A hybrid dynamic exploitation barebones particle swarm optimisation algorithm for time series segmentation

Antonio M. Durán-Rosal*, Pedro A. Gutiérrez, Ángel Carmona-Poyato, César Hervás-Martínez

Department of Computer Science and Numerical Analysis, University of Córdoba, Córdoba, Spain

Abstract

Large time series are difficult to be mined and preprocessed, hence reducing their number of points with minimum information loss is an active field of study. This paper proposes new methods based on time series segmentation, including the adaptation of the particle swarm optimisation algorithm (PSO) to this problem, and more advanced PSO versions, such as barebones PSO (BBPSO) and its exploitation version (BBePSO). Moreover, a novel algorithm is derived, referred to as dynamic exploitation barebones PSO (DBBePSO), which updates the importance of the social and cognitive components throughout the generations. All these algorithms are further improved by considering a final local search step based on the combination of two well-known standard segmentation algorithms (Bottom-Up and Top-Down). The performance of the different methods is evaluated using 15 time series from various application fields, and the results show that the novel algorithm (DBBePSO) and its hybrid version (HDBBePSO) outperform the rest of segmentation techniques.

Keywords:
Time series size reduction, time series segmentation, particle swarm optimisation, hybrid algorithm

1. Introduction

Recently, time series data mining (TSDM) has become an important field of research in science and engineering [1, 2]. Time series can be obtained from different areas, such as climate [3], hydrology [4], finances [5], satellite images [6], etc. They are used for different tasks depending on the objective of the researchers and the application areas, e.g. classification [7, 8], forecasting [9, 10], tipping point detection [11], clustering [12], similarity assessment [13, 14] or segmentation [15]. Specifically, time series segmentation is an important task, which consists of cutting the time series in some specific points trying to achieve different objectives, which are generally related to two points of view.

Firstly, time series segmentation can be used to discover useful patterns or segments in time series. Chung et al. [16] proposed a genetic algorithm for this purpose, using the similarities between the segments for optimising the cut points. Tseng et al. [17] combined a genetic algorithm with a clustering procedure and considered the discrete wavelet transformation (DWT) for the representation of the segments. The genetic algorithm proposed in [11] is aimed to characterise tipping points (TPs) and analyse the common patterns which occur before them, in order to create early warning signals in paleoclimate time series. Furthermore, a full analysis of different metrics for clustering evaluation and a first approximation to forecast TPs using the patterns previously identified were made in [3]. Fuzzy segmentation of multivariate time series was approached by a modified Gath-Geva clustering algorithm in [18], and an online recursive fuzzy clustering for indirect sequence clustering was proposed in [19]. Anomaly detection has been widely analysed for signal processing with the aim of locating abrupt changes along the time series [20]. There are many more applications of this kind of time series segmentation, such as the detection of important events in stock price time series [9, 21] or the detection and prediction of wave height extreme events combining a genetic algorithm with artificial neural networks [22].

On the other hand, the second group of time series segmentation algorithms tries to tackle the difficulty of processing and mining large time series. Their large amount of data (i.e. their high dimensionality) makes them very difficult to analyse. Because of this reason, and considering the fact that data mining is constrained by three types of limited resources (time, memory and sample size), different algorithms have been proposed with the aim of reducing the dimensionality or the number of points of time series. In the literature, time series segmentation techniques are also called time series representation procedures. These methods reduce the dimension of a given time-series by transforming it into a new representation space [23]. In general, TSDM tasks can be classified as first-hand processing (i.e. dimensionality reduction) or second-hand processing (further analysis of time series). Time series representation methods are first-hand processing algorithms, being useful for reducing the number of points of the time series while keeping their fun-

*Corresponding author at: Department of Computer Science and Numerical Analysis, University of Córdoba, Rabanales Campus, Albert Einstein Building 3rd Floor, 14071 Córdoba, Spain. Tel.: +34 957 218 349; Fax: +34 957 218 630. E-mail addresses: i92duroa@uco.es, pagutierrez@uco.es, chervas@uco.es

damental characteristics [24]. In this context, the authors in [25, 26] proposed a method based on dividing the time series using previously identified change points and represented the segments with suitable approximations. Piecewise linear approximation (PLA) is a global term referring to all the algorithms which reduce the number of points in the time series with a minimum information loss, based on linear interpolations or regressions [27]. Top-Down and Bottom-Up approaches proposed by Keogh et al. [27] are two simple PLA algorithms, based on iteratively reducing the approximation error. There are some other representations algorithms, such as adaptative piecewise constant approximation (APCA) [28] or symbolic aggregate approximation (SAX) [29].

The work presented in this paper is focused on the second group, specifically on PLA representation methods, whose objective is to reduce the number of elements in time series with minimum information loss. For this purpose, we use a modification of a particle swarm optimisation algorithm (PSO) [30] for segmenting time series, considering also two different related versions, barebones PSO (BBPSO) and exploiting barebones PSO (BBePSO). PSO is an evolutionary algorithm which simulates the social and cognitive behaviour of a set of particles, such as birds of fish when looking for food. In this way, PSO optimises problems considering a set of candidate solutions, denoted as particles (in our case, segmented time series), which move along the search space. In PSO, the social component refers to the best global position found by the algorithm, while the cognitive one is the best solution found by the individual particle. In general, PSO can be more easily adapted to the specific problem being tackled, as fewer parameters have to be configured when compared to other metaheuristics, such as genetic algorithms or ant colony systems. On the other hand, BBPSO avoids the use of velocities and, instead, considers a normal distribution to decide whether the update should take the best global position into account or the best local one [31]. Finally, BBePSO adds an exploting component to BBPSO, improving convergence [31]. PSO has been applied in many real problems, including hydrology prediction [32], video tracking [33], power system state estimation [34], etc.

In standard PSO, and also in BBPSO, and BBePSO, the importances of the social component (exploration) and the cognitive component (exploitation) are not updated during the generations. In this paper, we propose a new formulation, where the social component is more important at the beginning of the evolution, while the cognitive component is more important at the end, resulting in that we call dynamic BBePSO (DBBePSO).

On the other hand, evolutionary algorithms (EAs) are able to perform a global multi-point search, converging to high quality areas. In this sense, they are considered robust heuristics that can be applied in different problems. The main problem with EAs is that they are not good at finding the precise optimum in these high-quality areas [35]. To solve this issue, several authors combine EAs with a local search (LS) procedure to improve the best solutions. The idea is to combine the advantages of the EA (global explorer) and the advantages of the LS (local exploiter), resulting in hybrid algorithms. The hybridisation can be made in different ways, which are very important in terms of accuracy and computational cost. Some of the strategies previously used include the multi-start approach, the Lamarckian learning, the Baldwinian learning, the partial Lamarckianism and the process of random linkage [36, 37, 38]. In this way, we combine the previously presented algorithms with a LS procedure, consisting in removing a number of cut points with a Bottom-Up methodology and, then, adding the same number of cut points using the Top-Down procedure [27]. All the algorithms are applied to the segmentation of several time series in the experimental section of the paper, and hybrid DBBePSO obtains very good results which outperform the state of the art algorithms considered.

The rest of the paper is organised as follows: Section 2 briefly presents the main parts of the PSO, BBPSO, and BBePSO algorithms. Section 3 describes the new PSO proposal, while Section 4 includes the different considerations needed for adapting all the algorithms for time series segmentation. Section 5 shows the considered time series, which are extracted from real-world applications and different public repositories, the experimental setting and the statistical analysis of the results obtained. Finally, the paper is concluded in Section 6.

