149.962**	149.962	210.321	206.774	269.801	279.366 ± 9.788	203.896 ± 3.633
BBVA	**0.236**	0.236	0.320	0.250	0.405	0.447 ± 0.012	0.325 ± 0.008
DEUTSCHE	**1.421**	1.421	2.032	1.496	2.318	2.570 ± 0.097	1.883 ± 0.057
SAN PAOLO	**0.080**	0.080	0.112	0.086	0.136	0.158 ± 0.005	0.111 ± 0.003
SO Genéralé	**1.598**	1.598	2.292	1.710	2.472	2.807 ± 0.081	2.112 ± 0.054
B46001	**0.799**	0.799	1.088	0.840	1.011	1.158 ± 0.010	0.989 ± 0.011
B46075	**0.822**	0.822	1.145	0.866	1.056	1.202 ± 0.013	1.040 ± 0.014
B41043	**0.295**	0.295	0.426	0.311	0.449	0.487 ± 0.006	0.398 ± 0.008
B41044	**0.292**	0.292	0.419	0.308	0.425	0.404 ± 0.003	0.357 ± 0.005
Arrhytmia	**0.022**	0.022	0.032	0.023	0.091	0.485 ± 0.006	0.394 ± 0.010
Mean rankings ($\bar{r}$)	1.000	1.000	6.600	3.200	9.600	11.400	6.500

Table 3. RMSE values (2/2) obtained by all the algorithms in each time series. Proposed, Improved Salotti, Top-Down, Bottom-Up and Sarker are deterministic and they are run once, while GA, HGA, PSO, HPSO, BbePSO, HBBePSO, DBBePSO, HDBBePSO are run 30 times with different seeds due to their stochastic nature (Mean ± Standard deviation). The mean rankings of all algorithms are also included.

Time series \method	PSO	HPSO	BbePSO	HBBePSO	DBBePSO	HDBBePSO
Hand Outlines	0.029 ± 0.016	0.006 ± 0.000	0.012 ± 0.001	0.006 ± 0.000	0.007 ± 0.000	0.005 ± 0.000
Mallat	0.314 ± 0.050	0.162 ± 0.007	0.279 ± 0.020	0.166 ± 0.008	0.206 ± 0.013	0.149 ± 0.006
StarLightCurves	0.063 ± 0.019	0.024 ± 0.001	0.044 ± 0.003	0.023 ± 0.001	0.032 ± 0.002	0.021 ± 0.001
Donoho-Johnstone	3.469 ± 0.147	2.705 ± 0.072	3.127 ± 0.085	2.646 ± 0.070	2.976 ± 0.084	2.608 ± 0.050
IBEX	314.513 ± 30.383	207.619 ± 3.981	281.477 ± 10.900	203.527 ± 5.222	237.904 ± 7.236	198.671 ± 3.811
BBVA	0.463 ± 0.037	0.325 ± 0.008	0.428 ± 0.013	0.322 ± 0.008	0.366 ± 0.014	0.313 ± 0.008
DEUTSCHE	2.847 ± 0.247	1.938 ± 0.074	2.565 ± 0.089	1.921 ± 0.067	2.173 ± 0.081	1.828 ± 0.045
SAN PAOLO	0.159 ± 0.015	0.111 ± 0.004	0.147 ± 0.004	0.111 ± 0.002	0.124 ± 0.005	0.105 ± 0.002
SO Genéralé	3.042 ± 0.266	2.143 ± 0.062	2.782 ± 0.086	2.136 ± 0.045	2.456 ± 0.099	2.074 ± 0.050
B46001	1.250 ± 0.018	0.987 ± 0.010	1.170 ± 0.009	0.980 ± 0.007	1.142 ± 0.020	0.977 ± 0.006
B46075	1.317 ± 0.019	1.049 ± 0.010	1.236 ± 0.011	1.036 ± 0.008	1.208 ± 0.020	1.032 ± 0.009
B41043	0.528 ± 0.007	0.397 ± 0.006	0.490 ± 0.005	0.393 ± 0.005	0.464 ± 0.014	0.387 ± 0.005
B41044	0.526 ± 0.006	0.392 ± 0.008	0.489 ± 0.006	0.388 ± 0.008	0.467 ± 0.013	0.384 ± 0.006
Arrhytmia	0.109 ± 0.010	0.053 ± 0.003	0.085 ± 0.004	0.050 ± 0.002	0.066 ± 0.004	0.047 ± 0.002
Mean rankings ($\bar{r}$)	12.600	7.000	11.100	6.100	9.500	4.300

Table 4. Computational time (1/2) in seconds obtained by all the algorithms in each time series. Proposed, Improved Salotti, Top-Down, Bottom-Up and Sarker are deterministic and they are run once, while GA, HGA, PSO, HPSO, BbePSO, HBBePSO, DBBePSO, HDBBePSO are run 30 times with different seeds due to their stochastic nature (Mean ± Standard deviation). The mean rankings of all algorithms are also included.

