Information Sciences 511 (2020) 74–93

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

An evolutionary approach for efficient prototyping of large time series datasets

Pablo Leon-Alcaide [a], Luis Rodriguez-Benitez [a,*], Ester Castillo-Herrera [a], Juan Moreno-Garcia [b], Luis Jimenez-Linares [a]

[a] Department of Information and System Technologies, University of Castilla-La Mancha, Escuela Superior de Informatica, Paseo de la Universidad s/n, Ciudad Real, Spain
[b] Department of Information and System Technologies, University of Castilla-La Mancha, Escuela de Ingenieria Industrial, Avenida Carlos III s/n, Toledo, Spain

ARTICLE INFO

Article history:
Received 18 May 2018
Revised 8 September 2019
Accepted 21 September 2019
Available online 23 September 2019

Keywords:
Time series summarization
Genetic algorithms
Elastic distances
Data mining

ABSTRACT

We here describe an algorithm based on an evolutionary strategy to find the prototype series of a set of time series, and we use Dynamic Time Warping (DTW) as a distance measure between series, and do not restrict the search space to the series in the set. The problem of calculating the centroid of a set of time series can be addressed as a minimization problem, using genetic algorithms. Our proposal may be considered among the set of non-classical approaches to genetic algorithms, where an individual gene is a candidate time series for being the centroid or representative of the whole set of series. The representation and operators of genetic algorithms are redesigned, in order to generate efficient summaries, the fitness function of each candidate series to be a prototype is approximated, comparing them only with a subset of randomly selected time series from the original dataset. Three areas are looked at in order to assess the goodness of our proposal: the performance of the prototype generated in terms of a fitness function, the consistency of the prototype generation for use in classical grouping algorithms, and its use in classification algorithms based on the nearest prototypes.

1. Introduction

At present, due to the great increase in the volume of data available, in turn largely due to the increase in storage capacity and the growth of connected sensors, techniques for handling the data are increasingly necessary in order to obtain acceptable results within a reasonable time. One of the most important data types, because of its abundance and diversity, is the time series. For example, seismic series, temperature evolution, sound representation, stock exchange movements, and electrocardiograms. Due to the temporal nature of this type of data, time series have specific characteristics that require special treatment in order for interesting patterns to be discovered. For example, similarity searches among time series can be performed by applications such as clustering, classification, summarization, etc. Time series summarization tries to summarize a set of series by the use of a single representative, a common task in several grouping algorithms. The summary thus obtained can provide high level information that facilitates the identification of frequently appearing patterns among

* Corresponding author.
E-mail addresses: pablo.leon1@alu.uclm.es (P. Leon-Alcaide), luis.rodriguez@uclm.es (L. Rodriguez-Benitez), ester.castillo@uclm.es (E. Castillo-Herrera), juan.moreno@uclm.es (J. Moreno-Garcia), luis.jimenez@uclm.es (L. Jimenez-Linares).

https://doi.org/10.1016/j.ins.2019.09.044
0020-0255/

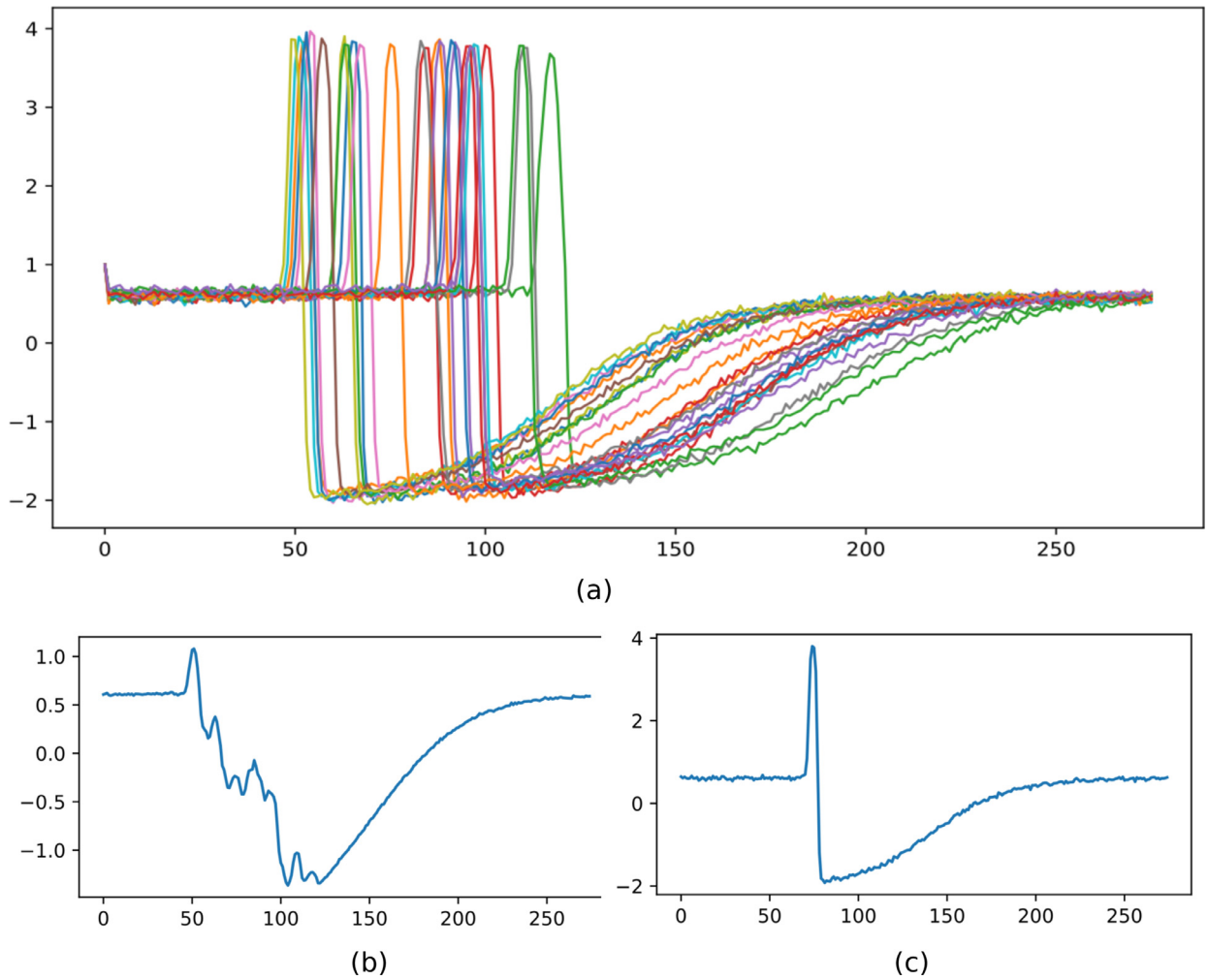


Fig. 1. (a) Set of series. (b) Centroid with Euclidean distance. (c) Centroid with DTW distance.

other applications. Summarizing a set of time series by one representative is a conceptually simple task when using a rigid distance, such as the Manhattan, Euclidean, Minkowski, or Chebyshev distance. When using an elastic distance, such as DTW, however, the problem of calculating the representative becomes much harder, in terms of computational complexity, although such use may be a determining factor in improving the results of supervised and unsupervised learning tasks. Basically, such distances take a natural approach to dealing with the nature of this type of data, leading to the development of more accurate and better learning models. For example, Fig. 1 shows how, on many occasions, the use of Euclidean distance is not suitable for extracting the representative/prototype from a collection of series. Fig. 1.(a) shows a collection of series that are summarized in Fig. 1.(b) using Euclidean distance and in Fig. 1.(c) using a DTW distance. The error in both amplitude and shape can be seen in Fig. 1.(b), while Fig. 1.(c) shows a correct and more natural summarization.

1.1. An elastic distance: dynamic time warping

Dynamic Time Warping (DTW) is a measure of the distance between two time series, belonging to the set of elastic distances [33]. These distances, as opposed to rigid ones, which measure the index to index distance, look for a link between the shapes of the time series. Thus, the distance is not measured between the *i*th element of one series and the *i*th element of the other: there can even exist a one to many relation between the positions. The DTW distance was first used in speech recognition in the 1970s, to measure the similarity between the templates representing the words and the sound that was recognized by them [28,32]. Its use was widespread in this field because it is a very robust measure in the face of the displacements, widening, and narrowing of the shapes of the series that arise due to the different speeds at which the same word is said. A few years later, due to the increase in the amount of data available, its use was suggested for the detection of patterns in time series in general, not only in those related to voice recognition [4]. Currently, DTW is probably the most common measure of elastic distance used between time series, as indicated in [23]. The current techniques most often used

Algorithm 1 Quadratic time implementation of the classical DTW algorithm.

```
1: function DTW(X, Y)
2:     M[1, 1] ← δ(x_1, y_1)
3:     for i ← 2 to |X| do
4:         M[i, 1] ← M[i − 1, 1] + δ(x_i, y_1)
5:     end for
6:     for j ← 2 to |Y| do
7:         M[1, j] ← M[1, j − 1] + δ(x_1, y_j)
8:     end for
9:     for i ← 2 to |X| do
10:        for j ← 2 to |Y| do
11:            M[i, j] ← δ(x_i, y_j) + min(M[i − 1, j], M[i, j − 1], M[i − 1, j − 1])
12:        end for
13:    end for
14:    return M[|X|, |Y|]
15: end function
```

to find the representative of a set of series using an elastic distance usually select the series of the set that most closely resembles the rest, and this kind of method has two clear drawbacks. First, a matrix of distances between all the series must be developed, which requires a huge number of calculations when the amount of data handled is large. And secondly, the series that most closely resembles the rest of the series of the set may not not remotely be the series that best sums up that set.

With respect to the implementation of the DTW, its goal is to align two sequences of characteristic vectors by deforming the time axis iteratively until an optimal match (according to a suitable metric) is found between the two sequences. A direct implementation of the classical definition of this distance gives rise to an algorithm of exponential complexity over time. Fortunately, due to the overlap of many of the sub-problems, partial results can be saved for reuse. The problem DTW(X,Y) can thus be solved with a complexity of $O(|X| \cdot |Y|)$, by means of a dynamic implementation, as shown in Algorithm 1, where $|X|$ and $|Y|$ represent the sizes of the time series. This algorithm makes use of a distance δ and a cost matrix M, in which the x-axis represents the X series and the y-axis the Y series. The position $M(i, j)$ is the cost of the best alignment of the sub-sequences X_i and Y_j. The total cost of the best alignment is thereforefound in the position $M(|X|, |Y|)$. Once we have the costs of each box, it is simple to calculate the optimal alignment by running through the lowest cost neighbors from position $M(1, 1)$ to $M(|X|, |Y|)$.

Thus, in this paper we present a method to summarize a set of time series using DTW, devoting special attention to making the algorithms scalable, so as to be applicable to large sets of series. This is possible if we address the problem with an evolutionary approach, where the computation of the centroid is treated as a minimization problem.