2. Particle Swarm Optimisation algorithm and its advanced versions

The particle swarm optimisation (PSO) [30] is an evolutionary-type algorithm for search and optimisation, based on the simulation of a swarm of particles, i.e., birds or fish, looking for food. In PSO, a swarm is formed by a set of P particles in a D-dimensional space, being D the length of the particles. Each particle i is a candidate solution of the studied problem, and it is represented by the following characteristics at iteration t: the current position of the particle $\mathbf{x}_i^t$, the current velocity of the particle $\mathbf{v}_i^t$ and the best position found by the particle $\mathbf{p}_i^t$. The fitness function evaluates the quality of a particle $\mathbf{x}_i$ and is presented by $f(\mathbf{x}_i)$. The velocity of the particle represents the direction and the rate of change in the movement of the particle at iteration t, while the best position $\mathbf{p}_i^t$ is the value of the $\mathbf{x}_i$ visited by the particle resulting in the best fitness. Moreover, an array with the best global solution ($\mathbf{p}_\mathrm{g}^t$) is also stored, which is defined as $\mathbf{p}_\mathrm{g}^t = \arg\max_{\mathbf{p}} \left\{ f(\mathbf{p}_\mathrm{g}^{t-1}), f(\mathbf{p}_1^t), f(\mathbf{p}_2^t), \ldots, f(\mathbf{p}_P^t) \right\}$ (considering a maximisation problem). Thus, the evolution is possible due to the cooperation of the particles, considering the local best position $\mathbf{p}_i$ (cognitive component) and the global best position $\mathbf{p}_\mathrm{g}$ (social component).

For each iteration of a PSO algorithm, the velocity $\mathbf{v}_i$ is updated in the following way:

$$\mathbf{v}_i^t = w \cdot \mathbf{v}_i^{t-1} + \rho_1^t \cdot C_1 \cdot \left(\mathbf{p}_i^{t-1} - \mathbf{x}_i^{t-1} \right) + \rho_2^t \cdot C_2 \cdot \left(\mathbf{p}_g^{t-1} - \mathbf{x}_i^{t-1} \right), \tag{1}$$

where w is the inertia weight, ρ_1^t, ρ_2^t are uniform random values obtained at iteration t, $\rho_1,\ \rho_2 \sim U(0, 1)$, and C_1, C_2 are the acceleration constants. The w parameter controls the impact of the memory with respect previous velocities. The cognitive component ($\mathbf{p}_i - \mathbf{x}_i$) represents the experience of the particle with respect to its best-found solution, while the social component

$(\mathbf{p}_g - \mathbf{x}_i)$, represents the experience with respect to the global best solution. The position of the particles is then updated using the expression:

$$\mathbf{x}_i^t = \mathbf{x}_i^{t-1} + \mathbf{v}_i^t. \tag{2}$$

Finally, the individual best position $\mathbf{p}_i$ and the global best position $\mathbf{p}_g$ are also updated in each iteration. $\mathbf{p}_i$ is updated as:

$$\mathbf{p}_i^t = \begin{cases} \mathbf{p}_i^{t-1} & \text{if} \quad f(\mathbf{x}_i^t) \leq f(\mathbf{p}_i^{t-1}), \\ \mathbf{x}_i^t & \text{if} \quad f(\mathbf{x}_i^t) > f(\mathbf{p}_i^{t-1}), \end{cases} \tag{3}$$

while, for the global best position, we have:

$$\mathbf{p}_{\mathrm{g}}^t = \arg\max_{\mathbf{p}} \left\{ f(\mathbf{p}_{\mathrm{g}}^{t-1}), f(\mathbf{p}_1^t), f(\mathbf{p}_2^t), \ldots, f(\mathbf{p}_P^t) \right\}. \tag{4}$$

The PSO algorithm is repeated during a predefined number of iterations or until velocity updates are near zero. The quality of the solutions (particles) is measured by a fitness function (in this section we have considered maximisation problems). Algorithm 1 illustrates the flowchart diagram of the PSO algorithm, which summarises the previously defined steps.

Algorithm 1 Pseudo-code for the PSO algorithm

Input: Valid values for the parameters controlling the PSO algorithm

Output: A solution with the best *fitness* value found by the algorithm

1: Initialise the swarm randomly
2: Evaluate the initial swarm
3: **while not** stop condition **do**
4: Update velocities (Eq. 1)
5: Update positions (Eq. 2)
6: Evaluate the new swarm
7: Update personal best positions (Eq. 3)
8: Update global best positions (Eq. 4)
9: **end while**
10: Return the best individual (final solution) from the swarm

2.1. Barebones PSO

One of the improved versions of PSO is the barebones PSO (BBPSO) [31]. This algorithm does not take into account the velocities to update the current position of the particles in the swarm. Instead, BBPSO replaces Equations 1 and 2 with the following expression for the j-th dimension:

$$x_{i,j}^t = N\left(\frac{p_{i,j}^{t-1} + p_{g,j}^{t-1}}{2}, |p_{i,j}^{t-1} - p_{g,j}^{t-1}| \right), \tag{5}$$

where $N(\mu, \sigma)$ is a normal random distribution with μ mean and σ standard deviation, and $i = 1, \ldots, P$, $j = 1, \ldots, D$. This equation is based on theoretical studies confirming that particles converge to a weighted average of the global and personal best positions [39]. In this way, each dimension of each particle is selected from a Gaussian distribution where the mean is the average value of the global and local best positions, and the difference between them is used as the standard deviation. This procedure allows taking large steps when the personal best positions are far away from the global best positions.

2.2. Exploiting barebones PSO

In [31], Kennedy also proposed an alternative version of the BBPSO, called exploiting barebones PSO (BBePSO), where the velocity and position updates are replaced with:

$$x_{i,j}^t = \begin{cases} N\left(\frac{p_{i,j}^{t-1} + p_{g,j}^{t-1}}{2}, |p_{i,j}^{t-1} - p_{g,j}^{t-1}| \right) & \text{if } U(0,1) < 0.5, \\ p_{i,j}^{t-1} & \text{otherwise.} \end{cases} \tag{6}$$

This equation establishes a 0.5 probability that the j-th dimension of the particle i changes to the corresponding personal best position. In this way, the BBePSO searches with a higher degree of exploitation than BBPSO. In general, this exploiting version outperforms other variants of PSO [40]. Unlike standard PSO, the barebones variants (BBPSO and BBePSO) do not need a value for the weight and the acceleration coefficients, so they are more suitable for those application problems where the value of the these parameters is difficult to be estimated.

3. Dynamic exploiting barebones PSO

In this work, a dynamic BBePSO (DBBePSO) algorithm is proposed, where the importance of the social and the cognitive components are updated along the generations.

As we mentioned before, DBBePSO updates the current positions of each particle ($\mathbf{x}_i$) in a similar way that BBePSO. However, in our proposal, the importance of the exploration and the exploitation are dynamically updated over the generations using a modified Gaussian distribution:

$$x_{i,j}^t = \begin{cases} N\left(\frac{p_{i,j}^{t-1} + p_{g,j}^{t-1}}{2}, \lambda |p_{i,j}^{t-1} - p_{g,j}^{t-1}| \right) & \text{if } U(0,1) < 0.5, \\ p_{i,j}^{t-1} & \text{otherwise.} \end{cases} \tag{7}$$

The novelty is the multiplicative parameter λ in the standard deviation of the distribution. It is known that evolutionary algorithms work better when the exploration is higher at the beginning but lower at the end [41]. To do so, λ is updated dynamically over the generations from an initial value of 1 to a final value of 0.1:

$$\lambda = \frac{0.9(L - l)}{L} + 0.1, \tag{8}$$

where L is the maximum number of evaluations allowed to the algorithm (stop criterion), and l is the current number of evaluations. As can be observed, when the number of evaluations is 0, then λ is 1.0; and it decreases to 0.1 when l is close to L. It is important to mention that the λ update is done at the beginning of each iteration t of the algorithm.

4. Adapting the algorithms for time series segmentation

4.1. Problem definition

Given a time series $Y = \{y_n\}_{n=1}^N$, the main goal is to split the time series by dividing the values into m consecutive segments, taking into account that the error approximation of these segments needs to be as lower as possible. In other words, from all the time indexes ($n = 1, \ldots, N$), a set of $m - 1$ cut points are selected, being presented in ascending order ($t_1 < t_2 < \cdots <$

t_{m-1}). In this way, the set of the resulting segments is composed by $s_1 = \{y_1, \ldots, y_{t_1}\}$, $s_2 = \{y_{t_1}, \ldots, y_{t_2}\}, \ldots, s_m = \{y_{t_{m-1}}, \ldots, y_N\}$, and the algorithm has to determine the values of the $m-1$ cut points. Note that the cut points are part of two segments, the precedent segment and the posterior one, while the rest of points belong to a single segment. In order to reduce the amount of information, each segment is approximated using linear interpolation between the initial and the final points (i.e. the cut points delimiting the segment).

It is important to mention that the search space is very large. Consequently, the use of evolutionary algorithms is proposed in this paper.

