RAIRO-Oper. Res. 53 (2019) 243–259
https://doi.org/10.1051/ro/2018089

RAIRO Operations Research
www.rairo-ro.org

GRASP HEURISTIC FOR TIME SERIES COMPRESSION WITH PIECEWISE AGGREGATE APPROXIMATION

Vanel Steve Siyou Fotso*, Engelbert Mephu Nguifo and Philippe Vaslin

Abstract. The Piecewise Aggregate Approximation (PAA) is widely used in time series data mining because it allows to discretize, to reduce the length of time series and it is used as a subroutine by algorithms for patterns discovery, indexing, and classification of time series. However, it requires setting one parameter: the number of segments to consider during the discretization. The optimal parameter value is highly data dependent in particular on large time series. This paper presents a heuristic for time series compression with PAA which minimizes the loss of information. The heuristic is built upon the well known metaheuristic GRASP and strengthened with an inclusion of specific global search component. An extensive experimental evaluation on several time series datasets demonstrated its efficiency and effectiveness in terms of compression ratio, compression interpretability and classification.

Mathematics Subject Classification. 90C59.

Received April 30, 2017. Accepted October 12, 2018.

1. Introduction

Time series databases are often large and several transformations have been introduced in order to represent them in a more compact way. One of these transformations is Piecewise Aggregate Approximation (PAA) [13], which consists in dividing a time series into several segments of fixed length and replacing the data points of each segment with their averages. Due to its simplicity and low computational time, PAA has been widely used as a basic primitive by other temporal data mining algorithms such as [16, 17, 25], in order

- to construct symbolic representations of time series [3, 26];
- to construct an index for time series [12, 14, 30]. Indeed, PAA allows queries which are shorter than length for which the index was built. This very desirable feature is impossible with Discrete Fourier Transform, Singular Value Decomposition and Discrete Wavelet Transform;
- to classify time series.

1.1. Why the use of PAA can improve alignment with Dynamic Time Warping

Time series comparison is an important task that can be done in two main ways. Either the comparison method considers that there is no time distortion as in Euclidian distance (ED), or it considers that some small time distortions exist between time axis of time series as in Dynamic Time Warping (DTW) alignment algorithm [29]. Since time distortion often exists between time series, DTW has often better results than the ED

Keywords. Time series, optimization, compression, classification.

University Clermont Auvergne, CNRS, LIMOS, 63000 Clermont-Ferrand, France.
*Corresponding author: vsiyou@gmail.com

[5]. An exhaustive comparison of time series algorithms [1] shows that DTW is among the efficient techniques to be used. However, DTW has two major drawbacks: the comparison of two time series under DTW is time-consuming [20] and DTW sometimes produces pathological alignments [15]. A pathological alignment occurs when, during the comparison of two time series X and Y, one datapoint of the time series X is compared to a large subsequence of datapoints of Y. A pathological alignment causes a wrong comparison.

Three categories of methods are used to avoid pathological alignments with DTW:

- The first one adds constraints to DTW [4, 11, 21–23, 28]. The main idea here is to limit the length of the subsequence of a time series that can be compared to a single datapoint of another time series.
- The second one suggests skipping datapoints that produce pathological alignment during the comparison of two time series [10, 18, 19].
- The third one proposes to replace the datapoints of time series with a high-level abstraction that captures the local behavior of those time series. A high-level abstraction can be a histogram of values that captures the distribution of time series datapoints in space [29] or a feature that captures the local properties of time series, such as the trend with Derivative DTW (DDTW) [15].

Another simple but yet interesting way to capture local properties of time series is to consider mean of segments of the time series as PAA does. Indeed, the use of the mean reduces the harmful effects of singularities contained in the data and thus allows to avoid pathological alignments. However, one major challenge with PAA is the choice of the number of segments to consider especially with long time series.

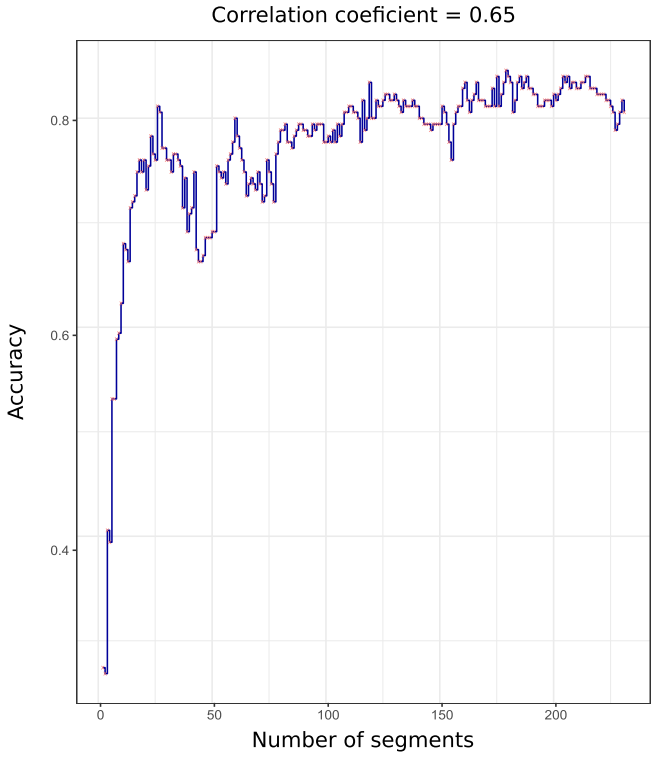


FIGURE 1. Relation between Accuracy and the number of segment on FISH dataset. The accuracy is computed from the algorithm one nearest neighbor (1NN) associated with PDTW. When the number of segments considered is very small, there is a loss of information and the accuracy is reduced. However, considering all the points in the time series, we also do not obtain maximum accuracy due to the presence of noise or singularities [15] in the data.