1.2. Formulating the computation of the centroid as a minimization problem

The problem of calculating the centroid of a set of time series $S = S_1, S_2, \ldots, S_n$ using the Euclidean distance is as simple as calculating the average of the elements of each series. Thus, if you have a set of n series with length m, the centroid $C = \{c_1, c_2, \ldots, c_m\}$ is a time series and is calculated as follows:

$$c_i = \frac{\sum_{j=1}^{n} S_{ij}}{n} \tag{1}$$

Solving this same problem using an elastic distance, such as DTW, is much more complicated. This is mainly due to the fact that when measuring the similarity between two series using an elastic distance, there may be a correspondence of several elements from one series to one of the other, rather than the one-to-one relationship that exists if the Euclidean distance is used. Therefore, the search for the centroid becomes an optimization problem, in which a series is sought that minimizes its DTW distance from the series of the set S. If we have a set of n series, the centroid C of that set is obtained by solving the following minimization problem:

$$\min_{C \in S} \sum_{i=1}^{n} DTW^2(C, S_i) \tag{2}$$

This minimization problem is resolved trivially if the sequencing space in which the centroid is sought is restricted to the set of series from which it started, that is, S. In this case, the average series must belong to the set of starting series, $C \in S$. To find the sequence that minimizes the distance to the rest it is necessary to calculate every distance $DTW(S_i, S_j),\ \forall\ i, j = \{1, ..., n\}$.

This solution therefore brings with it a complexity over time of $O(n^2)$, which is un feasible when the set of series grows. Without restricting the search space to the initial set of series, the problem is complicated. The necessary series is called

Steiner's series, related to Steiner's tree theory, and the algorithm that produces it has a temporal and spatial complexity of $O(m^n)$, which is unfeasible as soon as you have a set of several sequences. After more than 30 years of research, a scalable algorithm to calculate the Steiner series has not been found. Thus, it is proposed to develop an algorithm based on an evolutionary strategy where the use of genetic algorithms takes a non-classical approach. Furthermore, the representation of the genes is not binary and the design of the operators must depend on the representation, and on our previously stated goals.

1.3. Non-classical approaches to genetic algorithms

The classical genetic algorithm, while useful for solving many problems, has restrictions that can make it difficult to use. It can be modified to apply the crossover and mutation operations in a different order, and even define these operations in a different way, better adapting them to the problem to be solved. In the classical definition of a genetic algorithm, a chromosome is a sequence of binary symbols. However, some papers, such as [10], use alphabets of greater cardinality. Goldberg developed the theory of virtual alphabets [9], with which he shows why non-binary representations work well. Other experimental studies, in which high-cardinality alphabets are selected, have used chromosomes where each symbol represents an integer or decimal number [15,20]. In most problems, this type of representation facilitates the use of the GA because the parameters to be optimized have a numerical form, and the most natural coding of any problem is not having to perform any. Experiments have been carried out in [15] to show that representations using decimal numbers achieve better, more consistent, and faster solutions. Due to these new encodings, other types of mutation and crossover operators have also arisen, which deal more naturally with problems in which individuals are formed by decimal or integer numbers. With respect to the operators defined by the problem domain, although most GA investigations have used traditional crossover and mutation operators, some have advocated designing new operators for each problem, taking into account the specific knowledge domain of the problem. Because GAs are designed to solve real problems, incorporating knowledge of the problem makes sense if it leads to better results. In [30], it is argued that specific knowledge of the problem can be introduced into the crossover operator. This knowledge can be used to prevent generating chromosomes that are obviously bad or violate the restrictions of the problem. It can also be used to generate an initial population and prevent it from being completely random. In [10], Goldberg describes techniques for adding problem domain knowledge to operations such as crossover and mutation. Finally, another kind of approach presents a set of operators and techniques related to GAs to solve way multi-objective problems efficiently with large-scale training sets, combining evolutionary-based and heuristic-based algorithms [35], or proposing modifications of the shift-based density estimation (SDE) strategy [18].

1.4. Major contributions

Our main goal is to design an efficient algorithm based on an evolutionary strategy to find the prototype series of large sets of time series where:

- The prototype series of the set is not a series belonging to the set itself, as is the case with most classical k-medoids techniques.
- The representation of the genes is based on the similarities of the shapes of the series, by defining the concept of segment and using an elastic distance measurement.
- The efficiency of this technique is mainly based on considering an approximation of the fitness function by extracting a subset of the overall set of series. Then this subset is the one to be compared with the prototype population. This allows the method to be scalable and towork with large sets of time series.
- The crossover operator exchanges segments obtained by the alignment concept defined in DTW.
- The mutation operator modifies the shape of the time series represented by each individual in the population. Three different mutation operators are implemented, each one being applied with a certain probability. The main difference between them is that they have the capability of modifying the shape of the time series in different ways.

1.5. Structure of the paper

The rest of this paper is organized as follows: First, in Section 2, we give an overview of the problem of finding the prototype of a set of time series, focusing our attention on evolutionary approaches. Then, we introduce DBA and COMASA because of their close relationship with our proposal. Section 3 introduces the implementation of the genetic algorithm our method is based on. Lastly, Section 4 describes the experiments and compares their results with other similar techniques. Section 5 presents some conclusions and future research lines.

2. Background

Firstly, in this section we will look at different articles related to our work within the topic of the search for prototypes in time series. We shall then look at DBA and COMASA as they are the evolutionary techniques that most closely approximate our proposal.

2.1. Prototype of time series in clustering algorithms

Saeed [2] points out that clustering is a data-mining technique where similar data are placed into related or homogeneous groups without advanced knowledge of their definitions [1,26,34]. Clusters are formed by grouping objects that have maximum similarity with other objects within the same group, and this is considered a useful approach for exploratory data analysis as it identifies structures in an unlabeled dataset by objectively organizing data into similar groups. The cluster prototype or cluster representative is an essential subroutine in these approaches. One way to approach the problem of the low quality of time-series clustering is to solve the problem of inaccurate prototypes of clusters, especially in popular kinds of partitioning clustering algorithms, such as k-Means and k-Medoids, which require a prototype. In these algorithms, the quality of the clusters is fully dependent on the quality of the prototypes. Given time series in a cluster, it is clear that the cluster prototype must minimize the distance between all the elements of the group with respect to it, or in other words, it can be stated that the prototype is the time series that minimizes a Steiner sequence [13]. Rajesh et al. in [27] presents and classifies the methods for calculating prototypes published in the literature on time series. However, most of these publications have yet to prove the correctness of their methods [21]. Generally, there are three approaches for defining the prototypes:

k-medoids approaches. In these methods, the center of a cluster is defined as that time series of the group which minimizes the sum of squared distances to the other objects in the cluster. For each cluster, the similarity of all pairs of time series within the cluster's prototype is calculated using a distance measure, such as Euclidean or DTW. Cluster medoid approaches do not create a new element, they simply select the element that behaves best. This is very common in work related to time-series clustering, and has been used in many papers, such as [14,19].

Averaging approaches. If the time series have the same length and the distance metric is non-elastic (e.g., Euclidean distance) then the search for the prototype is a simple averaging technique which obtains the mean of the time series at each point. However, if the time series have different lengths [21] or when the similarity between the time series is based on similarity in shape, that one-to-one mapping does not allow the actual average shape to be captured. So, in such cases, it is appropriate to use distances such as Dynamic Time Warping (DTW) [36] or Longest Common Sub-Sequence (LCSS) [11].

Local search approaches. In this kind of approach, first the medoid of the cluster is computed, and then an averaged prototype is calculated based on warping paths. Next, new warping paths are calculated to the averaged prototype. Hautamaki et al. [14] propose a prototype obtained by local search. They apply medoid, average and local search on k-Medoids, Random Swap (RS) and Agglomerative Hierarchical clustering (where k-means is used to fine-tune the output) to evaluate their work. However, it is not clear how much improvement is obtained. A generalization of the use of local search is the use of evolutionary algorithms, where a collection of refined prototypes is maintained. An example of this is the use of genetic algorithms both to search for such prototypes [17,24,29] and to segment the time series [8,31] in the search for a simplification of the process. The problem which leads to the low accuracy of the clusters is the poor definition or updating method of the prototypes in the time-series clustering process, especially in partitioning approaches. Many clustering algorithms suffer from low accuracy of the representation methods [14]. Moreover, an inaccurate prototype can affect the convergence of the clustering algorithm, which means the resulting clusters are of low quality [21].

In our proposal, a local search approach is used to find the prototypes where a specific GA can be designed that takes a non-elastic distance metric, specifically, the DTW. We now introduce DBA and COMASA, due to their close relationship with our algorithm.

2.2. Centroid search with DBA and COMASA

The two most prominent algorithms in the literature on the search for the centroid of a set of time series using DTW as distance are DBA and COMASA. Both techniques are intimately related. These algorithms were introduced in [23] and [25], respectively.

2.2.1. DBA

This algorithm consists of improving iteratively a hypothetical mean sequence C of the set of series $S = \{S_1, \ldots, S_n\}$, initially established randomly. Given the nature of the alignment between two series provided by DTW, it is intuitively possible to split the sum of the distances between each series of S and C into the sum between each element of C and its related elements of each sequence in S. Thus, it is possible to establish the contribution of each element S to the final result in C. Note that a single element of one of the series in S can be related, and therefore contribute, to the sum of the total distances with more than one element of C. The fundamental idea of this method is to obtain the barycentre of each element of C and the associated elements of each series in S, in such a way that each of the partial sums that constitute the optimization problem is minimized. In other words, the objective of DBA is to obtain a sequence whose elements are the barycentres of the related elements of the sequences of S with each element of C.

2.2.2. COMASA

This method consists of a genetic algorithm that represents the genotype of its individuals using the so-called compact multiple alignment method, which is based on the idea of multiple alignment. Multiple alignment consists of a generalization of the alignment provided by DTW, whereby the alignment can be calculated between n sequences. Thus, for example,

Table 1
Compact multiple alignment of A={6, 20, 5, 5, 9}, B={5, 7, 20, 5, 5}, D={5, 5, 5, 20, 5} and representative sequence C.

A	{ 6 }	{ 20 }	{ 5,5,9 }
B	{ 5, 7 }	{ 20 }	{ 5,5 }
D	{5, 5, 5}	{ 20 }	{ 5 }
C	11	20	$\frac{34}{3}$

instead of calculating DTW by comparing three values in a matrix, three sequences can be aligned by comparing seven values in a three-dimensional matrix (cube). In the same way, DTW can be calculated in an n-dimensional matrix to obtain the alignment between n sequences. However, calculating the multiple alignment of n sequences of length m has a temporal and spatial complexity of $O(m^n)$, which makes it unfeasible for more than a small number of sequences. By means of this generalization of DTW, the multiple alignment of a set $S = \{S_1, \dots S_n\}$ of sequences is $W = w_1, \dots, w_k, \dots, w_L$, where w_k is determined by n indices, each relative to a time series of the set. From this alignment, the mean sequence is obtained as follows:

$$C = \lambda(S_1(w_1(1)), \dots, S_n(w_1(n))), \dots, \lambda(S_1(w_L(1), \dots, S_n(w_L(n))) \tag{3}$$

where λ is the function that calculates the arithmetic mean of the arguments. $S_j(w_i(j))$ denotes the time series j and the alignment i. Recall that in a position of alignment given by DTW, there is a relationship between two elements, one relating to each of the series on which it is applied. In this case, being a generalization of DTW to n series, a multiple alignment position consists of a relationship between n elements where each of them belongs to one of the n series. Using the above definition, it can be shown that making use of multiple alignment provides the possibility of obtaining the average sequence of an assembly by averaging those elements that are related. However, it is impossible to address this problem by calculating the optimal alignment between a set of series. Thus, COMASA seeks a way to address this problem by combining genetic algorithms with the idea of multiple alignment. The idea of compact multiple alignment consists of a representation of the average sequence of C by aligning its indices with the series indices of the set S as shown in Table 1.