4.2. Particle representation

The position of a particle is represented by an array (chromosome) of real values ($\mathbf{x}_i$), where the length of the chromosome is the same that the number of segments minus one ($m-1$), i.e. the number of cut points. Each chromosome element $x_{i,j}$ stores a real value, which is rounded to the closest integer in order to obtain the value of the j-th cut point ($t_{i,j}$). For example, the chromosome of length 5, $\mathbf{x}_i = \{1.68, 5.76, 12.12, 15.30, 20.10\}$ corresponds to the following cut points, $\mathbf{t}_i = \{2, 6, 12, 15, 20\}$. An example of a particle for this problem is shown in Figure 1. It is important to note that the values of the chromosome need to be presented in ascending order (see section 4.5).

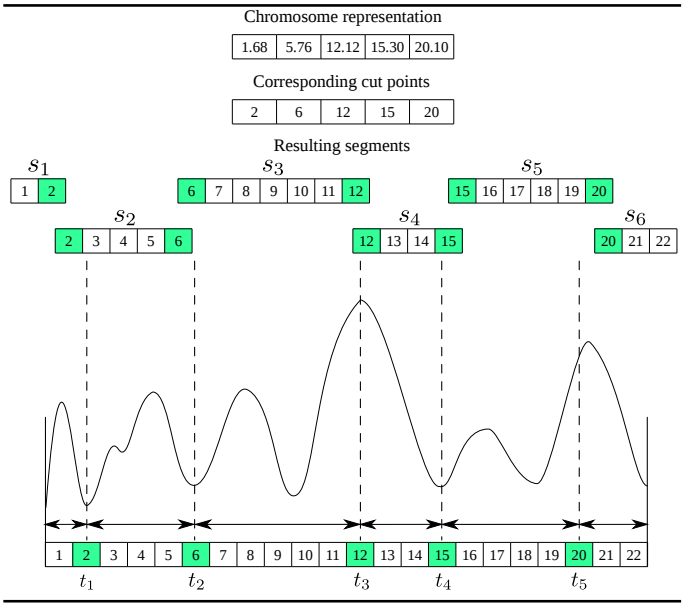


Figure 1: Chromosome representation: $\mathbf{x}_i = \{1.68, 5.76, 12.12, 15.30, 20.10\}$.

4.3. Initialisation of the swarm

The population of the swarm is a set of P arrays of real values with a length of $m-1$. In the initial population, the cut points are randomly selected taking into account that they must be subscripted in ascending order, and each cut point has to be unique (it is not possible to have two cut points with the same value). Note that the initial population is formed by integer values, but, during the generations, these positions are updated with real values.

4.4. Fitness evaluation

As we stated before, the main goal of this type of time series segmentation is to reduce the number of points without losing important information. For that, we optimise the error produced by the approximation with respect to the original time series values. Thus, the fitness function is defined as minimising the difference between each real value of the time series and its corresponding approximation. The approximation error of the n-th point of the time series in the swarm is defined as:

$$e_n(\mathbf{x}_i) = (y_n - \hat{y}_n(\mathbf{x}_i)), \tag{9}$$

where y_n is the real value of the n-th point in the time series, and $\hat{y}_n(\mathbf{x}_i)$ is the PLA approximation value obtained by a linear interpolation in the chromosome $\mathbf{x}_i$. The fitness function considered for the complete chromosome is the root mean square of the $e_n(\mathbf{x}_i)$ (RMSE), which is formally defined as:

$$RMSE(\mathbf{x}_i) = \sqrt{\frac{1}{N}\sum_{n=1}^{N} e_n^2(\mathbf{x}_i)}. \tag{10}$$

Due to the fact that this metric needs to be minimised, the final fitness function is $f = \frac{1}{1+RMSE(\mathbf{x}_i)}$, which is bounded in the interval $[0, 1]$. Figure 2 shows the evaluation process of the chromosome used in Figure 1.

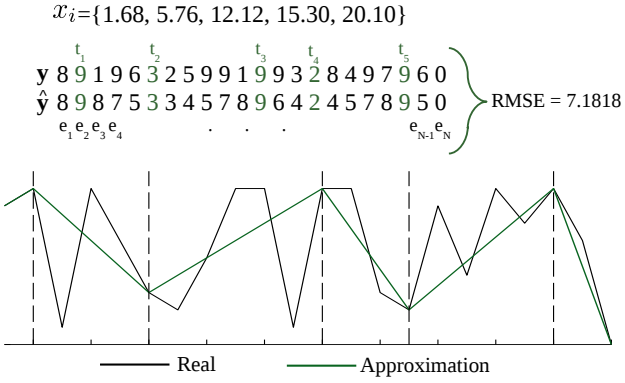


Figure 2: Example of evaluation.

4.5. Repair solutions

In time series segmentation, there are certain constraints that have to be satisfied to ensure proper solutions:

1. The first one is that the time index n must be presented in ascending order, and therefore the values of the chromosome ($x_{i,1} < x_{i,2} < \cdots < x_{i,m-1}$). After applying the position updates, it can be possible that a too large step is taken for one of the dimensions, making the value of the cut point to be higher than the next in the chromosome ($x_{i,k} > x_{i,k+1}$) or lower than the previous one ($x_{i,k} < x_{i,k-1}$). In order to avoid this problem, after updating the positions, the algorithm sorts the cut points of the chromosomes (particles) in ascending order.

2. The second constraint is related to the values of the first and the last genes of the chromosome. The value of the first gene should not be smaller than 1.5 ($x_{i,1} \geq 1.5$), and the value of the last one should not be higher than $N-0.5$ ($x_{i,m-1} < N - 0.5$). This is because the first and the last point of the time series can not be cut points, and the nearest integers of a value lower than 1.5 or a value higher than $N - 0.5$ are 1 and N, respectively. If this constraint is not satisfied, the chromosome is rescaled:

$$\mathbf{x}_i^{t'} = \frac{\mathbf{x}_i^t - \min\{\mathbf{x}_i^t\}}{\max\{\mathbf{x}_i^t\} - \min\{\mathbf{x}_i^t\}}(\max\{\mathbf{x}_i^{t-1}\} - \min\{\mathbf{x}_i^{t-1}\}) + \min\{\mathbf{x}_i^{t-1}\}, \tag{11}$$

where the $\min(\mathbf{x})$ and $\max(\mathbf{x})$ represent the minimum and the maximum value of the array $\mathbf{x}$, respectively.

4.6. Hybridisation procedure

A local search strategy is used to further improve the quality of the solutions, based on the combination of Bottom-Up and Top-Down algorithms [27]. Bottom-Up considers each element of the time series as a possible cut point, and, during the iterations, the two adjacent segments incurring in a lowest cost are merged, that is, those adjacent segments whose merging results in the minimum increase of error. Top-Down is the complementary algorithm, which works with the opposite philosophy. At the beginning, the complete time series is considered as a segment, and Top-Down recursively splits the segment considering the point resulting in the maximum error decrease. Both algorithms are run until some stopping criteria are met (related with the approximation error). Our proposed local search methods consists in removing a percentage of the cut points of the best solution using the Bottom-Up strategy and then adding the same number of cut points using the Top-Down algorithm.

To use these algorithms, we have modified the implementations proposed in [27] in such a way that, for both, the stopping criteria is the number of segments to merge or cut, respectively. Note that the implementation of Top-Down presented in [27] is recursive, so we have transformed it into an iterative method.

We have considered the following strategy for combining this local search with the metaheuristics (GA, PSO 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 a local search.

4.7. DBBePSO algorithm for time series segmentation

This section summarises the work-flow of the DBBePSO presented in Section 3 for time series segmentation, including all the considerations previously exposed. The main steps of the algorithm are summarised in Algorithm 2. Very similar pseudocodes are used for adapting the rest of PSO variants.

Algorithm 2 Dynamic BBePSO for time series segmentation

Input: Time series.
Output: Segmented time series.

```
Initialise a random initial particle swarm (population).
Evaluate the initial population.
while not stop condition do
    Update the importance of the social and cognitive components.
    Update the positions of the particles.
    Repair solutions.
    Evaluate the new population (particle swarm).
    Update the best global and the best local positions.
end while
Apply the local search to the best solution obtained by the DBBePSO.
return Best solution after the local search.
```

5. Experimental results and discussion

This section analyses the time series considered for validating the different methods, the experimental setting and the results obtained.