1.2. The problem of choosing a suitable segment number for PAA

If the number of segments considered with PAA is too small, the resulting representation is compact, but it contains less information. On the other hand, if the number of segments is too large, the obtained representation is less compact and more prone to the noise contained in the original time series (Fig. 1). Our idea is that a number of segments for PAA will be considered as good if it allows obtaining a compact representation of the time series, and also if it preserves the quality of the alignment of time series. So when considering classification task, one of the best classification algorithm to use for evaluating the quality of time series alignment is one nearest neighbor (1NN). Indeed, its classification error directly depends on time series alignment, since 1NN has no other parameters [27].

1.3. Summary of contributions

In this paper,

- We define the problem of preprocessing time series with PAA for a better classification with DTW.
- We propose a parameter free heuristic for aligning piecewise aggregate time series with DTW, which approximates the optimal value of the number of segments to be considered with PAA.
- We make our source code and all our results available to allow the reproducibility of our experiments.

The rest of the paper is organized as follows: in Section 2 we recall the definitions and background; Section 3 explains our approach; Section 4 presents experimental results and comparisons to others methods; Section 5 offers conclusions and venues for future work.

2. Background and related works

Let's recall some definitions.

Definition 2.1. A time series $X = x_1, \cdots, x_n$ is a sequence of numerical values representing the evolution of a specific quantity over time. x_n is the most recent value.

Definition 2.2. A segment X_i of length l of the time series X of length n ($l < n$) is a sequence constituted by l variables of X starting at the position i and ending at the position $i+l-1$. We have: $X_i = x_i, x_{i+1}, ..., x_{i+l-1}$.

Definition 2.3. The arithmetic average of the data points of a segment X_i of length l is noted $\bar{X}_i$ and is defined by:

$$\bar{X}_i = \frac{1}{l} \sum_{j=0}^{l-1} x_{i+j}. \tag{2.1}$$

Definition 2.4. Let T be the set of time series. The Piecewise Aggregate Approximation (PAA) is defined as follows:

$$\begin{aligned} &\text{PAA}: T \times \mathbb{N}^* \to T \\ &(X, N) \mapsto \text{PAA}(X, N) = \begin{cases} \bar{X}_k, k \in \{i \times \frac{n}{N} + 1, i = 0, \cdots, N-1\} \; if \; N < |X|, \\ X \text{ otherwise.} \end{cases} \end{aligned} \tag{2.2}$$

Definition 2.5. Let $d \subseteq T$ be a subset of time series, $N \in \mathbb{N}^*$, $\text{PAAset}(d, N) = \{\text{PAA}(X, N), \quad \forall X \in d\}$

2.1. Dynamic Time Warping algorithm.

DTW [22] is an algorithm of time series alignment algorithm that performs a non-linear alignment while minimizing the distance between two time series. To align two time series : $X = x_1, x_2, \cdots, x_n$; $Y = y_1, y_2, \cdots, y_m$, the algorithm constructs an $n \times m$ matrix where the cell (i, j) of the matrix corresponds to the squared distance $(x_i - y_j)^2$ between x_i and y_j. Then to find the best alignment between X and Y, DTW constructs the path that minimizes the sum of squared distances. This path, noted $W = w_1, w_2, \ldots, w_k, \ldots, w_K$, must respect the following constraints:

- Boundary constraint: $w_1 = (1, 1)$ and $w_K = (n, m)$
- Monotonicity constraint: given $w_k = (i, j)$ and : $w_{k+1} = (i', j')$ then : $i \leq i'$ and $j \leq j'$
- Continuity constraint: given $w_k = (i, j)$ and : $w_{k+1} = (i', j')$ then : $i' \leq i + 1$ and : $j' \leq j + 1$

The warping path is computed by an algorithm based on the dynamic programming paradigm that solves the following recurrence:

$$\begin{array}{l}\gamma(i, j) = d(x_i, y_j) + \min\{\gamma(i - 1, j - 1),\\ \gamma(i - 1, j), \gamma(i, j - 1)\},\end{array} \tag{2.3}$$

where $d(x_i, y_j)$ is the squared distance contained in the cell (i, j) and $\gamma(i, j)$ is the cumulative distance at the position (i, j) that is computed by the sum of the squared distance at the position (i, j) and the minimal cumulative distance of its three adjacent cells.