Each compact alignment constitutes a gene representing an individual, and the complete genotype that defines an individual is formed by the compact multiple alignment, i.e. the compact alignment of each series S_i with the mean sequence C. The essential operators of this genetic algorithm are listed below:

- Generation: The generation of individuals randomly is not trivial. Each individual must generate n genes, which consist of compact alignments. It starts from a single sequence of indices which it then divides iteratively into sets until a compact alignment of length m is obtained. In order for the genes to be appropriately generated, a number of properties of the multiple compact alignments are explained in [25].
- Crossover: Each genotype has n genes and each daughter, therefore, must also have n genes. The way two individuals cross each other is by exchanging a gene between them, giving rise to two new individuals.
- Mutation: This consists of joining two sets of one gene and splitting another set in two. As with the classical mutation operator, a number of properties must be satisfied.
- Evaluation: Consists of the initial objective of obtaining a sequence that minimizes the DTW distance to the other sequences of S. Therefore, an individual's fitness is calculated as the sum of the DTW distance from the individual's phenotype to each sequence of S.

Major limitations of COMASA involve the complexity of the operators. This will be detailed in Section 4, where it will be shown that the temporal complexity is quite large if the set of series is large and if they have many elements.

3. Proposed algorithm: GA-segments

The goal of this section is to describe the implementation of a GA with specific operators linked with the domain of the problem. This algorithm is referred to as GA-segments. Firstly, the representation of the individuals and the meaning of the basic operators will be described. We will then focus our attention on describing in detail the particular design of each operator in the genetic algorithm. Before starting, the general operation of a GA is outlined in Algorithm 2.

3.1. Population and basic description of the operators

Each individual constitutes a candidate series to be the centroid of the whole, a prototype. In this way, the genotype and phenotype of each individual coincide. We can formalize the representation of the total dataset of time series as:

$$S = \{S_1, \dots, S_n\} \tag{4}$$

The population in a generation t is represented as follows:

$$P_t \subset S \tag{5}$$

Algorithm 2 Genetic Algorithm.

```
Initial population is generated..
Fitness of each individual is calculated.
while not all generations have been executed or have not converged do
    repeat
        Select two individuals from the population.
        Crossing and mutation reproduces those two individuals.
        Two new individuals are obtained.
        Compute the fitness of new individuals.
        Add the new individuals into the population.
    until all generations have been executed or have converged.
end while
```

where each individual is a time series S_i represented by

$$S_i = (S_{i1}, \ldots, S_{im}) \tag{6}$$

The proposed operators are:

- Generation: Individuals in the starting population are randomly selected series from the complete dataset of series.
- Evaluation: The individual's fitness is approximated by measuring the DTW distance of the individual only from a subset of time series of the dataset. The smaller the value of the distance the more fit the individual will be. Formally, the fitness of an individual S_i is calculated as:

$$F(S_i) = \sum_{j=1}^{|S'|} DTW^2(S_i, S'_j) \tag{7}$$

  where $S_i \in P_t$ and $S'_j \in S'$ for $S' \subseteq S$, i.e., S' is a subset of series randomly chosen from the time series dataset S.
- Selection: A tournament selection strategy is used. In this way, the diversity of the population is guaranteed by providing opportunities for individuals who are not well suited to survive to the next generation.
- Crossing: This operator is defined using the concept of alignment between the two time series to be crossed.
- Mutation: Three mutation operators have been defined, in order to modify the shape of the individual in different ways.

Now, the proposed configuration for every operator is detailed.

3.2. Fitness operator

The design of this operator, introduced in this new genetic algorithm, evaluates an individual by selecting only a subset of series with which to calculate the fitness. In this way, when the set of series is large, evaluating an individual with only part of the set selected at random can allow for acceptable results to be obtained in a much shorter time. At the beginning of each generation, a subset S' of the set S of series is randomly selected. All offspring of that generation's population will be evaluated with S' alone. Note that unlike the other genetic algorithms, it is necessary to re-evaluate all individuals who will become part of the new generation, and not just those who have undergone some change. In this way the situation where an individual maintains a very good fitness that was obtained with a subset S' formed by a selection of favorable series, and that does not generalize well for all S, can be avoided. This variation introduced in the evaluation operator, as will be shown in the experimental results later on, leads to a very important improvement in the speed of convergence of the algorithm.

3.3. Crossover operators

The idea behind the implemented crossover operator is to combine the shapes of the individuals to be crossed with each other in some significant way. Then, the segments that are defined by the alignment provided by DTW are exchanged between the two series to be crossed. The segments considered for crossover are those sections of the series that are related. This relatedness is determined by DTW by alignment. Respecting the relationships between the segments, we avoid exchanging segments with different shapes. For example, exchanging a segment of a series where there is growth with a section in which there is a decrease. Hence, this gives rise to a new individual which has nothing to do with its two parents, is probably unfit, and will soon be discarded in future generations. Fig. 2 illustrates the idea of segmental division obtained by aligning two individuals using DTW. Each of the straight lines represents a segment. Thus, in this case there would be six different segments. Such a division into segments is formalized now whilst the crossover operation is detailed in Algorithm 3.

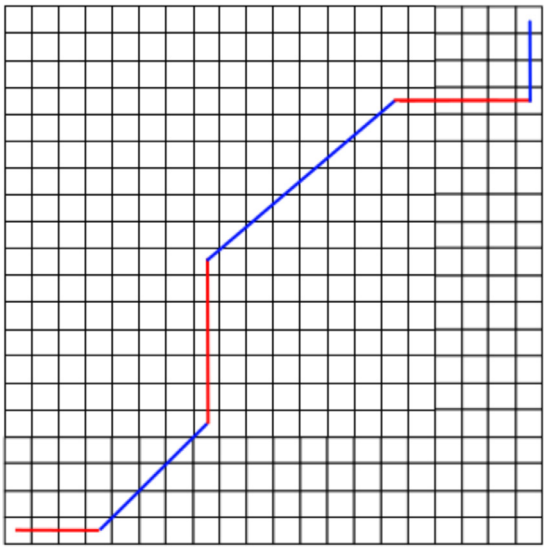


Fig. 2. Example of segments of an alignment.

Algorithm 3 Crossover algorithm.

Input: Two individuals (Time Series)
Output: Two new individuals
1. Calculate the alignment between the two individuals using DTW.
2. Split the alignment into segments.
3. Randomly select a sequence of segments.
4. Exchange the genes related to the alignment in the sequence of segments selected between the two individuals.
5. Interpolate the exchanged gene sequences.

Let $X = x_1, \ldots, x_m$ and $Y = y_1, \ldots, y_m$ be two individuals of the population. Suppose $W = w_1, \ldots w_k, \ldots, w_L$ is the optimal alignment between X and Y that results from applying DTW to both individuals, where $w_k = (w_k^x, w_k^y)$. Then $SG = s_1, \ldots, s_k, \ldots, s_p$ is defined as the sequence of segments where $s_k = w_{start}, \ldots, w_{end}$ is a sequence of alignment points. It has the following properties:

1. $s_i \cap s_j = \emptyset$, $\forall i, j \in \{1, \ldots, p\}$, $i \neq j$.
2. $\sum_{i=1}^{p} |s_i| = L$
3. If w_i is the first element of s_k and w_j is the first element of s_{k+1}, then $i < j$.

Furthermore, every segment $s_k = w_{start}, \ldots, w_{end}$, satisfies only one of the following properties:

1. $w_i^x \neq w_j^x \wedge w_i^y \neq w_j^y$, $\forall i, j \in \{start, \ldots, end\}, i \neq j$.
2. $w_i^x = w_j^x \wedge w_i^y \neq w_j^y$, $\forall i, j \in \{start, \ldots, end\}, i \neq j$.
3. $w_i^x \neq w_j^x \wedge w_i^y = w_j^y$, $\forall i, j \in \{start, \ldots, end\}, i \neq j$.

The segment sequence is, in short, a part of the alignment, that is, a relationship established by DTW between a subset of an individual's genes and another subset of another individual's genes. These genes are exchanged between the two individuals, resulting in two new individuals. Because gene sequences of different lengths may have been exchanged, the resulting individuals may not be the same size as the rest of the population. To solve this problem, individuals are re-interpolated so that they have the desired length. Figs. 3 and 4 give an example of all the steps followed to cross two individuals is represented. Fig. 3 shows the alignment between the two individuals as well as how it is segmented, with the selected sequence of segments being shaded. The genes of the individuals that are related by the sequence of segments of the selected alignment are those that cross each other. The series, with the genes corresponding to the selected alignment segments marked in red, are shown in Fig. 4. Furthermore, The same figure shows the result of the crossover with interpolation already carried out on the sections exchanged between individuals. In this example it can be observed how by means of this crossover operator, those sections of the individuals corresponding to the same forms are selected to be crossed. In this particular case, the one corresponding to the first peak has been selected. To cross individuals in this way is a great advantage over approaches that cross individuals arbitrarily.

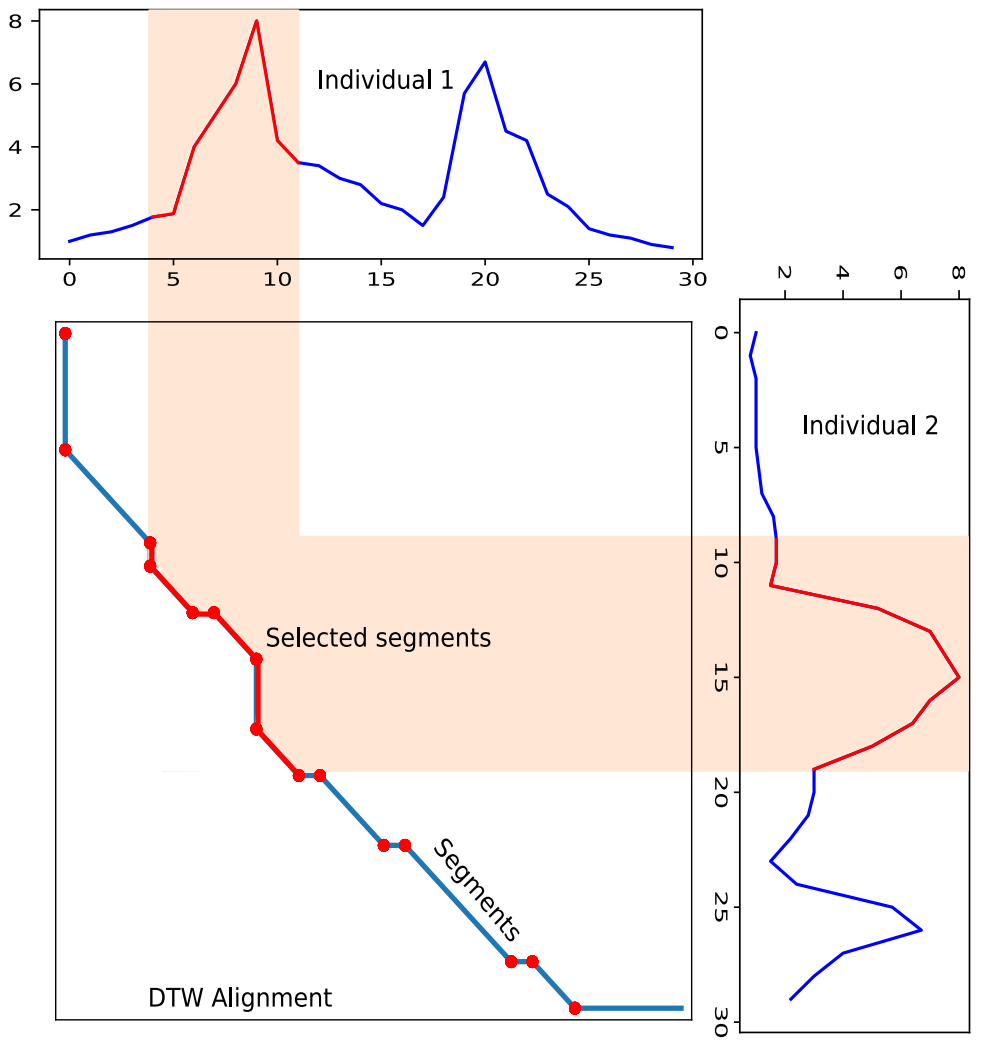


Fig. 3. Segmented alignment and selected sequence of segments to be exchanged.