5.1. Datasets used in our experiments

In this work, we evaluate the performance of the DBBePSO algorithm 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:

- Synthetic time series
  - UCR time series: four datasets from the UCR Time Series Classification Archive [42] has been selected. Originally, these time series are divided into training and test, because it is a time series classification repository. As we are facing time series segmentation, we have joined some of the training patterns in order to have larger length time series. The time series selected are Hand Outlines, with a total of 8127 points, and Mallat, Phoneme and StarLightCurves, all of them with 8192 observations.
  - Donoho-Johnstone time series: this series is extracted from a benchmark repository [43, 44, 45], which is widely used in the neural net and machine learning community. 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[1].
- Real-world application time series

[1] All these time series can be downloaded from `https://sites.google.com/site/icdmmdl/`

- Stock prices time series from financial applications: five different indexes has been selected. The first one is IBEX35 time series (called IBEX since this moment for simplicity). It is one of the Spanish official indexes of the Madrid stock market. The time series consists of a total of 5730 observations considering daily values from 14 January 1992 to 26 September 2014. The rest of time series includes market rates collected from four banks (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 [46] have been used. 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 [47, 48]. We used the MLII signal of the record 108 (9000 observations) to test the algorithm in this dataset.

All time series considered are shown in Figures 3 and 4.

5.2. Experimental design

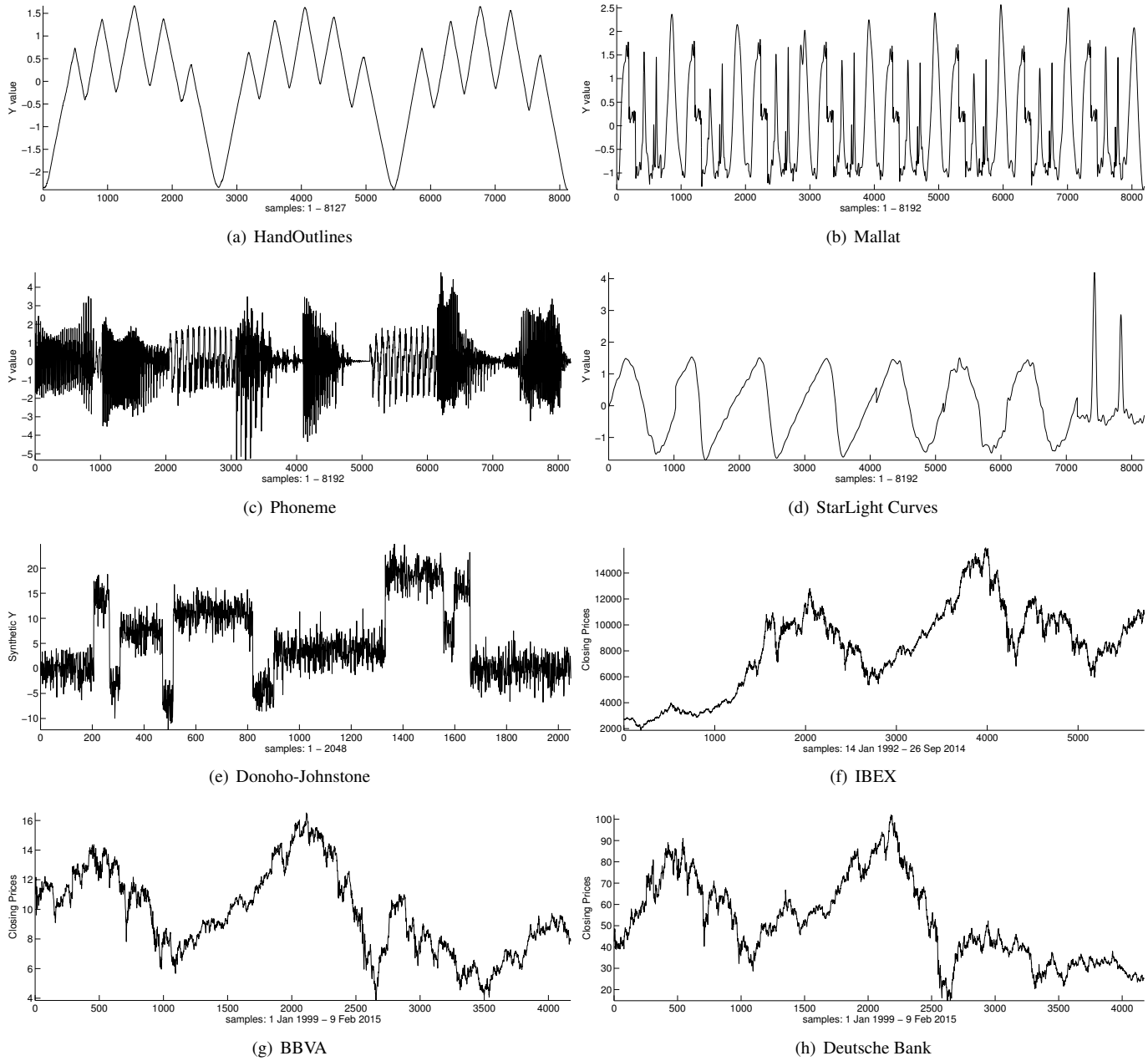
The experimental design for the time series under study is presented in this subsection. We compare the following algorithms:

- An optimal algorithm which is able to obtain the minimum error segmentation for a given time series. We consider the algorithm proposed by Salotti [49], which is based on finding the shortest path in a graph, whose complexity is close to $O(N^2)$. We consider the improved version proposed in [50], where the computational time is reduced about a 16%. This algorithm was originally proposed for obtaining optimal polygonal approximations in closed curves, which is exactly the same problem than time series segmentation, with two main differences: the first and the last points are fixed, and the error is calculated in the vertical line, instead of in a line perpendicular to the approximation. Both adaptations can be easily included to perform optimal time series segmentation.
- Two iterative versions of the Bottom-Up and Top-Down algorithms explained in Section 4.6 have been run, with the aim of obtaining an approximation of the time series with a predefined number of points or segments.
- A genetic algorithm (GA) has been run with crossover and mutation probabilities set to $p_c = 0.8$ and $p_m = 0.2$, respectively.
- A basic particle swarm optimisation algorithm (PSO) is run with the following specific parameters: initial velocities of the particles are set to values close to zero [51], the inertia coefficient (w) is set to 0.72, and the constant parameters (C_1 and C_2) are fixed to 1.49, as previously proposed in [39].
- The exploiter version of the barebones PSO (BBePSO) proposed by [31] has also been tested. Note that this version is better than BBPSO (see [31]), so we have not considered this last algorithm in our experiments.
- Finally, the DBBePSO proposed in this paper is also run, and, as mentioned before, the λ parameter is set to 1 at the beginning, and it linearly decreases to 0.1. No other parameters have to be set.

According to [52], a 40% of the in the GA, BBePSO, and DBBePSO are fine-tuned according to the method presented in Section 4.6 resulting in the HGA, HBBePSO, and HDBBePSO methods, respectively. 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, by considering the equation $3.5N$. Given the stochastic nature of the evolutionary algorithms, they have been run 30 times with different seeds. The error approximation results, measured in RMSE, and the computational time in seconds are analysed. Finally, some statistical tests are performed to determine the existence of significant differences in the results, which will be later detailed.