Piecewise Dynamic Time Warping Algorithm (PDTW) [14] is the DTW algorithm applied on Piecewise Aggregate time series [13]. Let $N \in \mathbb{N}^*$, X and Y be two time series:

$$\text{PDTW}(X, Y, N) = \text{DTW}(\text{PAA}(X, N), \text{PAA}(Y, N)). \tag{2.4}$$

The number of segments N that one considers greatly influences the quality of the alignment of the time series. However, PDTW does not give any information on the way to choose it. For making this choice, Chu *et al.* [6] proposes the Iterative Deepening Dynamic Time Warping Algorithm (IDDTW).

2.2. Iterative deepening dynamic time warping

For determining the number of segments, IDDTW only considers values that are power of 2 and for each value, computes an error distribution by comparing PDTW with the standard DTW at each level of compression. It takes as inputs: the query Q, the dataset D, the user's confidence (or tolerance for false dismissals) $user\_conf$, and the set of standard deviations $StdDev$ obtained from the error distribution. Example: Let C and Q be two time series of the dataset D, let $best\_so\_far$ be the DTW distance between two time series of the dataset. Suppose the distance $D_{pdtw}(Q, D)$ is 40 and the $best\_so\_far$ is 30. The difference between the estimated distance and the $best\_so\_far$ is 10. Using the error distribution centred around the approximation (40), we can determine the probability that the candidate could be better by examining the area beyond the location of the $best\_so\_far$ (shown in solid black in Fig. 2): We disqualify a candidate if this probability is less than the user's specified error acceptance, the candidate is disqualified; otherwise, a finer approximation is used and the test is re-applied to the next depth. This process continues until the full DTW is performed.

More precisely, IDDTW proceeds as follows:

- the algorithm starts by applying the classic DTW to the first K candidates from the dataset. The results of the best matches to the query are contained in R, with $|R| = K$. The $best\_so\_far$ is determined from $argmax R$;
- both the query Q and each subsequent candidate C are approximated using PAA representations with N segments to determine the corresponding PDTW;

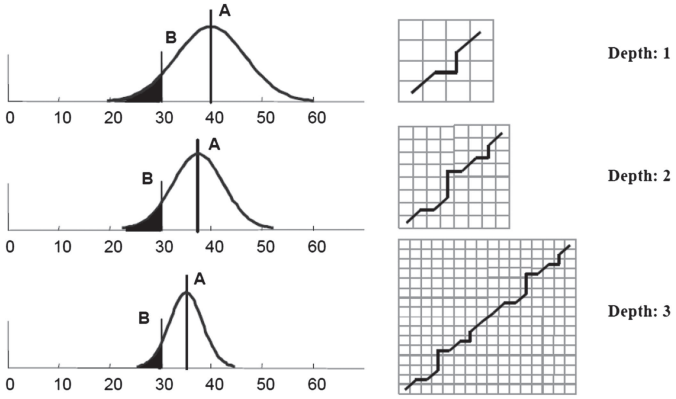


FIGURE 2. IDDTW operating principle. Depth represents approximation levels, A represents approximate distance and B is best_so_far [6].

- a test is performed to determine whether the candidate C can be pruned off or not. If the result of the test is found to have a probability that it could be a better match than the current $best\_so\_far$, a higher resolution of the approximation is required. Then each segment of the candidate is split into two segments to obtain a new candidate;
- the process of approximating Q and C to determine the PDTW should be reapplied and the test is repeated for all approximations levels until they fail the test or their true distance DTW is determined.

In this way, IDDTW finds the number of segments that best approximates DTW and speeds up its computation. However, IDDTW has three main limitations:

- it only considers the numbers of segments for PDTW that are power of 2;
- it requires a user-specified tolerance for false dismissals that influences the quality of the approximation, but the algorithm does not give any indication on how to choose the tolerance;
- it considers DTW as a reference while looking for the number of segments that best aligns the time series. However, because of pathological alignments, DTW sometimes fails to align time series properly [15].

Our goal is to find the number of segments that best aligns the time series and also speeds up the computation of DTW. We propose a heuristic named parameter Free piecewise DTW (FDTW) based on Greedy Randomized Adapted Search Procedure that deals with all the limitations of IDDTW: it considers all the possible values for the number of segments, it is parameter-free and it finds a number of segments for PDTW based on the quality of the time series alignment, namely the error rate for classification task. The next section introduces FDTW.

3. GRASP BASED HEURISTIC

3.1. Evaluation procedures for the compression quality

Before explaining how to evaluate the quality of time series compression, we first describe the time series datasets that we considered. They are made up of time series associated with labels that identify the shape of the latter. For instance, in the ECG dataset, each time series traces the electrical activity recorded during one heartbeat. The two classes are a normal heartbeat and a Myocardial Infarction.

Time series classification is a classic problem with time series which consists in guessing the label of an unlabeled time series based on its shape. The quality of a time series classification model is evaluated from its classification error (ϵ), or its accuracy ($a = 1 - \epsilon$). When considering classification task, one of the best classification algorithm to use for evaluating the quality of time series alignment is *one nearest neighbor (1NN)*.

Indeed, its classification error directly depends on time series alignment, since 1NN has no other parameters [27].