3.4. Mutation operators

The mutation implemented in the algorithm aims to modify the shape of the time series represented by each individual. The operator created mutates the gene sequences and consists of three operations, shown in Fig. 5, which are applied exclusively with a certain probability. The first of these operators is that of narrowing and widening, the second is that of zone ascent or descent, and the last is that of radical ascent or descent. Through the combination of these three operators, it is possible to introduce great diversity into the mutated individuals, which is for enlarging the explored search space during the running of the genetic algorithm. Each of these operators is explained in detail below.

3.4.1. Narrowing/widening operator

The basic idea is to select two parts of the individual, and reduce the length of one while enlarging the other, preserving the shape of both. This selection of two sections of the sequence in which an individual is represented is made randomly. The preservation of the shape after the reduction or extension of the segment length is achieved by interpolation. Algorithm 4 describes the operation.

In the first step, the size of the sections to be decreased and enlarged is limited by the parameter d, whose values are between 0 and 1. The product of this parameter by the length of the individual will be the upper dimension of the size of the decrease and enlargement. So, the lower the value of d, the more conservative the mutation will be. For example, if you have an individual of 20 elements and $d = 0.25$, the set of possible values to select for decrease and expansion will be 1, 2, 3, 4, 5, since 20 will be $0.25 = 5$. A value of 3 could be selected, for example. Once the variation has been determined, the segments are randomly selected. One segment will have its original length minus 3, and the other its original length plus 3, re-interpolating them. Fig. 6 shows three examples of the individuals resulting from the application of this mutation

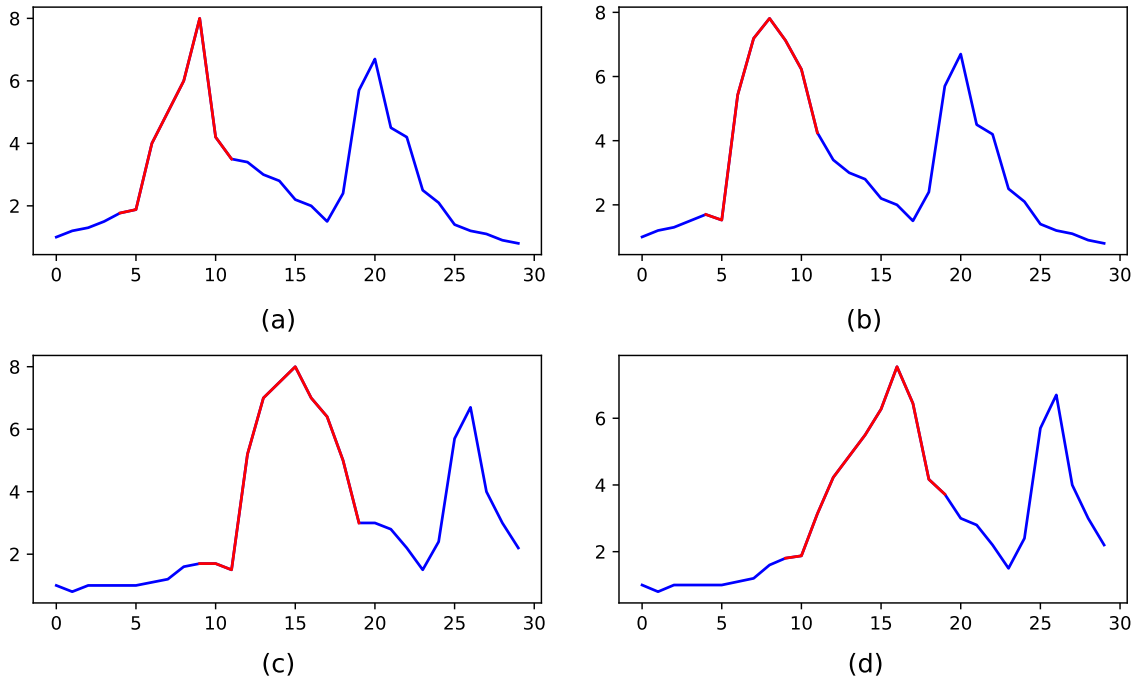


Fig. 4. (a) Individual 1, (b) new individual from 1 crossed with 2, (c) Individual 2, (d) new individual from 2 crossed with 1.

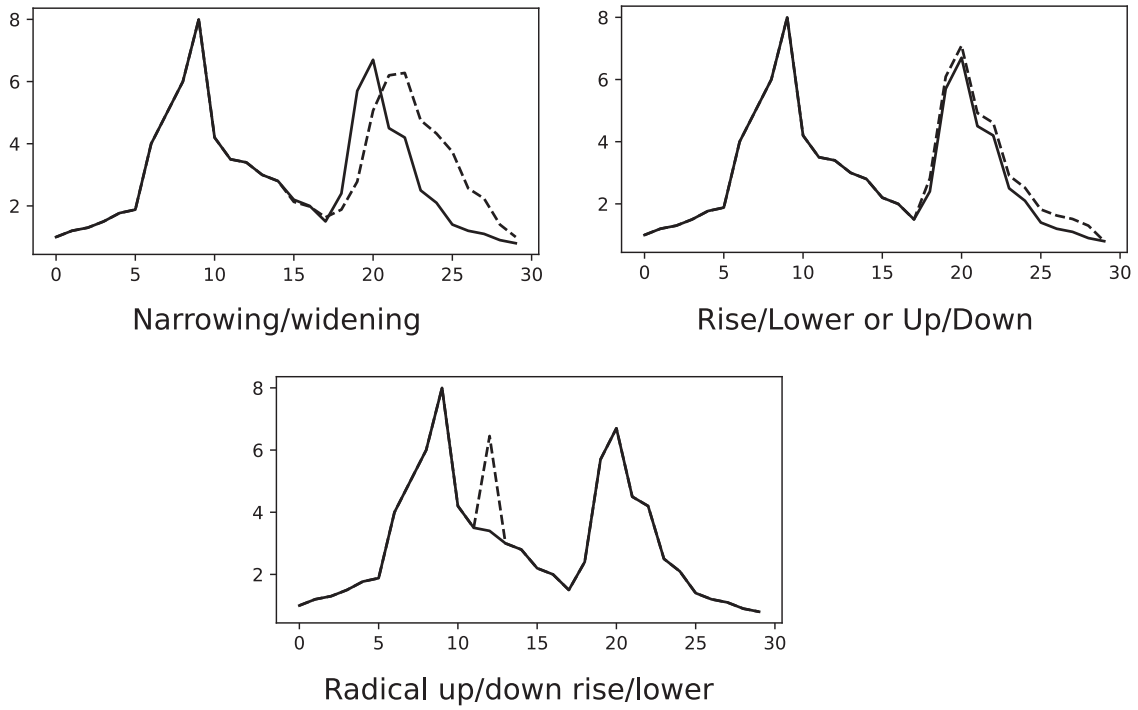


Fig. 5. Mutation operators. Only one is applied to a new generation, depending on a probability.

operator. Fig. 6(b) shows how the second peak has widened, while the flat end has narrowed. In Fig. 6(c), however, the flat part before the first peak is narrowed and the part between the two peaks are enlarged. In Fig. 6(d), the first peak has widened, while the second one has narrowed. These examples show that as the selected value for decreasing and expanding segments increases, the mutation is less conservative. Therefore, it is important to look for a value of d that appropriately limits the maximum size of the narrowing and widening applied to the individuals.

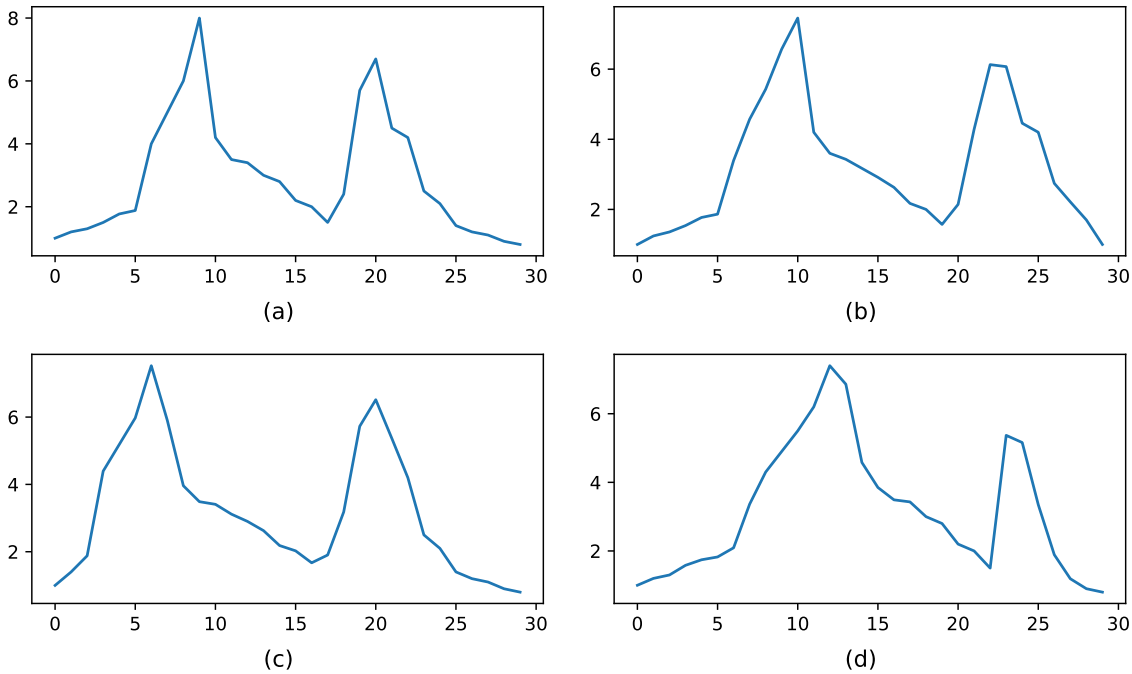


Fig. 6. (a) Original Individual, (b) variation= 3, (c) variation= 4, (d) variation= 5.

Algorithm 4 Narrowing/widening algorithm.

Input: Individual (Time Series)
Output: New mutated individual
1. Randomly determine the size of the section reduced/enlarged.
2. Randomly select two segments of the individual.
3. Randomly decide which segment is narrowed and which is widened.
4. Re-interpolate both segments are to the desired lengths.