5.3. Discussion

RMSE results are shown in Table 1. For the deterministic algorithms (Salotti, Bottom-Up and Top-Down), there is a single result for each dataset-algorithm pair. In the case of the evolutionary algorithms (GA, PSO, BBePSO, DBBePSO and their hybrid versions), the table summarises the mean and the standard deviation of the 30 runs using different seeds. The mean ranking of each algorithm is also included, considering a 1 for the best method for each dataset and an 11 for the worst one. Firstly, we can observe that the proposed local search method improves the solutions of the evolutionary algorithms to a large extent. That is, hybrid algorithms reduce the approximation error of their corresponding standard evolutionary ones. In this way, the mean ranking of the GA is improved from 9.03 to 4.03, the ranking of PSO increases from 10.93 to 4.57, BBePSO improves from 9.17 to 3.83, and the DBBePSO ranking is 7.53, this ranking being 2.43 in the corresponding hybrid version (HDBBePSO). Obviously, the best method in error terms for all databases is the optimal algorithm of Salotti. In general, if we do not consider the optimal algorithm, the best results are



(a) HandOutlines (b) Mallat

(c) Phoneme (d) StarLight Curves

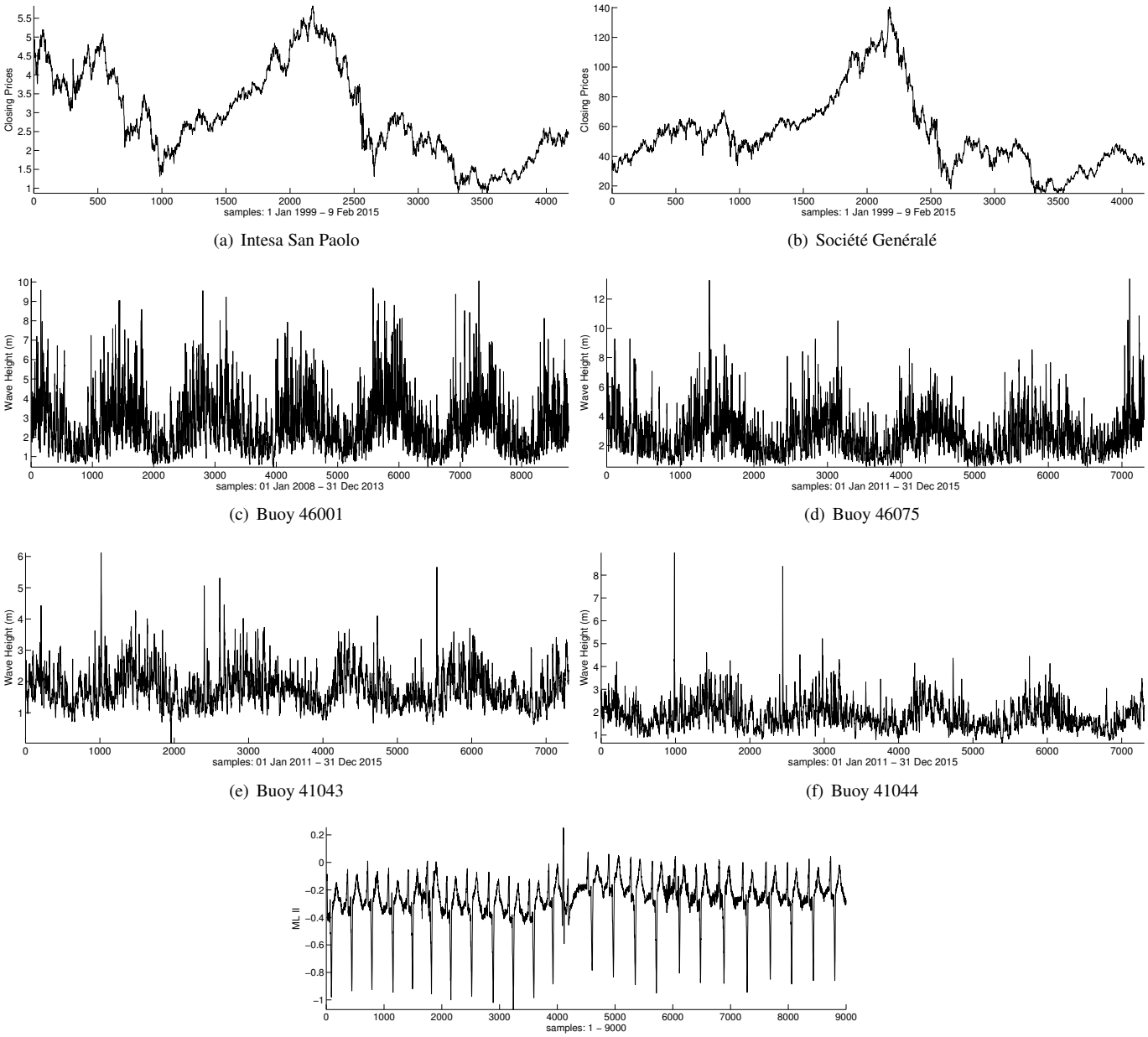
(e) Donoho-Johnstone (f) IBEX

(g) BBVA (h) Deutsche Bank

Figure 3: Time series considered for the experiments (1/2).

obtained with the HDBBePSO algorithm, with the lowest error for 10 out of 15 time series, and the second-best RMSE in the rest of series. HGA, HPSO and HBBePSO results seem to be very similar in performance with a mean rank of 4.03, 4.57, and 3.83, respectively. Furthermore, the standard deviations of HDBBePSO are the lowest ones, showing the robustness of the proposed method (the performance does not depend so much on the initialisation).

If we only observe standard evolutionary algorithms without considering hybrid versions, we can also conclude that the novel DBBePSO outperforms the rest of methods. Bottom-Up appear to be better finding low approximation error solutions when compared with all evolutionary algorithms except DBBePSO (again without considering hybrid versions). This is due to the bad performance of evolutionary methods in finding the precise optimum in high-quality areas, this reason motivating the use of hybrid algorithms. However, as can be seen, the dynamic adaptation of the exploration and exploitation of DBBePSO reduces this problem, obtaining the same error approximation that Top-Down and a slightly worse error, but comparable, with respect to Bottom-Up. Moreover, this problem is completely solved with the hybridisation proposed in this paper, which results, in the experiments, in the lowest error approximations, improving Top-Down and Bottom-Up methods to a



(a) Intesa San Paolo

(b) Société Genéralé

(c) Buoy 46001

(d) Buoy 46075

(e) Buoy 41043

(f) Buoy 41044

(g) MIT BIH Arrhythmia

Figure 4: Time series considered for the experiments (2/2).

great extent.

It is known that an important inconvenient of evolutionary algorithms (based on populations of solutions) is their higher computational cost when compared to algorithms based on a single solution which are not optimal. Table 2 summarises runtime for the deterministic algorithms and the mean and the standard deviation values of 30 run times for the evolutionary, ones 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). As can be seen, the worst computational times are obtained by the Salotti's method showing the necessity of using non optimal algorithms in order to derive good solutions in acceptable computational time. Specifically, when compared to HDBBePSO, the computational cost of Salotti's method is approximately nine times higher in the worst case and twice in the best one. The rest of the results confirm that the fastest algorithms are Bottom-Up and Top-Down. However, we should take into account that the approximation error obtained by these methods is clearly worse than that obtained by hybrid methodologies (specially, by HDBBePSO, see Table 1). Obviously, the hybrid versions of the algorithms are slightly costlier than their pure evolutionary alternatives. PSO is faster than the rest of the evolutionary methods but the second fastest method is DBBePSO, while being much better obtaining lower RMSE.

In order to analyse the results from the point of view of statistical hypothesis contrasts, a set of statistical tests have been used. Given that Salotti's algorithm is the optimal method, it does not make sense to include it the statistical test (it will be always the best performing method, at the cost of much higher computational resources). Also, the hybrid methods are always better and slightly costlier than the pure evolutionary ones. For these reasons, for the statistical tests, we only consider the deterministic methods and the hybrid versions of the algorithms, i.e. Top-Down, Bottom-Up, HGA, HPSO, HBBePSO and HDBBePSO. Firstly, we analyse the RMSE results. To do so, a Friedman test [53] has been considered using the different RMSE rankings, which shows that, for a level of significance $\alpha = 5\%$, the confidence interval is $C_0 = (0, F_{0.05} = 2.35)$, and the F-distribution statistical value is $F^* = 22.19$. Consequently, the test rejects the null-hypothesis, which states that all algorithms perform equally in mean ranking of RMSE, that is, the algorithm effect is statistically significant. Due to this rejection, we consider the best performing method in RMSE as control method for a post-hoc test [54], comparing this algorithm with the rest of methods. It has been noted that comparing all algorithms to a given one (control method) is more sensitive than comparing all algorithms to each other.