Time series \method	Proposed	Improved Salotti	Bottom-Up	Sarker	Top-Down	GA	HGA
Hand Outlines	32.834	810.559	10.750	11.630	**9.620**	55.496 ± 4.976	55.670 ± 4.982
Mallat	33.868	732.615	23.760	25.730	**21.780**	72.943 ± 6.602	73.125 ± 6.605
StarLightCurves	50.274	1056.290	25.020	28.110	**21.470**	71.913 ± 7.030	72.086 ± 7.036
Donoho-Johnstone	2.825	10.855	1.780	1.930	**1.240**	4.789 ± 0.386	4.829 ± 0.388
IBEX	42.948	248.947	7.130	10.260	**4.600**	25.967 ± 0.731	26.082 ± 0.734
BBVA	15.149	85.047	3.630	4.080	**2.470**	14.631 ± 0.302	14.720 ± 0.304
DEUTSCHE	10.968	95.779	3.590	4.230	**2.510**	14.422 ± 0.473	14.508 ± 0.477
SAN PAOLO	12.681	86.459	2.990	3.200	**2.530**	14.623 ± 0.757	14.710 ± 0.761
SO Genéralé	13.505	108.374	2.480	3.280	**2.480**	14.322 ± 0.848	14.405 ± 0.853
B46001	297.875	1092.100	13.340	17.400	**11.140**	60.387 ± 2.388	60.586 ± 2.394
B46075	205.776	627.355	**7.270**	12.750	7.540	39.696 ± 0.644	39.850 ± 0.647
B41043	157.429	504.465	**7.260**	10.740	7.510	39.739 ± 0.391	39.891 ± 0.395
B41044	137.891	760.920	**7.230**	11.450	7.790	58.311 ± 2.428	58.508 ± 2.430
Arrhytmia	46.912	919.173	**11.490**	13.840	12.410	39.870 ± 1.722	40.033 ± 1.735
Mean rankings ($\bar{r}$)	9.100	13.000	1.600	3.100	**1.400**	10.200	11.200

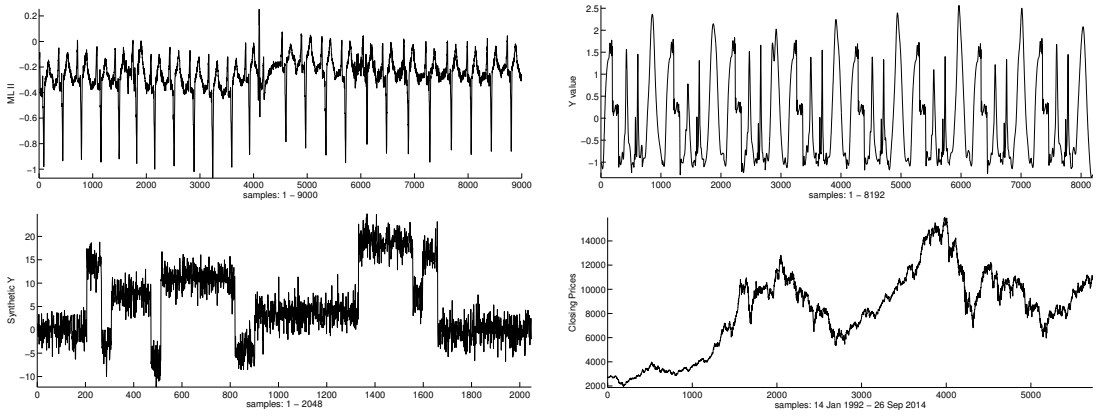

Fig. 2. Some representatives time series. From left to right and from top to bottom: Arrhythmia, Mallat, Donoho-Johnstone and IBEX

Table 5. Computational time (2/2) in seconds obtained by all the algorithms in each time series. Proposed, Improved Salotti, Top-Down, Bottom-Up and Sarker are deterministic and they are run once, while GA, HGA, PSO, HPSO, BbePSO, HBBePSO, DBBePSO, HDBBePSO are run 30 times with different seeds due to their stochastic nature (Mean ± Standard deviation). The mean rankings of all algorithms are also included.