Table 2
Parametrization of the Mutation operator narrowing/widening.

Data / d	0.01	0.03	0.04	0.05
50 words	23908.74	**23400.04**	23635.89	23406.00
cricket	243793.95	214443.83	**188209.90**	237195.99
earthquakes	**387736.93**	391467.28	390560.85	397946.66
gestures	447687.35	**353267.13**	421639.16	410236.65
lighting	332044.69	357936.59	**311837.78**	321764.33
medical_images	4958.69	4634.98	4976.01	**4481.02**
synthetic_control	14499.64	**14000.29**	14473.84	14610.61
yoga	441674.15	**359496.31**	377992.37	400300.82

In order to have more information about the effects of modifying the parameter d in Table 2, we show the results of a set of experiments with some datasets. Our algorithm for finding prototypes has been executed for each dataset three times, and the fitness average of the best individuals has been calculated. This was done in this way with the aim of minimizing the importance of randomness in the results. The results are measured for different values of d, ranging from 0.01 to 0.05. It was tested only within this range of d because with higher values the mutation is very aggressive and makes convergence difficult. Thus, it can be seen that the best value of d for these particular datasets is 0.03, since, in four of the eight sets of series, it gives the best results; in some sets, such as for example *gestures*, the difference from the other values is quite significant. So, in the experimental part we use 0.03 as the value of d.

The variability provided by this mutation operator is key in the algorithm here described, because it is capable of mutating forms that are more relevant to DTW than the mutation of a single gene alone.

Algorithm 5 Raise/Lower algorithm.

Input: Individual (Time Series)
Output: Modified individual
1. A section of the individual, i.e. a gene sequence is randomly selected.
2. The intensity of the vertical shift of the genes in the selected section is randomly selected using a Gaussian distribution of zero mean and standard deviation equal to σ.
3. The displacement obtained in the previous step is applied to each of these genes, with a slight random variation in each one, determined by a uniform distribution.

Table 3
Parametrization of the Mutation operator raise/lower.

Data / σ	0.01	0.02	0.025
50 words	22616.13	**21248.76**	23557.50
cricket	231119.11	201284.65	**186319.48**
earthquakes	**388314.75**	396258.51	390698.65
gestures	356672.81	**315414.88**	440236.65
lighting	360418.96	**338578.12**	356509.14
medical_images	5207.02	**4391.12**	5192.96
synthetic_control	15548.99	**14973.14**	15113.83
yoga	457962.26	**377992.37**	448300.82

Table 4
Parametrization of the Mutation operator radical raise/lower.

Data / σ	0.1	0.2	0.3
50 words	21209.07	21662.21	**20506.84**
cricket	224934.18	214596.13	**180275.04**
earthquakes	393012.37	**385832.21**	393771.88
gestures	392303.23	328749.26	**249884.30**
lighting	**306326.56**	335398.01	325607.66
medical_images	4486.60	4236.54	**4065.57**
synthetic_control	14699.29	14389.24	**14149.21**
yoga	352508.67	323992.10	**287021.49**

3.4.2. Raise or lower

This second operator also uses the idea of mutating forms rather than isolated genes. Thus, the mutation is applied to sets of genes and these are modified together. The modification of the genes selected to be mutated by this operator consists of moving them vertically together. Algorithm 5 details this mutation operator.

In this way, sections of the individual are mutated, in which all the genes are displaced in the same way, rather than genes in isolation. As in the previous operator, in this case it is also necessary to set the value of a parameter, which in this case is the standard deviation, σ, of the Gaussian distribution. Too low a value of σ that could slow the algorithm up, because the variability would be too low. In order for the parametrization to make sense, it is necessary for the individuals of the initial population to be normalized by means of Eq. (8).

$$x'_i = \frac{x_i - \overline{x}}{\sigma^*} \tag{8}$$

where x_i is an individual gene and x'_i is the normalized gene. The average, $\overline{x}$, and the standard deviation, σ^*, are calculated from all individuals together.

Table 3 shows the experiments; three run-throughs of our algorithm, performed to select the value of σ. The algorithm was executed with the parameter values 0.005 to 0.03 on each dataset, although in this table we only show where a minimum appears. The individuals had been previously normalized. It can be noted that the best value for σ is 0.02, as it gives the best result in most datasets.

3.4.3. Radical raise/lower

This operator modifies a single gene of the individual. For this purpose, it uses a Gaussian mutation with a normal distribution with mean equal to zero and a standard deviation with a value considerably higher than that of the previous mutation operator. In this way, the mutation of a gene can be more aggressive. Table 4 shows experiments performed to select the value of the standard deviation. The results show that the best value for the standard deviation of the normal distribution of the Gaussian mutation is σ equal to 0.3.

Table 5
Series Set: Centroid Calculation Experiments (I).

Name	Series	Size
ElectricDevices	16,637	96
uWaveGestureLibrary-X	4478	315
Yoga	3300	426
StarLightCurves	9236	1024

4. Experiments

We wished to evaluate three things: the performance of the prototype generated in terms of its fitness function, the consistency of the generation of the prototype for use in classic grouping algorithms, and finally its use in classification algorithms based on the nearest prototypes.

4.1. Fitness function performance of the prototype generated by GA-Segments versus COMASA.

This section shows the results of various experiments carried out with our proposal and comparing it with COMASA. The aim of this section is to compare algorithms and not implementations. Thus, the version of the COMASA algorithm used rigorously follows the indications given in the original paper [23], and the DTW function implemented in C and the framework DEAP have been used for the implementation of both COMASA and GA-Segments. In addition, all the results that are compared have been obtained by running the different programs on the same machine and in the same conditions. The experiments conducted were focused on studying the performance of the algorithm in terms of the values of the fitness function and the time of execution when used with data in unsupervised and supervised learning tasks. In the first of these cases, we looked for a centroid among different series which did not have to follow the same pattern. In the second case, the centroid search was done on a set of series with similar patterns.

4.1.1. Experiments: Unsupervised learning

The data used for these experiments corresponds to some of the four largest sets of series in the entire UCR Archive [7]. Specifically, the text dataset StarLightCurves is the largest in the archive. The description of the data used is shown in Table 5.

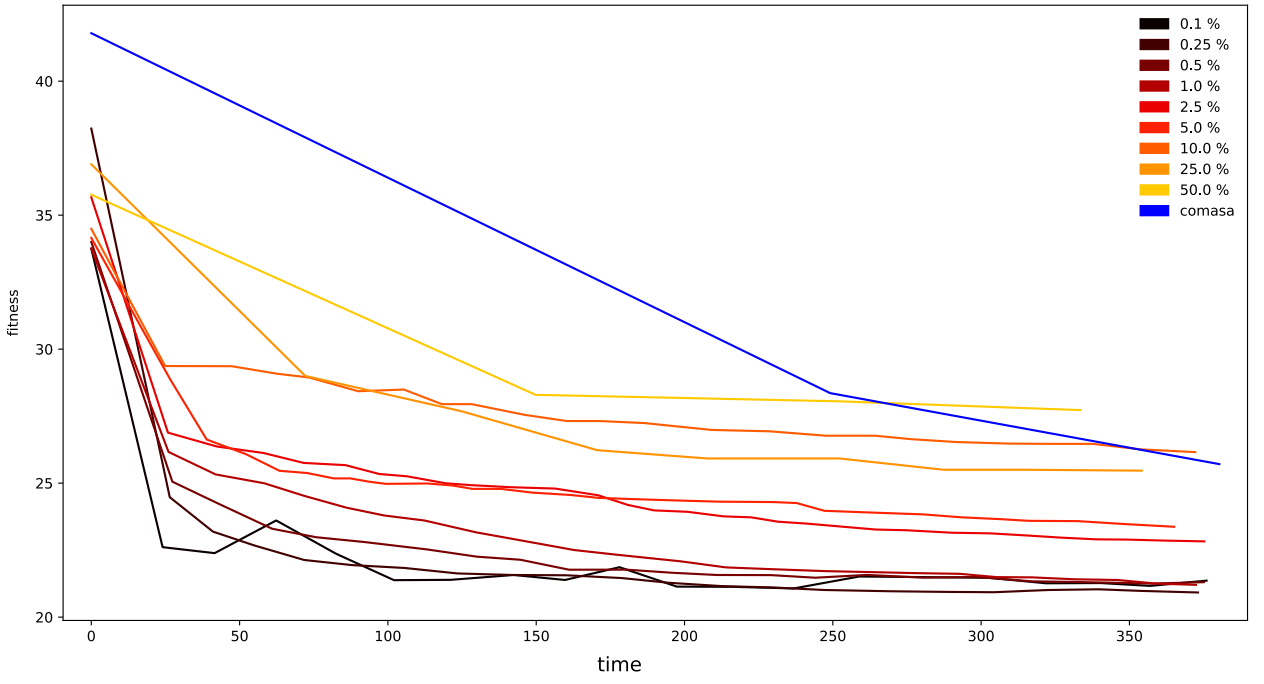
Due to memory limitations stemming from the size of the datasets used, the COMASA algorithm was not executed on the sets *yoga* and *StarLightCurves*. The experiments performed consist of running the GA-segments algorithm, using different subset sizes for evaluation, and COMASA, where possible, on these datasets. It measures the decline in fitness over time and not generations. This is because the time it takes COMASA to get a new generation is very different from the time it takes GA-Segments. There are situations in which COMASA can be up to 1000 times slower, depending, obviously, on the size of the subset chosen. Fig. 7 shows the results of COMASA and GA-segments on the *uWaveGestureLibrary-X* and *ElectricDevices* datasets. These graphs show that when the GA-segments algorithm uses a small percentage p of the complete series set, it obtains better results faster than COMASA. In these runs, COMASA makes only two generations, using about 5 Gigabytes of memory, while the GA-segments runs required about 300 Megabytes. This is because, as noted above, COMASA uses a very expensive representation of the individuals in memory, as it keeps the alignment of each sequence of the set with the hypothetical mean sequence.

Fig. 8 shows the results of *yoga* and *StarLightCurves* from the GA-segments algorithm. Looking at the results, it can be seen that the lower the value of p, the faster the evolution converges. This is because, for example, in the execution in which a $p = .1\%$ is used, 4000 generations are carried out in the same time as four generations with $p = 5\%$. This enormous difference in the number of generations running per unit time allows the evolution to be much faster. Another interesting detail is that not only do you get faster results for lower values of p, but they are better. As can be seen in the graphs, when the value of p is higher, the evolution converges to fitness values faster than when p is lower. Another interesting observation is that when p takes small values, the evolution is unstable.

4.1.2. Experiments: supervised learning

This section shows the results of experiments carried out with GA-segments and COMASA on sets of time series belonging to the same class of the UCR Archive dataset. In this case, the series from which the centroid is calculated have similar shape, so GA-segments will start from a population with a good fitness in most cases. Recall that the initialization of the population in COMASA is random, while in GA-segments it is a selection of the set of series. The datasets used for these experiments correspond to those classes of sets of series from the UCR Archive that contain more than 50 series, in order to be able to use the partial evaluation correctly, and that have a length of less than 500. This means the memory occupied in the running of COMASA is not an obstacle to completing these experiments. Fig. 9 shows some graphs similar to those shown in Fig. 7, which shows the decrease in fitness with respect to time.