Algorithm	Salotti	Bottom-Up	Top-Down	GA	HGA	PSO	HPSO	BBePSO	HBBePSO	DBBePSO	HDBBePSO
Hand Outlines	**0.004**	0.005	0.006	0.023 ± 0.003	0.005 ± 0.000	0.036 ± 0.014	0.005 ± 0.000	0.010 ± 0.001	0.005 ± 0.000	0.007 ± 0.000	**0.004 ± 0.000**
Mallat	**0.072**	*0.097*	0.502	0.305 ± 0.016	0.111 ± 0.004	0.345 ± 0.056	0.110 ± 0.004	0.246 ± 0.016	0.106 ± 0.003	0.203 ± 0.010	0.104 ± 0.003
Phoneme	**0.746**	1.057	0.940	0.957 ± 0.011	*0.857 ± 0.005*	1.132 ± 0.024	0.859 ± 0.005	1.042 ± 0.027	0.859 ± 0.005	1.019 ± 0.025	0.858 ± 0.005
StarLightCurves	**0.011**	*0.016*	0.026	0.054 ± 0.004	0.017 ± 0.000	0.081 ± 0.029	0.017 ± 0.000	0.037 ± 0.002	0.017 ± 0.000	0.030 ± 0.002	0.017 ± 0.000
Donoho-Johnstone	**2.218**	2.639	3.466	2.961 ± 0.070	*2.414 ± 0.033*	3.545 ± 0.236	2.431 ± 0.035	3.030 ± 0.078	2.431 ± 0.035	2.896 ± 0.081	2.425 ± 0.035
IBEX	**149.962**	210.321	269.801	261.067 ± 8.732	180.941 ± 1.836	321.976 ± 29.118	181.408 ± 1.772	264.590 ± 11.581	180.431 ± 1.645	229.682 ± 9.437	*179.365 ± 1.928*
BBVA	**0.236**	0.320	0.405	0.431 ± 0.009	0.286 ± 0.003	0.464 ± 0.039	0.286 ± 0.003	0.404 ± 0.012	0.285 ± 0.003	0.356 ± 0.010	*0.282 ± 0.003*
DEUTSCHE	**1.421**	2.032	2.318	2.428 ± 0.093	1.673 ± 0.019	2.899 ± 0.227	1.677 ± 0.021	2.428 ± 0.112	1.675 ± 0.020	2.105 ± 0.087	*1.658 ± 0.018*
SAN PAOLO	**0.080**	0.112	0.136	0.154 ± 0.003	0.097 ± 0.001	0.163 ± 0.018	0.097 ± 0.001	0.138 ± 0.005	0.097 ± 0.001	0.120 ± 0.005	*0.096 ± 0.001*
SO Generálé	**1.598**	2.292	2.472	2.663 ± 0.085	1.902 ± 0.020	3.168 ± 0.296	1.905 ± 0.021	2.708 ± 0.087	1.898 ± 0.022	2.378 ± 0.087	*1.882 ± 0.023*
B46001	**0.799**	1.088	1.011	1.137 ± 0.010	0.931 ± 0.005	1.261 ± 0.019	0.931 ± 0.005	1.166 ± 0.012	0.931 ± 0.005	1.117 ± 0.026	*0.927 ± 0.005*
B46075	**0.822**	1.145	1.056	1.182 ± 0.011	0.978 ± 0.007	1.334 ± 0.019	0.978 ± 0.007	1.224 ± 0.023	0.978 ± 0.007	1.191 ± 0.025	*0.975 ± 0.007*
B41043	**0.295**	0.426	0.449	0.478 ± 0.006	0.360 ± 0.004	0.528 ± 0.009	0.360 ± 0.004	0.483 ± 0.010	0.359 ± 0.004	0.451 ± 0.013	*0.356 ± 0.004*
B41044	**0.292**	0.419	0.425	0.476 ± 0.008	0.351 ± 0.003	0.531 ± 0.011	0.351 ± 0.003	0.485 ± 0.009	0.351 ± 0.003	0.453 ± 0.014	*0.348 ± 0.003*
Arrhytmia	**0.022**	*0.032*	0.091	0.100 ± 0.004	0.038 ± 0.001	0.114 ± 0.008	0.038 ± 0.001	0.078 ± 0.004	0.037 ± 0.001	0.064 ± 0.003	0.037 ± 0.001
Mean rankings ($\bar{r}$)	**1.03**	5.50	7.93	9.03	4.03	10.93	4.57	9.17	3.83	7.53	*2.43*

Table 1: RMSE values obtained by all the algorithms in each time series. Salotti, Top-Down, and Bottom-Up are deterministic and they are run once, while GA, HGA, PSO, HPSO, BBePSO, HBBePSO, DBBePSO, and 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.

Algorithm	Salotti	Bottom-Up	Top-Down	GA	HGA	PSO	HPSO	BBePSO	HBBePSO	DBBePSO	HDBBePSO
Hand Outlines	594.84	*9.97*	**9.61**	41.94 ± 2.25	58.06 ± 1.73	13.64 ± 0.78	24.26 ± 0.94	39.33 ± 3.89	51.32 ± 2.35	41.41 ± 1.347	52.39 ± 3.94
Mallat	556.55	*22.40*	**22.01**	55.53 ± 0.57	71.03 ± 0.94	25.26 ± 1.26	36.53 ± 1.29	53.77 ± 5.62	63.96 ± 2.80	56.01 ± 3.950	63.83 ± 0.45
Phoneme	720.58	*21.48*	**21.14**	56.97 ± 0.70	70.96 ± 0.90	25.16 ± 0.76	35.65 ± 0.67	55.61 ± 3.18	63.65 ± 3.02	55.20 ± 0.630	62.95 ± 0.60
StarLightCurves	680.64	**20.77**	*21.86*	55.76 ± 1.45	69.85 ± 2.25	26.65 ± 1.17	35.42 ± 0.81	56.20 ± 7.66	62.87 ± 0.62	54.35 ± 2.175	65.02 ± 3.16
Donoho-Johnstone	9.71	**1.08**	*1.16*	4.13 ± 0.21	6.22 ± 0.18	1.47 ± 0.16	3.45 ± 0.14	2.87 ± 0.10	4.96 ± 0.37	3.051 ± 0.271	4.95 ± 0.15
IBEX	205.40	**4.37**	*4.55*	22.17 ± 0.39	30.64 ± 0.90	6.87 ± 0.17	13.60 ± 0.50	19.02 ± 0.08	27.35 ± 2.29	20.88 ± 1.427	27.80 ± 2.41
BBVA	75.24	**2.32**	*2.44*	12.21 ± 0.21	17.94 ± 0.73	3.55 ± 0.04	8.18 ± 0.12	11.14 ± 1.42	15.52 ± 1.20	10.68 ± 0.295	15.32 ± 0.84
DEUTSCHE	72.39	**2.31**	*2.77*	12.11 ± 0.24	17.19 ± 0.19	3.59 ± 0.08	8.11 ± 0.08	11.06 ± 1.39	15.21 ± 1.28	10.91 ± 0.802	14.78 ± 0.28
SAN PAOLO	69.69	**2.31**	*2.59*	12.18 ± 0.30	17.10 ± 0.10	3.61 ± 0.11	8.19 ± 0.08	10.16 ± 0.74	14.44 ± 0.09	11.05 ± 0.785	14.63 ± 0.09
SO Genéralé	73.62	**2.31**	*2.65*	12.12 ± 0.20	17.12 ± 0.10	3.60 ± 0.09	8.14 ± 0.09	9.93 ± 0.64	14.43 ± 0.07	11.01 ± 0.781	14.63 ± 0.11
B46001	807.24	**10.64**	*12.01*	48.99 ± 1.10	63.07 ± 0.80	16.95 ± 0.49	27.66 ± 0.53	51.52 ± 4.50	55.88 ± 0.64	48.59 ± 2.472	57.65 ± 3.22
B46075	520.19	**7.18**	*7.78*	33.75 ± 0.25	45.05 ± 0.38	11.37 ± 0.34	19.52 ± 0.44	30.70 ± 3.00	40.66 ± 2.05	33.76 ± 2.959	40.17 ± 0.21
B41043	423.73	**7.19**	*8.11*	33.88 ± 0.19	44.96 ± 0.36	11.28 ± 0.30	19.32 ± 0.37	37.12 ± 0.07	41.48 ± 3.66	33.48 ± 2.663	39.96 ± 0.20
B41044	478.03	**7.40**	*8.58*	33.86 ± 0.60	42.15 ± 1.52	11.22 ± 0.18	19.43 ± 0.68	33.77 ± 3.96	43.31 ± 4.60	36.21 ± 4.428	40.75 ± 2.56
Arrhytmia	714.73	**11.32**	*11.98*	51.41 ± 1.03	60.38 ± 0.48	18.19 ± 0.47	28.76 ± 1.27	53.29 ± 5.29	61.69 ± 8.03	49.98 ± 3.301	59.70 ± 4.93
Mean rankings ($\bar{r}$)	11.00	**1.20**	*1.80*	6.53	9.87	3.00	4.13	5.67	8.73	5.67	8.40

Table 2: Computational time in seconds obtained by all the algorithms in each time series. Salotti, Top-Down, and Bottom-Up are deterministic and they are run once, while GA, HGA, PSO, HPSO, BBePSO, HBBePSO, DBBePSO, and 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.