Time series \method	PSO	HPSO	BbePSO	HBBePSO	DBBePSO	HDBBePSO
Hand Outlines	17.368 ± 2.533	17.538 ± 2.541	50.477 ± 3.671	50.656 ± 3.677	48.708 ± 4.526	48.898 ± 4.531
Mallat	34.316 ± 5.244	34.506 ± 5.253	61.966 ± 3.655	62.139 ± 3.658	64.456 ± 6.446	64.636 ± 6.452
StarLightCurves	33.530 ± 4.151	33.701 ± 4.155	65.068 ± 5.185	65.237 ± 5.190	64.551 ± 6.421	64.731 ± 6.424
Donoho-Johnstone	1.935 ± 0.392	1.976 ± 0.392	3.479 ± 0.301	3.518 ± 0.301	3.579 ± 0.287	3.619 ± 0.289
IBEX	9.217 ± 1.104	9.341 ± 1.106	22.702 ± 1.169	22.821 ± 1.173	22.843 ± 1.790	22.966 ± 1.797
BBVA	4.500 ± 0.397	4.590 ± 0.399	12.115 ± 0.762	12.200 ± 0.766	12.287 ± 1.152	12.374 ± 1.158
DEUTSCHE	4.748 ± 0.489	4.839 ± 0.492	11.797 ± 0.534	11.880 ± 0.536	11.812 ± 0.580	11.897 ± 0.584
SAN PAOLO	4.492 ± 0.316	4.582 ± 0.318	12.310 ± 1.056	12.397 ± 1.060	11.858 ± 0.393	11.942 ± 0.395
SO Genéralé	4.363 ± 0.290	4.450 ± 0.291	11.880 ± 0.935	11.961 ± 0.939	11.751 ± 1.041	11.835 ± 1.048
B46001	21.950 ± 2.033	22.139 ± 2.036	53.613 ± 4.494	53.785 ± 4.502	55.966 ± 4.809	56.145 ± 4.816
B46075	15.097 ± 2.327	15.257 ± 2.325	35.825 ± 1.359	35.965 ± 1.361	36.503 ± 1.954	36.644 ± 1.958
B41043	15.017 ± 1.549	15.173 ± 1.551	34.401 ± 0.921	34.537 ± 0.923	34.377 ± 0.932	34.513 ± 0.935
B41044	15.294 ± 1.939	15.455 ± 1.953	34.230 ± 0.252	34.365 ± 0.252	40.238 ± 3.521	40.390 ± 3.531
Arrhytmia	24.916 ± 2.056	25.116 ± 2.060	54.187 ± 3.456	54.369 ± 3.468	54.186 ± 4.159	54.378 ± 4.175
Mean rankings ($\bar{r}$)	4.000	5.000	7.300	8.400	7.600	8.900

- The *OSTS* method, taking into account the computational time, is near of HDBBePSO method, that is the third one taking into account the RSME value.

5. Conclusions

The conclusions of this work can be summarized as follows:

- The present work proposes an optimal method, called *OSTS*, of data compression applied to time series segmentation.
- The *OSTS* method uses an improved and adapted version of Salotti algorithm (Carmona-Poyato et al., 2017; Duran-Rosal et al., 2018b).
- In order to reduce the computational time, the *OSTS* method uses a bottom-up suboptimal method (Pikaz and Dinstein, 1995) to obtain pruning values.
- The results show that:

1. The *OSTS* method is optimal, and reduces the computational time of the previous version between the 65% and 97%.
2. The computational time of the *OSTS* method is less than the computational time of GA and HGA methods.
3. In some series (Hand Outlines, Mallat, Star Light Curves and Donoho-Johnstone), the computational time of the *OSTS* method is less than the computational time of the methods based on metaheuristics.
4. The *OSTS* method, taking into account the computational time, is near of HDBBePSO method, that is the third one considering the RSME value.

.

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.