The behaviour of the algorithm when performing a partial evaluation is still better than when calculating the fitness from the whole set of series. However, since they are small sets of series, the generalization capacity of the partial evaluation



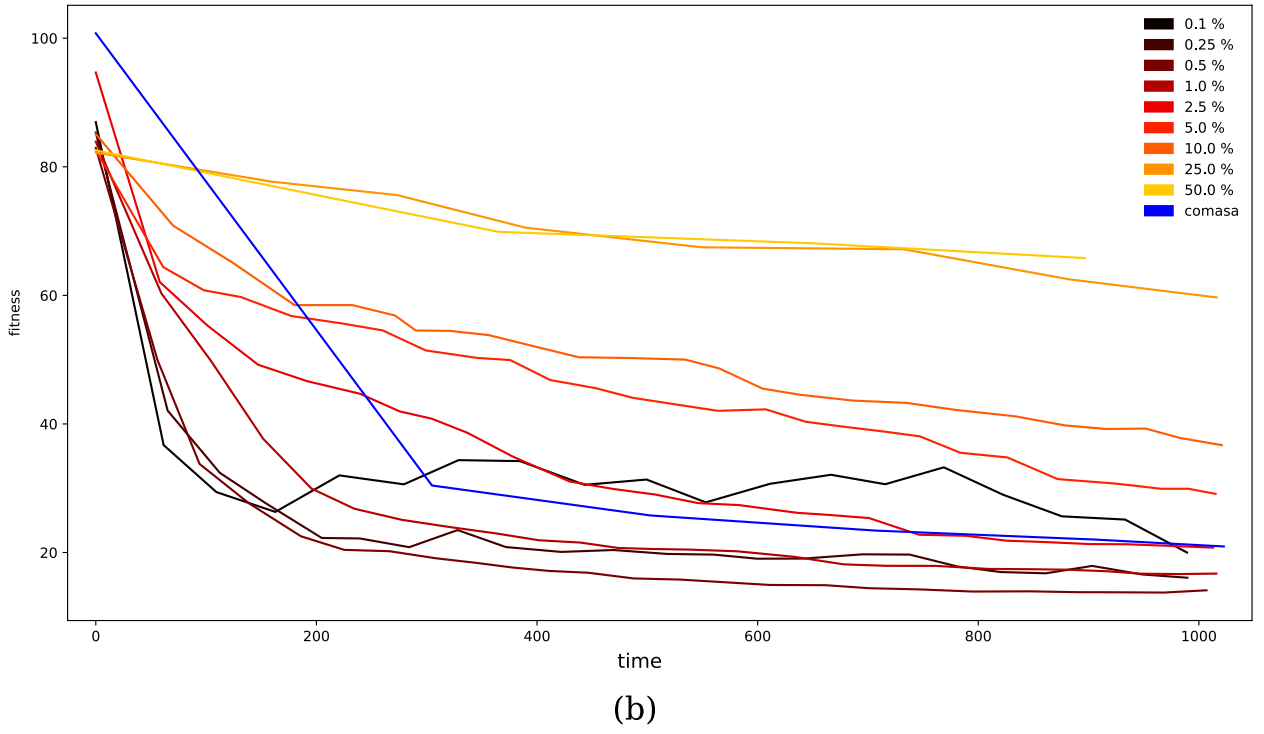


Fig. 7. Fitness curves for unsupervised learning (I) where p is the percentage of population taken: (a) ElectricDevices and (b) uWaveGestureLibrary-X.

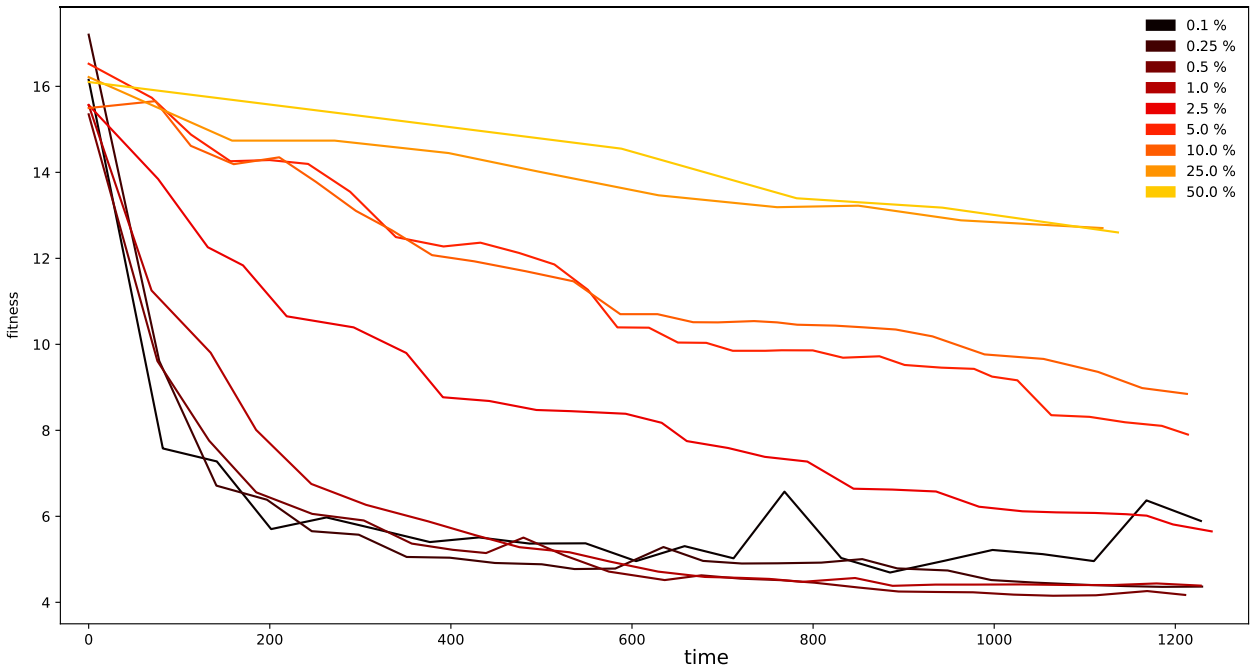


Fig. 8. Fitness curves related to unsupervised learning for Yoga data.

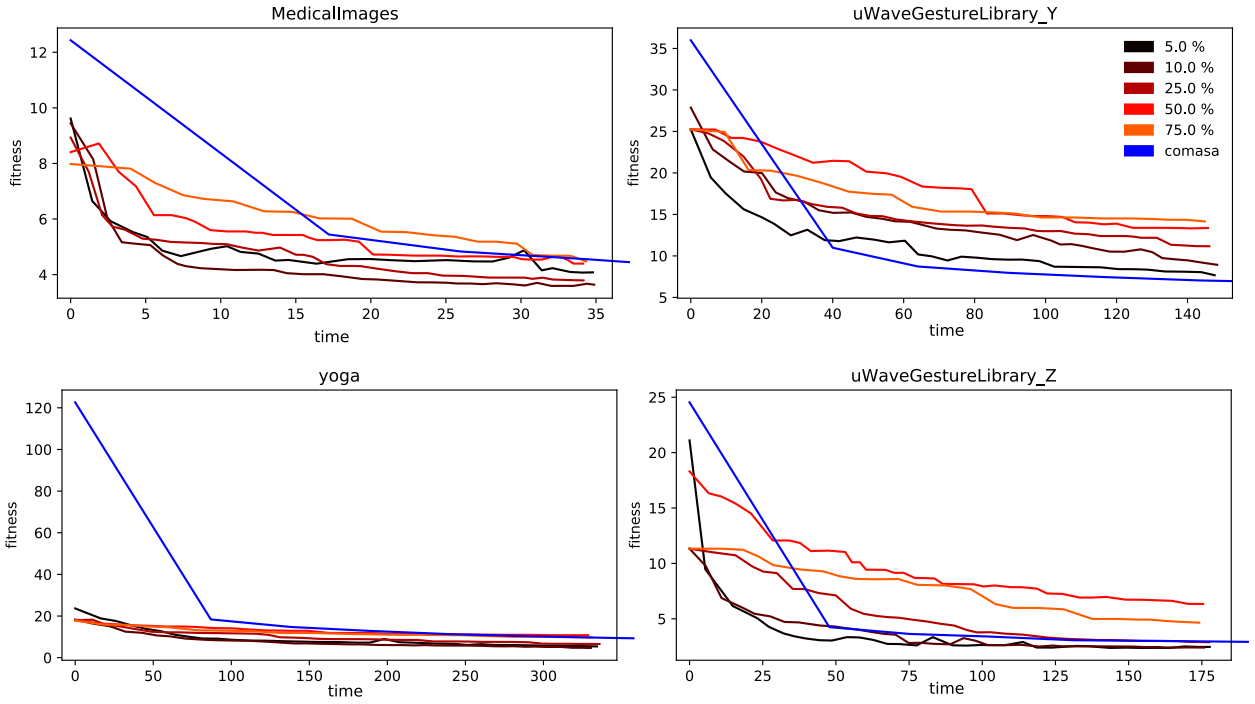


Fig. 9. Fitness curves related to supervised learning tasks.

decreases. It can also be observed that the initial population of the COMASA algorithm is worse than that of GA-segments. In the case of the set of series *yoga*, this illustrates how important the difference in population generation is when the sets of series are already very similar to each other. Due mainly to the partial evaluation operator, and the fact that it generalizes better with large datasets, the results obtained in unsupervised learning experiments are better than those obtained in supervised experiments. The reason for this is the size of the data sets used rather than their nature.

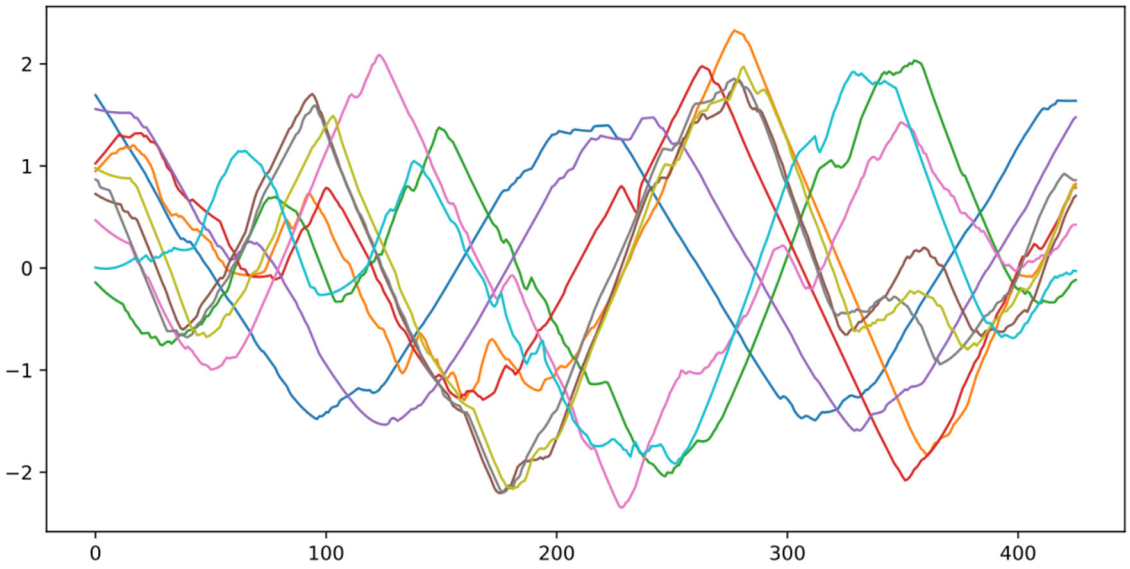


Fig. 10. Series examples from *Yoga* dataset.