	CA:HDBBePSO	RMSE	
i	$\alpha^*_{0.05}$	Algorithm	p_i
1	0.010	Top-Down	0.000 (*)
2	0.013	Bottom-Up	0.000 (*)
3	0.017	HPSO	0.002 (*)
4	0.025	HGA	0.021 (*)
5	0.050	HBBePSO	0.045 (*)

Table 3: Results of the Holm test using HDBBePSO as control algorithm (CA) when comparing its average RMSE to those of Top-Down, Bottom-Up, HGA, HPSO, and HBBePSO: corrected α values, compared methods and p-values, all of them ordered by the number of comparison (i). CA results statistically better than the compared algorithm are marked with (*).

	CA:HDBBePSO	Run time (s)	
i	$\alpha^*_{0.05}$	Algorithm	p_i
1	0.010	Bottom-Up	0.000 (-)
2	0.013	Top-Down	0.000 (-)
3	0.017	HGA	0.032
4	0.025	HPSO	0.040
5	0.050	HBBePSO	0.626

Table 4: Results of the Holm test using HDBBePSO as control algorithm (CA) when comparing its average runtime to those of Top-Down, Bottom-Up, HGA, HPSO, and HBBePSO: corrected α values, compared methods and p-values, all of them ordered by the number of comparison (i). CA results statistically worse than the compared algorithm are marked with (−).

The Holm's test compares the i-th and j-th algorithms with the following statistic:

$$z = \frac{\bar{r}_i - \bar{r}_j}{\sqrt{\frac{k(k+1)}{6N}}},$$

where $\bar{r}_i$ is the mean ranking of the i-algorithm, k is the number of algorithms, and N is the number of datasets. With the value of z, we find the probability of a normal distribution and compared it with a level of significance α. Holm's test adjusts the value for α to compensate multiple comparisons, using a procedure that sequentially tests the hypotheses ordered by their significance. The ordered p-values are denoted by $p_1, p_2, \ldots, p_k$, so that $p_1 < p_2 < ... < p_k$. The test compares each p_i with $\alpha^*_i = \alpha/(k-i)$, starting with the most significant p-value. If p_1 is lower than $\alpha/(k-1)$, the corresponding hypothesis is rejected, and then we compare p_2 with $\alpha/(k-2)$, and so on. When a certain null hypothesis is accepted the remaining ones are also accepted.

The results of the Holm's test are shown in Table 3. When using HDBBePSO as control algorithm (CA), Holm's test shows that $p_i < \alpha^*_i$ in all cases, for $\alpha = 0.05$, confirming that there are statistically significant differences favouring HDBBePSO.

In the same way, to determine the existence of statistical significance of the rank differences in runtime (seconds) for the six algorithms and all databases, we perform another Friedman test with their mean runtime rankings. We observe that, for a level of significance of 5%, the F-distribution statistical value is $F^* = 201.33$ with a confidence interval of $C_0 = (0, F_{0.05} = 2.35)$, rejecting the null-hypothesis and concluding that the differences are statistically significant. Then, we apply the Holm's test, considering HDBBePSO, again, as the control algorithm. The results are shown in Table 4. Using HDBBePSO as CA, Bottom-Up and Top-Down are significantly better in mean run time than the proposed algorithm (marked with "(-)" in Table 4). This is because the optimisation of Bottom-Up and Top-Down is based on a single solution and the methods are not optimal, while the evolutionary approaches are based on populations. Finally, with respect the remaining methods (HGA, HPSO, and HBBePSO), there are no statistically significant differences in runtime, but HDBBePSO outperforms them in quality of solutions.

6. Conclusions

This paper proposes a novel algorithm for time series segmentation based on reducing the number of points of the time series by minimising the approximation error of the linear interpolation of each segment. The contributions include the adaptation of the particle swarm optimisation algorithm (PSO) and its exploiter barebones variant (BBePSO) for time series segmentation, along with the improvement of them using a dynamic adaptation of the exploration and exploitation importances (dynamic BBePSO, DBBePSO). All algorithms are hybridised with a local search which combines the Bottom-Up and Top-Down strategies. The proposed method is then compared with other state-of-the-art algorithms: a genetic algorithm (GA), a standard particle swarm optimisation (PSO), the exploiting barebones PSO (BBePSO), all their hybrid versions, the traditional Top-Down and Bottom-Up procedures, and Salotti's optimal algorithm.

The results conclude that the hybrid versions (HGA, HPSO, HBBePSO, HDBBePSO) improve the solutions obtained by their standard versions (GA, PSO, BBePSO, DBBePSO), showing that the hybridisation proposed is suitable for this type of problems. Salotti's algorithm is the best method in terms of RMSE, but the computational cost is much higher than that of the rest of algorithm. Furthermore, without considering Salotti's method, HDBBePSO results in the best results, obtaining the lowest approximation error, where the differences are found to be statistically significant. These results conclude that the dynamic adaptation of the BBePSO allows the algorithm to escape the initial local optima and converge to optimal solutions at the end of the evolution. The algorithm proposed is statistically lower than traditional approaches (Top-Down and Bottom-UP), but their solutions are much worse.

For a future line of work, other distributions instead the Gaussian distribution could be taken into account, e.g. the Weibull distribution. We also plan to extend this work using the original and the approximated time series in posterior tasks, such as clustering or classification, observing if the method reduces the noise of the time series. Moreover, linear regression could be also considered instead of linear interpolation, or even using polynomials with degree greater than one.

Acknowledgement

This work has been subsidized by the projects TIN2017-85887-C2-1-P, TIN2014-54583-C2-1-R and TIN2015-70308-REDT of the Spanish Ministry of Economy and Competitiveness (MINECO), and FEDER funds (FEDER EU). Antonio M. Durán-Rosal's research has been subsidized by the FPU Predoctoral Program of the Spanish Ministry of Education, Culture and Sport (MECD), grant reference FPU14/03039.

References

[1] P. Esling, C. Agon, Time-series data mining, ACM Computing Surveys (CSUR) 45 (1) (2012) 12.

[2] C. H. Fontes, H. Budman, A hybrid clustering approach for multivariate time series–a case study applied to failure analysis in a gas turbine, ISA transactions.

[3] M. Pérez-Ortiz, A. Durán-Rosal, P. Gutiérrez, J. Sánchez-Monedero, A. Nikolaou, F. Fernández-Navarro, C. Hervás-Martínez, On the use of evolutionary time series analysis for segmenting paleoclimate data, Neurocomputing.doi:https://doi.org/10.1016/j.neucom.2016.11.101.

[4] W. Deng, G. Wang, A novel water quality data analysis framework based on time-series data mining, Journal of Environmental Management 196 (2017) 365–375.

[5] X. Gong, Y.-W. Si, S. Fong, R. P. Biuk-Aghai, Financial time series pattern matching with extended ucr suite and support vector machine, Expert Systems with Applications 55 (2016) 284–296.

[6] T. Guyet, H. Nicolas, Long term analysis of time series of satellite images, Pattern Recognition Letters 70 (2016) 17–23.

[7] A. Bagnall, J. Lines, J. Hills, A. Bostrom, Time-series classification with COTE: the collective of transformation-based ensembles, IEEE Transactions on Knowledge and Data Engineering 27 (9) (2015) 2522–2535.

[8] J. Zhao, L. Itti, Classifying time series using local descriptors with hybrid sampling, IEEE Transactions on Knowledge and Data Engineering 28 (3) (2016) 623–637.

[9] M.-Y. Chen, B.-T. Chen, A hybrid fuzzy time series model based on granular computing for stock price forecasting, Information Sciences 294 (2015) 227–241.