During this work, a compact representation of time series is considered to be good if it reduces the length of the original time series, but also if the classification error obtained by classifying the compact time series is small. The classification error is small when the time series keep their characteristic shape despite compression.

3.2. Problem definition

Let $D = \{d_i\}$ be a set of datasets composed of time series. We note $|d_i|$ the number of time series of the dataset d_i.

Let $X \in d_i$ be a time series of the dataset d_i; we note $|X| = n$ the length of the time series X. For simplicity of notation we suppose that all the time series of d_i have the same length.

Definition 3.1.

$$1NNDTW : D \to [0,1] \tag{3.1}$$
$$d_i \mapsto 1NNDTW(d_i) \tag{3.2}$$

$1NNDTW(d_i)$ is the classification error of one nearest neighbour with Dynamic Time Warping on the dataset d_i.

Definition 3.2.

$$\begin{aligned} &1NNPDTW : D \times \{1 \dots n\} \to [0,1] \\ &(d_i, N) \mapsto 1NNPDTW(d_i, N) \\ &= 1NNDTW \circ PAAset(d_i, N) \end{aligned} \tag{3.3}$$

$1NNPDTW(d_i, N)$ is the classification error of 1-NN with PDTW using N segments on d_i.

Our goal is to find the number of segments that allows PDTW to best align time series. PDTW gives a good alignment when its classification error with 1NN is low [20]. Our problem is then to find the number of segments N that minimizes $1NNPDTW(d_i, N)$.

Formally, given a dataset d_i, whose time series have a length n, we look for the number of segments $N \in \{1 \dots n\}$ such that

$$\min_{1 \le N \le n} \{1NNPDTW(d_i, N)\}. \tag{3.4}$$

3.3. Brute-force search

The simplest way to find the value for the number of segments that minimized the classification error is to test all the possible values. Obviously, this method is time consuming as it requires to test n values to find the best one. The time complexity is :

$$O(|d|^2 \times n^3). \tag{3.5}$$

To reduce the time of the search, the FDTW proposes to look for the number of segments with the minimal classification error without testing all the possible values.

3.4. Greedy randomized adaptive search procedures

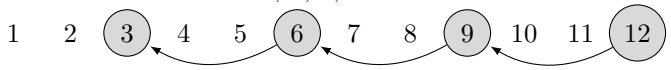
The Greedy Randomized Adaptive Search Procedures (GRASP) is a multi-start, or iterative metaheuristic proposed by Feo and Resende [8], in which each iteration consists of two phases: firstly a new solution is constructed by a greedy randomized procedure and then is improved using a local search procedure.

The greediness criterion establishes that elements with the best quality are added to a restricted candidate list and chosen at random when building up the solution. The candidates obtained by greedy algorithms are not necessarily optimal. So, those candidates are used as initial solutions to be explored by local search. The heuristic we proposed is build upon GRASP and strengthened with an inclusion of specific global search component.

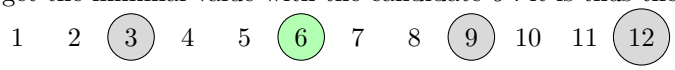
3.5. Parameter free heuristic

The idea of our heuristic is the following:

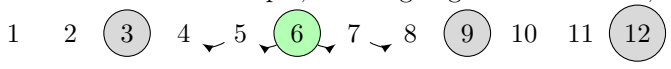
1. We choose M candidates distributed in the space of possible values to ensure that we are going to have small, medium and large values as candidates. The candidates values are: n, $n-\lfloor\frac{n}{M}\rfloor$, $n-2\times\lfloor\frac{n}{M}\rfloor$, ..., $n-M\times\lfloor\frac{n}{M}\rfloor$. For instance, if the length of time series is $n = 12$ and the number of candidates is $M = 4$, we are going to select the candidates 12, 9, 6, 3.



2. We evaluate the classification error with $1NNPDTW$ for each chosen candidate, and we select the candidate that has the minimal classification error: it is the best candidate. In our example, we may suppose that we get the minimal value with the candidate 6 : it is thus the best candidate at this step.



3. We respectively look between the predecessor (*i.e.*, 3 here) and successor (*i.e.*, 9 here) of the best candidate for a number of segments with a lower classification error: this number of segments corresponds to a local minimum. In our example, we are going to test values 4, 5, 7 and 8 to see if there is a local minimum.



4. We restart at step one while choosing different candidates during each iteration to ensure that we return a good local minimum. We fix the number of iterations to $k \leq \lfloor\log(n)\rfloor$. At each iteration, the first candidate is $n - (number\_of\_iteration\ -\ 1)$.

In short, in the worst case, we test the first M candidates to find the best one. Then, we test $\frac{2n}{M}$ other candidates to find the local minimum. We finally perform $nb(M) = M + \frac{2n}{M}$ tests. The number of tests to be performed is a function of the number of candidates. Hence, how many candidates should we consider to reduce the number of tests? The first derivative of nb function vanishes when $M = \sqrt{2n}$ and its second derivative is positive; so the minimal number of tests is obtained when the number of candidates is: $M = \sqrt{2n}$. At each iteration, the heuristic tests $nb(\sqrt{2n}) = \sqrt{8n}$ number of segments. As we have k iterations the number of candidates tested is: $|C| = k\sqrt{8n}$. The details of the heuristic are presented in Algorithm 1.