4.2. Performance in clustering tasks

In order to test the performance of GA-segment in clustering algorithms, the *k*-Means algorithm has been implemented. This algorithm calculates the centroid of each group of elements in each iteration, so that in the next the elements are regrouped within that whose centroid is closer. The implementation of the algorithm for regrouping the elements uses the DTW distance between each centroid and each element instead of the Euclidean distance, and our GA-segments genetic algorithm is used to calculate the centroids. In order to reduce the dependence of the results of the *k*-Means grouping algorithm on the initialization values, several runs of the *k*-Means algorithm have been performed with different random initializations of centroids. An important challenge is described in [22], which is the inability to use the *k*-Means algorithm with the DTW distance due to the lack of a method that can calculate the centroid of a set of time series using DTW. This indicates that the centroid obtained by current methods is not good enough for this algorithm to work properly. One of the reasons given is the inconsistency of the forms between centroids obtained in different executions of the algorithm proposed in [12], as it reorders the set of series on which it was applied differently. GA-segments does not obtain the exact prototype of a set of time series, but an estimate. Even so, as shown below, the centroids obtained in different executions of the algorithm present very similar shapes, thus demonstrating the consistency of the method developed. It was decided to use a set with series that were so different that no obvious pattern could be observed, and the resulting shape of the centroid after the execution of the algorithm is not obvious. For this reason, the Yoga dataset was chosen. Some of its series are shown in Fig. 10. Fig. 11 shows the centroids obtained after running the GA-segments algorithm four times. These centroids present very similar shapes, thus showing the consistency of the GA-segments algorithm, even in a data set as chaotic as the one studied.

4.3. Performance in classifying tasks

To finalize the experiments, the behaviour of our GA-Segments algorithm in sorting tasks was studied. A sorting algorithm, such as Nearest Centroid (NC), which is a variant of the KNN algorithm, was chosen for this purpose. This algorithm, in which instead of classifying a new element by constructing a distance matrix and assigning it the majority class of its neighborhood, classifies it only according to the distances from the centroids of the classes. Therefore, it is important to obtain good representatives from the different classes in order to achieve good results. The implemented algorithm integrates the DTW into it, replacing the Euclidean distance. The use of an elastic distance, such as DTW, in the calculation of the class centroids, and in the subsequent process of measuring each series to be classified relative to the centroids, is a differentiating factor in a considerable number of data sets of a temporal nature. This is mainly due to its ability to adapt the shapes of some series deformations over time when measuring distance, as previously explained. In addition, the option has been chosen to implement the NC variant of the KNN algorithm, because it is much less complex when using a pseudo-metric distance that does not comply with the triangular inequatlity, such as DTW. The selected datasets of the UCR Archive correspond to those in which each of its classes has at least 50 series, in order to be able to use the GA-segments algorithm with the evaluation with subsets. Those assemblies whose series have a length of more than 500 have also been discarded, so that COMASA could be run properly. Following this criterion, the selected data are shown in Table 6.

A comparison of the errors of the NC classifications is shown in Table 7, using the COMASA and GA-segments (GA-Seg) algorithms for the centroid calculation. Classifications have also been made using the Euclidean distance (Euc).

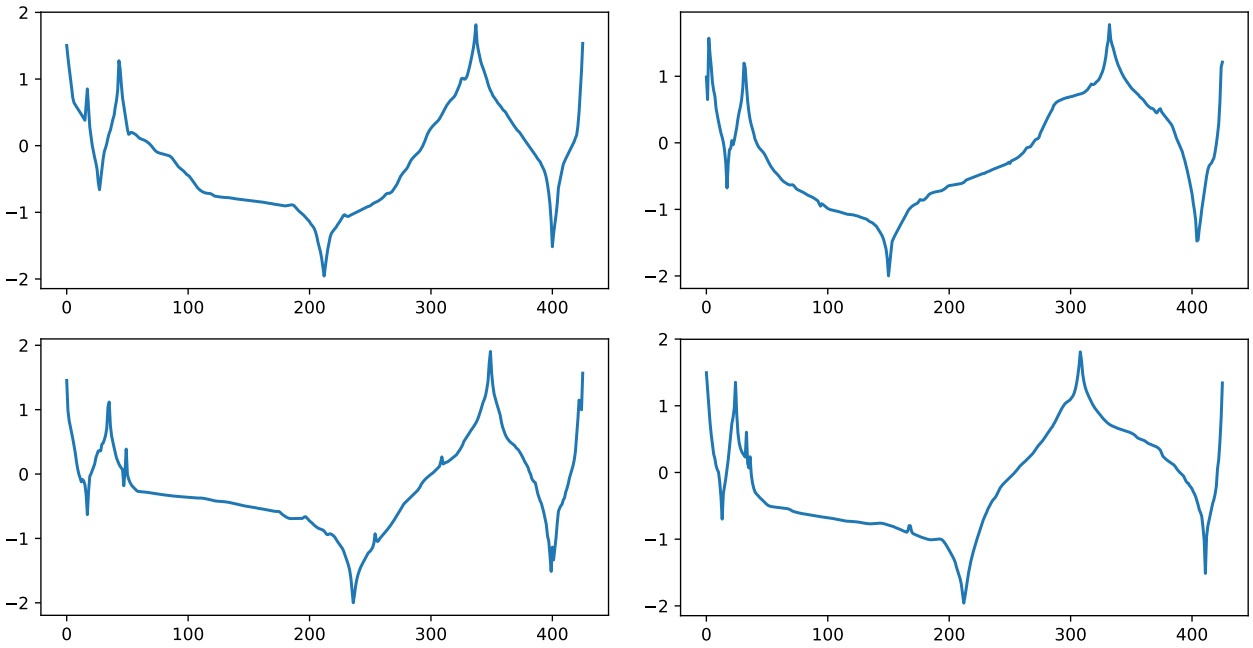


Fig. 11. Centroids of *Yoga* obtained in different executions.

Table 6
Selected data sets.

Dataset	Classes	Learning	Test	Size
Strawberry	2	370	613	235
ChlorineConcentration	3	467	3840	166
PhalangesOutlinesCorrect	2	1800	858	80
DistalPhalanxOutlineCorrect	2	276	600	80
uWaveGestureLibrary-X	8	896	3582	315
uWaveGestureLibrary-Y	8	896	3582	315
uWaveGestureLibrary-Z	8	896	3582	315
Ham	2	109	105	431
wafer	2	1000	6174	152
Two-Patterns	4	1000	4000	128
ProximalPhalanxOutlineCorrect	2	600	291	80
ProximalPhalanxOutlineAgeGroup	3	400	205	80
yoga	2	300	3000	426
MiddlePhalanxOutlineCorrect	2	291	600	80

Table 7
NC experiments.

Datasets	COMASA	GA-Seg.	Euc.
Strawberry	0.393	0.377	**0.331**
ChlorineConcentration	**0.645**	0.719	0.672
PhalangesOutlinesCorrect	0.427	**0.338**	0.369
DistalPhalanxOutlineCorrect	**0.393**	0.519	0.527
uWaveGestureLibrary-X	0.326	**0.303**	0.369
uWaveGestureLibrary-Y	0.447	**0.423**	0.450
uWaveGestureLibrary-Z	0.389	**0.386**	0.462
Ham	0.278	0.298	**0.230**
wafer	0.313	**0.279**	0.345
Two-Patterns	0.038	**0.007**	0.536
ProximalPhalanxOutlineCorrect	**0.279**	0.372	0.355
ProximalPhalanxOutlineAgeGroup	0.225	0.191	**0.181**
yoga	0.453	**0.430**	0.513
MiddlePhalanxOutlineCorrect	**0.442**	0.524	0.450

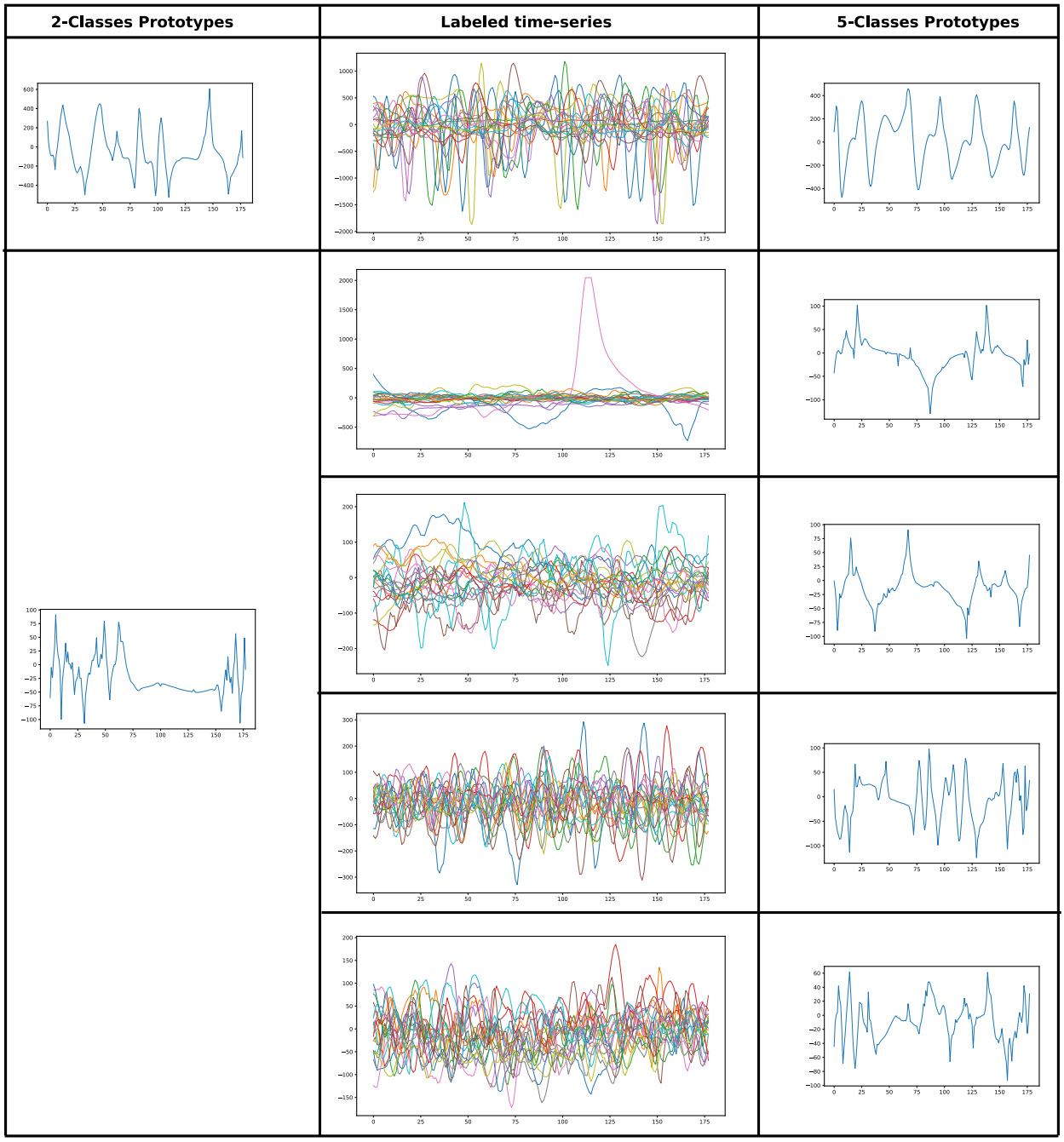


Fig. 12. Epileptic Seizure Recognition Data Set.