[10] B. Sun, H. Guo, H. R. Karimi, Y. Ge, S. Xiong, Prediction of stock index futures prices based on fuzzy sets and multivariate fuzzy time series, Neurocomputing 151 (2015) 1528–1536.

[11] A. Nikolaou, P. A. Gutiérrez, A. Durán, I. Dicaire, F. Fernández-Navarro, C. Hervás-Martínez, Detection of early warning signals in paleoclimate data using a genetic time series segmentation algorithm, Climate Dynamics 44 (7-8) (2015) 1919–1933.

[12] L. N. Ferreira, L. Zhao, Time series clustering via community detection in networks, Information Sciences 326 (2016) 227–242.

[13] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, E. Keogh, Experimental comparison of representation methods and distance measures for time series data, Data Mining and Knowledge Discovery (2013) 1–35.

[14] H. Kaya, Ş. Gündüz-Öğüdücü, A distance based time series classification framework, Information Systems 51 (2015) 27–42.

[15] A. M. Durán-Rosal, P. A. Gutiérrez, S. Salcedo-Sanz, C. Hervás-Martínez, A statistically-driven coral reef optimization algorithm for optimal size reduction of time series, Applied Soft Computing 63 (2018) 139–153.

[16] F.-L. Chung, T.-C. Fu, V. Ng, R. W. Luk, An evolutionary approach to pattern-based time series segmentation, Evolutionary Computation, IEEE Transactions on 8 (5) (2004) 471–489.

[17] V. S. Tseng, C.-H. Chen, P.-C. Huang, T.-P. Hong, Cluster-based genetic segmentation of time series with dwt, Pattern Recognition Letters 30 (13) (2009) 1190–1197.

[18] J. Abonyi, B. Feil, S. Nemeth, P. Arva, Modified gath–geva clustering for fuzzy segmentation of multivariate time-series, Fuzzy Sets and Systems 149 (1) (2005) 39–56.

[19] Y. Gorshkov, I. Kokshenev, Y. Bodyanskiy, V. Kolodyazhniy, O. Shylo, Robust recursive fuzzy clustering-based segmentation of biological time series, in: Evolving Fuzzy Systems, 2006 International Symposium on, IEEE, 2006, pp. 101–105.

[20] E. Fuchs, T. Gruber, J. Nitschke, B. Sick, On-line motif detection in time series with swiftmotif, Pattern Recognition 42 (11) (2009) 3015 – 3031.

[21] A. M. Durán-Rosal, M. de la Paz-Marín, P. A. Gutiérrez, C. Hervás-Martínez, Identifying market behaviours using european stock index time series by a hybrid segmentation algorithm, Neural Processing Letters (2017) 1–24.

[22] A. Durán-Rosal, J. Fernández, P. Gutiérrez, C. Hervás-Martínez, Detection and prediction of segments containing extreme significant wave heights, Ocean Engineering 142 (2017) 268–279.

[23] S. Aghabozorgi, A. S. Shirkhorshidi, T. Y. Wah, Time-series clustering–a decade review, Information Systems 53 (2015) 16–38.

[24] S. Rani, G. Sikka, Recent techniques of clustering of time series data: a survey, International Journal of Computer Applications 52 (15).

[25] J. Oliver, C. Forbes, Bayesian approaches to segmenting a simple time series, Tech. Rep. 14/97, Monash University, Department of Econometrics and Business Statistics (1997).
URL `http://EconPapers.repec.org/RePEc:msh:ebswps:1997-14`

[26] J. J. Oliver, R. A. Baxter, C. S. Wallace, Minimum message length segmentation, in: X. Wu, R. Kotagiri, K. Korb (Eds.), Research and Development in Knowledge Discovery and Data Mining, Vol. 1394 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1998, pp. 222–233.

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

[28] K. Chakrabarti, E. Keogh, S. Mehrotra, M. Pazzani, Locally adaptive dimensionality reduction for indexing large time series databases, ACM Transactions on Database Systems (TODS) 27 (2) (2002) 188–228.

[29] J. Lin, E. Keogh, S. Lonardi, B. Chiu, A symbolic representation of time series, with implications for streaming algorithms, in: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, ACM, 2003, pp. 2–11.

[30] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Neural Networks, 1995. Proceedings., IEEE International Conference on, Vol. 4, 1995, pp. 1942–1948.

[31] J. Kennedy, Bare bones particle swarms, in: Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE, 2003, pp. 80–87.

[32] K. Chau, Particle swarm optimization training algorithm for anns in stage prediction of shing mun river, Journal of Hydrology 329 (3) (2006) 363 – 367.

[33] M. Zhang, M. Xin, J. Yang, Adaptive multi-cue based particle swarm optimization guided particle filter tracking in infrared videos, Neurocomputing 122 (Supplement C) (2013) 163 – 171, advances in cognitive and ubiquitous computing.

[34] D. Tungadio, B. Numbi, M. Siti, A. Jimoh, Particle swarm optimization for power system state estimation, Neurocomputing 148 (Supplement C) (2015) 175 – 180.

[35] C. R. Houck, J. A. Joines, M. G. Kay, J. R. Wilson, Empirical investigation of the benefits of partial lamarckianism, Evol. Comput. 5 (1) (1997) 31–60.

[36] N. L. J. Ulder, E. H. L. Aarts, H.-J. Bandelt, P. J. M. v. Laarhoven, E. Pesch, Genetic local search algorithms for the travelling salesman problem, in: Proceedings of the 1st Workshop on Parallel Problem Solving from Nature, PPSN I, Springer-Verlag, London, UK, UK, 1991, pp. 109–116.

[37] A. Kolen, E. Pesch, Genetic local search in combinatorial optimization, Discrete Applied Mathematics 48 (3) (1994) 273 – 284.

[38] J. A. Joines, M. G. Kay, Utilizing hybrid genetic algorithms, in: Evolutionary Optimization, Vol. 48 of International Series in Operations Research & Management Science, Springer US, 2002, pp. 199–228.

[39] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE transactions on Evolutionary Computation 6 (1) (2002) 58–73.

[40] M. G. Omran, A. Engelbrecht, A. Salman, Barebones particle swarm for integer programming problems, in: Swarm Intelligence Symposium, 2007. SIS 2007. IEEE, IEEE, 2007, pp. 170–175.

[41] V. K. Koumousis, C. P. Katsaras, A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance, IEEE Transactions on Evolutionary Computation 10 (1) (2006) 19–28.

[42] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, The ucr time series classification archive, `www.cs.ucr.edu/~eamonn/time_series_data/` (July 2015).

[43] D. L. Donoho, J. M. Johnstone, Ideal spatial adaptation by wavelet shrinkage, Biometrika 81 (3) (1994) 425–455.

[44] D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, D. Picard, Wavelet shrinkage: Asymptopia?, Journal of the Royal Statistical Society. Series B (Methodological) 57 (2) (1995) 301–369.

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

[46] http://www.ndbc.noaa.gov/, National buoy data center, National Oceanic and Atmospheric Administration of the USA (NOAA), 2015.

[47] G. Moody, R. Mark, The impact of the MIT-BIH arrhythmia database, Engineering in Medicine and Biology Magazine, IEEE 20 (3) (2001) 45–50.

[48] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, H. E. Stanley, Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals, Circulation 101 (23) (2000) e215–e220.

[49] M. Salotti, An efficient algorithm for the optimal polygonal approximation of digitized curves, Pattern Recognition Letters 22 (2) (2001) 215–221.

[50] A. Carmona-Poyato, E. Aguilera-Aguilera, F. Madrid-Cuevas, M. Marín-Jiménez, N. Fernández-García, New method for obtaining optimal polygonal approximations to solve the min-\varepsilon ε problem, Neural Computing and Applications 28 (9) (2017) 2383–2394.

[51] A. Engelbrecht, Particle swarm optimization: Velocity initialization, in: Evolutionary Computation (CEC), 2012 IEEE Congress on, IEEE, 2012, pp. 1–8.

[52] A. M. Durán-Rosal, P. A. Gutiérrez-Peña, F. J. Martínez-Estudillo, C. Hervás-Martínez, Time Series Representation by a Novel Hybrid Segmentation Algorithm, Springer International Publishing, Cham, 2016, pp. 163–173.

[53] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, The Annals of Mathematical Statistics 11 (1) (1940) 86–92.

[54] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research 7 (Jan) (2006) 1–30.