Time complexity: We use the training set to find the number of segments that should be considered with PDTW. For that purpose, we applied $1NN$ on the training set that costs

$$O(|d|^2 \times n^2\sqrt{n}), \tag{3.6}$$

where $|d|^2$ comes from $1NN$ algorithm and $n^2\sqrt{n}$ comes from $PDTW$.

Lemma 3.3. *For a given dataset d_i,the quality of the alignment of our heuristic is better than that of DTW: $FDTW(d_i) \leq 1NNDTW(d_i)$.*

Proof. 1NNDTW(d_i) = 1NNPDTW(d_i, n). Then, 1NNDTW(d_i) is one of the candidates considered by the heuristic FDTW. Since FDTW returns the minimal classification error from all candidates, the classification error of 1NNDTW is always greater than or equal to FDTW.

Algorithm 1: Parameter Free Dynamic Time Warping

Input: training_set, length of a time serie : n,
number of iterations : nb_rep
Output: The number of segments to be used N
The accuracy associated to N

function FDTW($training\_set, n, nb\_rep$)
  $l \leftarrow floor(n/sqrt(2 * n))$
  $tab\_N \leftarrow ones(n)$
  forall the $i \in \{0, 1, \ldots, (nb\_rep - 1)\}$ **do**
    $tab\_N\_possible\_candidates \leftarrow seq(from = (n - i), to = 1, by = -l)$
    $nb\_candidats \leftarrow |tab\_N\_possible\_candidates|$
    for i *in* $\{1, 2, \ldots, nb\_candidates\}$ **do**
      if $tab\_N[tab\_N\_possible\_candidates[i]] \neq 0$ **then**
        $tab\_N\_candidates[j] \leftarrow tab\_N\_possible\_candidates[i]$
        $tab\_N[tab\_N\_candidates[j]] \leftarrow 0$
        $j \leftarrow j + 1$
    $mat\_r \leftarrow 1NNPDTW(training\_set, tab\_N\_candidats)$
    /* $1NNPDTW$ `return a matrix of couple` $(N, error)$ */
    $min \leftarrow minimun(mat\_r)$
    /* $minimum$ `return the couple` $(N, error)$ `with the minimum error` */
    $result[(i + 1)] \leftarrow localMinimun(min.N, min.error, training\_set, tab\_N)$
  $m \leftarrow minimun(result)$
  return m

Nevertheless, a heuristic does not always give the optimal value. To ensure that it gives a result not far from the optimal value, one approach is to guarantee that the result of the heuristic always lies in an interval with respect to the optimal value [9].

In our case, we are looking for the number of segments that allows a good alignment of time series. The alignment is good when the classification error with 1NN is minimal or when the accuracy is maximal.

Let d_i be a dataset:

$acc_{\max(d_i)} = 1 - \min\limits_{1 \leq N \leq n} \{1NNPDTW(di, N)\}$ is the maximal accuracy for the dataset d_i,

$acc_{DTW} = 1 - 1NNDTW(d_i)$ is the accuracy obtained with d_i and 1NNDTW, and

$acc_{FDTW} = 1 - FDTW(d_i)$ is the accuracy of our heuristic.

To ensure the quality of our heuristic FDTW, we hypothesized that 1NNDTW is better than Zero Rule classifier. Zero Rule classifier is a simple classifier that predicts the majority class of test data (if nominal) or average value (if numeric). Zero Rule is often used as baseline classifier [7]. The minimal value of the accuracy of Zero Rule is $\frac{1}{c}$ where c is the number of classes of the dataset.

Proposition 3.4. *For a given dataset d_i that has c_i classes, $c_i \in \mathbb{N}^*$,*

$$if\ acc_{DTW} \geq \frac{1}{c_i}\ then\ \frac{1}{c_i} \times acc_{max} \leq acc_{FDTW} \leq acc_{max}$$

Proposition 1 shows that when 1NN associated with DTW has a better accuracy than the baseline classifier Zero Rule, the FDTW heuristic is a parametric approximation.

Proof. By definition, $acc_{FDTW} \leq acc_{max}$ We look for $\beta \in \mathbb{N}$ such that

$$\frac{1}{\beta} \times acc_{max} \leq acc_{FDTW} \Leftrightarrow \frac{acc_{max}}{acc_{FDTW}} \leq \beta \tag{3.7}$$

$$\text{However, } \frac{acc_{max}}{acc_{FDTW}} \leq \frac{1}{acc_{FDTW}} \text{ because } acc_{max} \leq 1 \tag{3.8}$$

$$\text{And, } \frac{1}{acc_{FDTW}} \leq \frac{1}{acc_{DTW}} \text{ because } acc_{DTW} \leq acc_{FDTW} \tag{3.9}$$

$$\text{So, } \frac{1}{acc_{DTW}} \leq c_i \text{ because } \frac{1}{c_i} \leq acc_{DTW} \text{ by hypothesis} \tag{3.10}$$

4. Experiment and results

Throughout the experiments described in this paper, FDTW performs three iterations ($k = 3$) when searching for the appropriate number of segments for a dataset. To evaluate the ability of FDTW heuristic to propose a good number of segments for PAA. It has been compared to the IDDTW algorithm in terms of:

- heuristic execution speed;
- time series compression ratio;
- classification error associated with the number of segments found by the heuristic.