As reflected in Table 7, a better result is obtained, in terms of the errors of the classifications, using a non-elastic distance, such as the Euclidean, than using an elastic distance, in only three out of the 14 experiments carried out. The error is computed as $1 - accuracy$, that is, the number of labels assigned incorrectly. It can also be observed that for 9 experiments, the errors in classification obtained when calculating the centroids with GA-segments are lower than those obtained with COMASA, due mainly to the fact that the centroids obtained are more precise in most cases. It is important to note that the calculation of centroids with COMASA involved a time cost of approximately 10 times that of GA-segments.

Finally, to compare the efficacy of GA-Segments vs. algorithms with the Euclidean distance, we performed two experiments with the Epileptic Seizure Recognition electrocardiograms dataset [3], which consists of a collection of 11,500 electrocardiograms with 179 time items classified into 5 classes. Two classification experiments were considered, one where there are two classes: one with the series labeled with 1 and the other with the series labeled with 2,3,4,5. In the other experiment, all five classes were considered. Fig. 12 shows some examples of series labeled with the five classes within the

Table 8
Epileptic Seizure Recognition experiments.

Algorithm	Distance	2-Classes	5-Classes
GA-Segments	DTW	94.43%	53.13%
NC	DTW	63.26%	24.82%
KNN	Euclidean	94.47%	53.86%

prototypes obtained by GA-segments. Table 8 shows that the success ratio in the classification with the NC algorithm using the DTW distance and the prototypes obtained with GA-Segments is always better than that obtained from an algorithm with the prototypes obtained with the Euclidean distance, and is similar to the efficiency of a KNN algorithm with $k = 1$ and the Euclidean distance, but with only 2 and 5 evaluations as against the 11,500 required by the KNN due to the fact that it requires n comparisons, where n is the number of time series, in this case 11,500.

5. Conclusions and future work

This section presents the conclusions obtained from this study, analysing the achievement assessing the fulfilment of the objectives set at the beginning of the paper. It also details possible improvements and future work.

5.1. Goal fulfilment analysis

An algorithm was implemented capable of obtaining a centroid from a set of time series using DTW distance measurement. In addition, the algorithm implemented is scalable to large sets of series, achieving more accurate representations in less time than other techniques. The algorithm has also been integrated with other automatic learning algorithms, both grouping and classification, obtaining satisfactory results in both cases. It highlights the case of the classification of time series using the NC algorithm integrated with the genetic algorithm, whose results are much better than when using the Euclidean distance. Specific objectives have also been successfully achieved. The first of these related to the use of an evolutionary method for obtaining the centroid from a set of time series. The hypothesis that a genetic algorithm was the right strategy to obtain a good estimate from a representative of a set of series was verified. The second specific objective has also been attained. The genetic algorithm was correctly integrated into other automatic learning algorithms, improving the results obtained by using a rigid distance such as the Euclidean.

5.2. Future work

Although the has yielded good results, the authors think that it can be improved. A number of possibilities are therefore set out below:

For instance, new possibilities can be explored using genetic algorithms using a cooperative strategy. For example, designing some criteria according to which when a species ceases to contribute to good individuals, it is eliminated and replaced by a new one. Furthermore, in the implemented mutation and crossover operators, interpolations must be performed to keep all series at the same length. The simplest method used was linear interpolation. However, other types of interpolation could preserve the structure of the series in a more precise way, thus probably achieving better results. Finally, the DTW distance, although very useful for measuring the distance between similar series, can exhibit some undesirable behavior when there are certain types of displacements, narrowings, and widenings. In addition, its complexity is a factor to consider when the length of the series is large. For this reason, it may be interesting to use some DTW variants, such as FastDTW or DDTW, to limit the search space for the alignments. Today, there is also a considerable number of articles using techniques related to deep learning for time series classification [16], classifiers based on convolutional neural networks for large-scale training datasets [5] or parallel approaches for prediction and recognition of time series patterns in big data environments [6], it would also be useful to identify a specific field of application where specific time series could be optimized by mathematical optimization techniques comparing the results with the proposed method.

Declaration of Competing Interest

The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

Acknowledgements

Supported by Project TIN2015-64776-C3-3-R of the Spanish Ministry of Science and Innovation, co-funded by the European Regional Development Fund.

References

[1] S. Aghabozorgi, A.S. Shirkhorshidi, T.Y. Wah, Time-series clustering–a decade review, Inf. Syst. 53 (2015) 16–38.
[2] S. Aghabozorgi, T.Y. Wah, Clustering of large time series datasets, Intell. Data Anal. 18 (5) (2014) 793–817.
[3] R.G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, C.E. Elger, Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: dependence on recording region and brain state, Phys. Rev. E 64 (6) (2001) 061907.
[4] D.J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series, in: Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, in: AAAIWS'94, AAAI Press, 1994, pp. 359–370. URL: http://dl.acm.org/citation.cfm?id=3000850.3000887.
[5] J. Chen, K. Li, K. Bilal, X. Zhou, K. Li, P. Yu, A bi-layered parallel training architecture for large-scale convolutional neural networks, IEEE Trans. Parallel Distrib. Syst. (2018).
[6] J. Chen, K. Li, H. Rong, K. Bilal, K. Li, P.S. Yu, A periodicity-based parallel time series prediction algorithm in cloud computing environments, Inf. Sci. (2018), doi:10.1016/j.ins.2018.06.045.
[7] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, The UCR time series classification archive, 2015, www.cs.ucr.edu/~eamonn/time_series_data/.
[8] F.-L. Chung, T.-C. Fu, V. Ng, R.W.P. Luk, An evolutionary approach to pattern-based time series segmentation, IEEE Trans. Evol. Comput. 8 (5) (2004) 471–489, doi:10.1109/TEVC.2004.832863.
[9] D. Goldberg, The theory of virtual alphabets, Parallel Probl. Solv. Nat. (1991) 13–22.
[10] D.E. Goldberg, J.H. Holland, Genetic algorithms and machine learning, Mach. Learn. 3 (2) (1988) 95–99.
[11] L. Gupta, D.L. Molfese, R. Tammana, P.G. Simos, Nonlinear alignment and averaging for estimating the evoked potential, IEEE Trans. Biomed. Eng. 43 (4) (1996) 348–356, doi:10.1109/10.486255.
[12] L. Gupta, D.L. Molfese, R. Tammana, P.G. Simos, Nonlinear alignment and averaging for estimating the evoked potential, IEEE Trans. Biomed. Eng. 43 (4) (1996) 348–356.
[13] D. Gusfield, Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology, Cambridge university press, 1997.
[14] V. Hautamaki, P. Nykanen, P. Franti, Time-series clustering by approximate prototypes, in: Proceedings of the 19th International Conference on Pattern Recognition, IEEE, 2008, pp. 1–4.
[15] C.Z. Janikow, Z. Michalewicz, An experimental comparison of binary and floating point representations in genetic algorithms., in: Proceedings of the ICGA, 1991, pp. 31–36.
[16] F. Karim, S. Majumdar, H. Darabi, S. Chen, Lstm fully convolutional networks for time series classification, IEEE Access 6 (2018) 1662–1669.
[17] H. Kaya, S. Gndz-dc, Saga: a novel signal alignment method based on genetic algorithm, Inf. Sci. 228 (2013) 113–130.
[18] M. Li, S. Yang, X. Liu, Shift-based density estimation for Pareto-based algorithms in many-objective optimization, IEEE Trans. Evol. Comput. 18 (3) (2014) 348–365.
[19] T.W. Liao, C.-F. Ting, P.-C. Chang, An adaptive genetic clustering method for exploratory mining of feature vector and time series data, Int. J. Prod. Res. 44 (14) (2006) 2731–2748.
[20] Z. Michalewicz, C.Z. Janikow, Handling constraints in genetic algorithms., in: Proceedings of the ICGA, 1991, pp. 151–157.
[21] V. Niennattrakul, C.A. Ratanamahatana, Inaccuracies of shape averaging method using dynamic time warping for time series data, in: Y. Shi, G.D. van Albada, J. Dongarra, P.M.A. Sloot (Eds.), Proceedings of the Computational Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 513–520.
[22] V. Niennattrakul, C.A. Ratanamahatana, On clustering multimedia time series data using k-means and dynamic time warping, in: Proceedings of the International Conference on Multimedia and Ubiquitous Engineering, IEEE, 2007, pp. 733–738.
[23] F. Petitjean, P. Gancarski, Summarizing a set of time series by averaging: from Steiner sequence to compact multiple alignment, Theor. Comput. Sci. 414 (1) (2012) 76–91.
[24] F. Petitjean, P. Gancarski, Summarizing a set of time series by averaging: from Steiner sequence to compact multiple alignment, Theor. Comput. Sci. 414 (1) (2012) 76–91.
[25] F. Petitjean, A. Ketterlin, P. Gançarski, A global averaging method for dynamic time warping, with applications to clustering, Pattern Recognit. 44 (3) (2011) 678–693.
[26] S. Pravilovic, M. Bilancia, A. Appice, D. Malerba, Using multiple time series analysis for Geosensor data forecasting, Inf. Sci. 380 (2017) 31–52.
[27] T. Rajesh, Y.S. Devi, K.V. Rao, Hybrid clustering algorithm for time series data - a literature survey, in: Proceedings of the International Conference on Big Data Analytics and Computational Intelligence (ICBDAC), IEEE, 2017, pp. 343–347.
[28] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Trans. Acoust. Speech Signal Process. 26 (1) (1978) 43–49, doi:10.1109/TASSP.1978.1163055.
[29] P. Siirtola, P. Laurinen, J. Roning, Mining an optimal prototype from a periodic time series: an evolutionary computation-based approach, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2009, pp. 2818–2824, doi:10.1109/CEC.2009.4983296.
[30] J.Y. Suh, D. Van Gucht, Incorporating Heuristic Information into Genetic Search, in: Proc. 2rd Int. Conf. Genetic Algorithms, Lawrence Erlbaum Associates, 1987, pp. 100–107.
[31] C.M. Taylor, A. Salhi, On partitioning multivariate self-affine time series, IEEE Trans. Evol. Comput. 21 (6) (2017) 845–862, doi:10.1109/TEVC.2017.2688521.
[32] T.K. Vintsyuk, Speech discrimination by dynamic programming, Cybern. Syst. Anal. 4 (1) (1968) 52–57.
[33] 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 Min. Knowl. Discov. 26 (2) (2013) 275–309.
[34] R. Xu, D. Wunsch, Survey of clustering algorithms, IEEE Trans. Neural Netw. 16 (3) (2005) 645–678, doi:10.1109/TNN.2005.845141.
[35] Y. Xu, K. Li, J. Hu, K. Li, A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues, Inf. Sci. 270 (2014) 255–287.
[36] Z. Zhang, R. Tavenard, A. Bailly, X. Tang, P. Tang, T. Corpetti, Dynamic time warping under limited warping path length, Inf. Sci. 393 (2017) 91–107.