Table 6. Mean rankings of all algorithms for the values of $error_{AA}$, $error_{AM}$, $error_{MA}$ and $error_{MM}$. Proposed, Improved Salotti, Top-Down, Bottom-Up and Sarker are deterministic and they are run once, while GA, HGA, PSO, HPSO, BbePSO, HBBePSO, DBBePSO, HDBBePSO are run 30 times with different seeds due to their stochastic nature

error \method	Proposed	Salotti	Bottom-Up	Sarker	Top-Down	GA	HGA	PSO	HPSO	BbePSO	HBBePSO	DBBePSO	HDBBePSO
$error_{AA}$	**1.0**	1.0	9.9	3.4	4.4	10.8	6.4	11.5	7.9	11.4	6.8	9.9	5.6
$error_{AM}$	**2.5**	2.5	8.6	2.7	6.1	11.3	7.9	10.0	6.6	9.2	6.2	9.2	7.2
$error_{MA}$	**1.4**	1.4	9.6	3.4	4.1	10.3	6.7	11.2	8.2	11.4	6.6	10.0	5.8
$error_{MM}$	1.9	1.9	6.6	**1.7**	10.5	11.5	6.3	11.8	6.6	10.5	5.7	8.8	6.3

Fig. 3. Graphical comparison between the proposal and the Sarker method, using the last 750 points of the IBEX time series. Original is IBEX time series. The values of $error_{AA}, error_{AM}, error_{MA}$ and $error_{MM}$ are (108,186,305,570) for proposed method and (139,286,352,685) for Sarker method

References

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

Bajcsy, R., Solina, F., Gupta, A., 1990. Analysis and Interpretation of Range Images. Springer-Verlag New York, Inc., New York, NY, USA. doi:10.1007/978-1-4612-3360-2_4.

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

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

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

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

Duncan, S., Bryant, G., 1996. A new algorithm for segmenting data from time series, in: Proceedings of 35th IEEE Conference on Decision and Control, pp. 3123–3128. doi:10.1109/CDC.1996.573607.

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

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

Duran-Rosal, A., Gutierrez-Pena, P.A., Martinez-Estudillo, F.J., Hervas-Martinez, C., 2016. Time series representation by a novel hybrid segmentation algorithm, in: Hybrid Artificial Intelligent Systems, pp. 163–173. doi:10.1007/978-3-319-32034-2_14.

Ferreira, L.N., Zhao, L., 2016. Time series clustering via community detection in networks. Information Sciences 326, 227–242. doi:10.1016/j.ins.2015.07.046.

Fuchs, E., Gruber, T., Nitschke, J., Sick, B., 2010. Online segmentation of time series based on polynomial least-squares approximations. IEEE Transactions on Pattern Analysis and Machine Intelligence 32, 2232–2245. doi:10.1109/TPAMI.2010.44.

Gonzalez-Vidal, A., Barnaghi, P., Skarmeta, A., 2018. Beats: Blocks of eigenvalues algorithm for time series segmentation. IEEE Transactions on Knowledge and Data Engineering 30, 2051–2064. doi:10.1109/TKDE.2018.2817229.

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

Kennedy, J., 2003. Bare bones particle swarms, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706), pp. 80–87. doi:10.1109/SIS.2003.1202251.

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

Lin, J., Keogh, E., Wei, L., Lonardi, S., 2007. Experiencing sax: A novel symbolic representation of time series. Data Mining and Knowledge Discovery 15, 107–144. doi:10.1007/s10618-007-0064-z.

Masood, A., 2008. Optimized polygonal approximation by dominant point deletion. Pattern Recognition 41, 227–239. doi:10.1016/j.patcog.2007.05.021.

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

Nikolaou, A., Gutierrez-Pena, P., Duran-Rosal, A., Dicaire, I., Fernandez-Navarro, F., Hervas-Martinez, C., 2015. Detection of early warning signals in paleoclimate data using a genetic time series segmentation algorithm. Climate Dynamics 44, 1919–1933. doi:10.1007/s00382-014-2405-0.

NOAA, 2015. National buoy data center. http://www.ndbc.noaa.gov/.

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

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

Sangeeta, R., Geeta, S., 2012. Recent techniques of clustering of time series data: A survey. International Journal of Computer Applications 52, 1–9. doi:10.5120/8282-1278.

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

Sarker, I.H., Colman, A., Han, J., 2019. Recencyminer: mining recency-based personalized behavior from contextual smartphone data. Journal of Big Data 6. doi:10.1186/s40537-019-0211-6.

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

Zhao, J., Itti, L., 2016. Classifying time series using local descriptors with hybrid sampling. IEEE Transactions on Knowledge and Data Engineering 28, 623–637. doi:10.1109/TKDE.2015.2492558.