4.1. Case studies

PAA is widely used in temporal data mining and often as a primitive by other algorithms such as those allowing to construct a symbolic representation of time series, those allowing to index a time series or even those allowing to classify time series. In this section, we present some algorithms for which the pre-processing performed by FDTW allows to improve the final results.

4.1.1. Datasets

The experiments have been performed first on 45 datasets and then on 84 datasets of UCR time series datamining archive [5], which provided a large collection of datasets that covers various categories of domains. Each data set is divided into a training set and a testing set. The 84 datasets possess between 2 and 60 classes, the length of time series varies from 24 to 2709, the training sets contain between 16 and 8926 time series and the testing sets contain between 20 and 8236 time series. All datasets are publicly available on the UCR time series classification page.

4.1.2. Compression

Compression ratio: An immediate way to evaluate the quality of the segmentation is to compare the compression ratios. A segment number N_1 will be better than a segment number N_2 if it makes it possible to obtain a more compact representation with PAA. The compression ratio is given by:

$$r = \frac{n - N}{n}$$

where n is the length of the time series and N is the number of segments considered with PAA. The closer r is to 1 the better is the compression.

The numbers of segments used here are shown in Table 1. For the considered datasets, the mean compression ratio of *IDDTW ($r = 0.654$)* is slightly higher than that of *FDTW ($r = 0.605$)*. However, this difference is not significant. Indeed, the wilcoxon test gives us a p-value greater than 0.1 ($p > 0.1$). Therefore, we cannot reject the hypothesis that the compression ratios of IDDTW and FDTW are equal.

Applicaion: PAA used with a suitable segment number allows compression of the time series of the *Coffee* dataset without loss of information. Although they are more compact, the obtained time series capture the main variations of the original time series (Fig. 3).

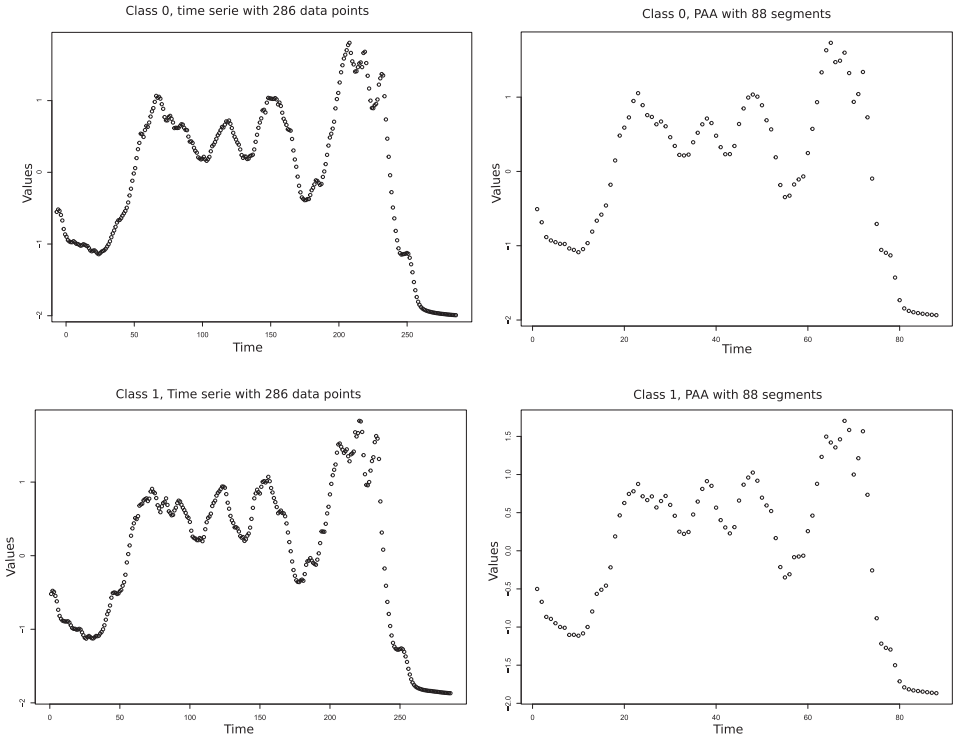


FIGURE 3. Visual comparison of two time series from the two classes of the coffee dataset. *Left*: the original time series, *right*: representation using PAA with 88 segments.

4.1.3. Classification

Piecewise Aggregate Approximation is used by ShapeDTW [30] and DTW_F [12] to classify time series. However, to evaluate the actual impact of the segment number considered on the classification, we tested FDTW to choose the number of segments to use with 1NN and PDTW.

PDTW was designed to speed up the calculation of DTW without degrading the accuracy. Here, we observe that when the number of segments is chosen, this may even lead to an improvement of the results of the classification.

Quality of the number of segments found:
A segment number N_1 is better than a segment number N_2 if the classification error associated with N_1 is smaller than that associated with N_2. So, to evaluate the quality of our heuristic FDTW, we compared its classification errors with that of IDDTW. The classification error was calculated based on the threefold cross validation applied on the training set. IDDTW tested all the values of N that were equal to a power of two and kept the one that had a minimum classification error (Tab. 1).

Application:
According to the announcement in Lemma 3.3, the classification error of FDTW during the learning phase (training error) is less than or equal to that of DTW for all the considered datasets. We used *Wilcoxon signed rank test* with continuity correction to test the significance of FDTW against IDDTW. The Wilcoxon signed rank test gives a p-values, $p < 0.01$, which demonstrates that FDTW achieves a significant reduction of the classification error of IDDTW. This also demonstrates that FDTW allows to find segment numbers for PAA that are of better quality than those found by IDDTW during the learning phase.

TABLE 1. Classification errors associated with the number of segments N chosen by the heuristics IDDTW and FDTW. When two numbers of segments N_1 and N_2 are associated with the same classification error, the smallest is considered. The classification error is calculated based on the threefold cross validation applied on the training set.

No.	Datasets (training set) (training set)	DTW	IDDTW	N	FDTW	N
1	50Words	0.349	0.340	256	0.318	80
2	Adiac	0.462	0.426	128	0.426	140
3	ArrowHead	0.250	0.167	16	0.111	14
4	Beef	0.567	0.900	8	0.567	169
5	Car	0.400	0.233	8	0.217	385
6	CBF	0.000	0.000	128	0.000	22
7	Coffee	0.033	0.133	64	0.000	88
8	Cricket_X	0.210	0.244	256	0.190	84
9	Cricket_Y	0.279	0.285	256	0.272	214
10	Cricket_Z	0.267	0.272	256	0.249	250
11	DistalPhalanxOutlineAgeGroup	0.570	0.541	16	0.534	14
12	DistalPhalanxTW	0.375	0.339	16	0.317	40
13	Earthquakes	0.266	0.266	512	0.223	101
14	ECG	0.240	0.170	8	0.170	11
15	ECG Five Days	0.387	0.220	32	0.220	7
16	Face (all)	0.875	0.873	128	0.870	50
17	Face (four)	0.208	0.125	32	0.083	140
18	Fish	0.343	0.314	16	0.303	27
19	Gun-point	0.201	0.039	32	0.020	38
20	Ham	0.650	0.512	32	0.512	32
21	Haptics	0.587	0.536	64	0.516	239
22	InlineSkate	0.519	0.519	64	0.499	48
23	ItalyPower Demand	0.045	0.060	8	0.045	20
24	Lightning-2	0.183	0.150	16	0.100	179
25	Lightning-7	0.315	0.344	64	0.200	155
26	Medical Images	0.286	0.307	64	0.278	94
27	MiddlePhalanxTW	0.429	0.442	32	0.429	80
28	MoteStrain	0.246	0.246	16	0.190	46
29	OliveOil	0.367	0.367	32	0.333	423
30	OSU leaf	0.310	0.335	32	0.270	33
31	Plane	0.000	0.000	32	0.000	32
32	ProximalPhalanxTW	0.317	0.283	4	0.283	4
33	ShapeletSim	0.786	0.246	8	0.143	45
34	SonyAIBORobot Surface	0.198	0.095	16	0.048	22
35	SonyAIBORobot Surface II	0.148	0.111	64	0.037	42
36	Swedish	0.250	0.238	64	0.218	59
37	Symbols	0.037	0.037	32	0.000	34
38	Synthetic Control	0.350	0.410	32	0.350	60
39	Trace	0.000	0.000	64	0.000	108
40	Two patterns	0.000	0.000	32	0.000	32
41	Two Lead ECG	0.125	0.083	64	0.083	52
42	Wafer	0.014	0.012	8	0.008	111
43	Wine	0.684	0.632	128	0.632	20
44	Words Synonyms	0.419	0.423	64	0.382	57
45	Yoga	0.233	0.187	128	0.187	356

Comparison with IDDTW :
To evaluate the quality of FDTW, we compared its classification errors with that of IDDTW and the minimal one. The minimal classification error was find by applying Brute-force search (BF) on both training set and testing set. FDTW and IDDTW used the training set to find the segment number N with minimal training error using threefold cross validation, and then used this number of segments on the testing set to compute the classification error. The value of the segment number N found on the training set may in some cases not be appropriate for the testing set. We speak of a generalization error which is due to the representativeness of the training set (Tab. 2).

If two numbers of segments N_1 and N_2 are associated with the same training error, we retain the largest. IDDTW tested all the values of N that were equal to a power of two during the learning phase and kept the one that had a minimum classification error.

The experiments showed that FDTW is more performant than IDDTW. Actually, FDTW resulted in a lower generalization error than IDDTW on 22 datasets and the same generalization error than IDDTW on eight datasets. The Wilcoxon signed rank test gives a p-values, $0.01 < p \leq 0.05$, which demonstrates that FDTW achieved a significant reduction of the generalization error of IDDTW. Results also show that FDTW managed to find the minimum error for nine datasets (Coffee, ECGFiveDays, Gun-point, ItalyPowerDemand, OliveOil, Plane, Synthetic control, Trace, Two patterns) and outperforms the smallest classification error reported in the literature on dataset CBF (No. 5).

Heuristic execution speed:
As already suggested by the time complexity of FDTW and IDDTW heuristics, IDDTW tests fewer candidates than FDTW and is therefore faster. However, the number of candidates tested by FDTW reduces exponentially with the length of the time series (Fig. 4). Actually, the number of candidates to be tested ranges from 1 to n, n being the length of time series, and FDTW considers $\sqrt{n}$ candidates for each iteration.

In average, FDTW is 8 times faster than Brute-force search with an average execution time of 176 minutes against 1386 min for Brute-force search. IDDTW is seven times faster than FDTW and remains the fastest with an average execution time of 24 min. The execution time increases with the length of the time series (Fig. 5). The increase of Brute-force search execution time is faster than that of FDTW and IDDTW. This is observable from the datasets Lightning-2 whose time series have a length equal to 637 data points. Note: The experiments were conducted on a PC with an Intel Core i7 processor, 16GB of RAM and a Windows 7 64-bit operating system.

Comparison with other classification algorithms:
To evaluate the quality of FDTW, we compared its classification errors (generalization error) with that of 35 other classification algorithms [2] of the literature on 84 datasets of UCR archive. The performances of the algorithms are compared using the Nemenyi test that compares all the algorithms pairwise and provides an intuitive way to visualize the results (Fig. 6). The Nemenyi test allows ranking classification algorithms according to their average accuracy on 84 datasets. FDTW obtained good results on the simulated datasets in terms of average accuracy (3rd/37 algorithms, Fig. 6) because the data of the training set and the testing set are generated by the same models.

However, to evaluate the significance of the difference between the classification algorithms on 84 datasets, we used the Wilcoxon signed rank test with continuity correction, which has more statistical power.The results of these experiments show that despite data compression,

- FDTW have better performance than Naive Bayes (NB), C45, logistic regression (Logistic), BN;
- FDTW has similar performance to that of 26 other algorithms in the literature, namely : SVMQ, RANDF, ROTF, MLP, EUCLIDEAN_1_NN, DDTW_R1_1NN, DDTW_RN_1NN, ERP_1NN, LCSS_1NN, MSM_1NN, TWE_1NN, WDDTW_1NN, WDTW_1NN, DD_DTW, DTD_C, LS, BOP, SAXVSM, TSF, TSBF, LPS, PS, CID_DTW, SVML, FS, ACF;
- Only five algorithms DTW_F, Shapelet Transform (ST), BOSS, Elastic Ensemble (EE) and COTE perform better overall than FDTW.

TABLE 2. Comparison of generalization errors. In **italics**, the smallest generalization error. In **bold**, the smallest generalization error between IDDTW and FDTW. N is the number of segments selected and ℓ is the number of data points in a segment ($l = \lfloor \frac{n}{N} \rfloor$). The generalization error is computed on the testing set.

	Results reported in [1,5]				Our experiments				
No.	1NN Euclidean distance	1NN DTW	1-NN DTW (r)	Brute force search	$N(\ell)$	IDDTW	$N(\ell)$	FDTW	N(ℓ)
1	0.369	0.310	0.242 (6)	0.262	251(1)	**0.268**	256(1)	**0.268**	258(1)
2	0.389	0.396	0.391 (3)	0.379	162(1)	0.432	128(1)	**0.414**	143(1)
3	0.333	0.367	0.333 (0)	0.233	286(2)	**0.3**	8(59)	0.367	94(5)
4	0.425	0.274	0.288 (5)	*0.192*	150(2)	**0.301**	64(5)	**0.301**	170(2)
5	0.26	0.25	0.253 (5)	0.233	27(2)	**0.283**	2(40)	**0.283**	385(2)
6	0.148	0.003	0.004 (11)	*0*	118(1)	0.003	128(1)	***0.001***	128(1)
7	0.000	0.000	0.000 (0)	*0*	13(22)	***0***	64(4)	***0.000***	286(1)
8	0.423	0.246	0.228 (10)	0.228	142(2)	**0.256**	256(1)	0.269	84(4)
9	0.433	0.256	0.238 (17)	0.231	271(1)	**0.241**	256(1)	0.244	294(1)
10	0.413	0.246	0.254 (5)	0.221	249(1)	**0.223**	256(1)	0.233	276(1)
11	0.218	0.208	0.228 (1)	0.2	78(1)	0.225	16(5)	**0.223**	80(1)
12	0.273	0.29	0.272 (0)	*0.263*	35(2)	0.288	16(5)	**0.278**	80(1)
13	0.326	0.258	0.258 (22)	*0.198*	176(2)	**0.258**	512(1)	0.276	101(5)
14	0.120	0.230	0.120 (0)	0.13	38(3)	0.19	8(12)	**0.180**	11(9)
15	0.203	0.232	0.203 (0)	0.117	11(12)	0.289	32(4)	**0.117**	11(12)
16	0.286	0.192	0.192 (3)	0.091	79(2)	0.194	128(1)	**0.148**	99(1)
17	0.216	0.170	0.114 (2)	0.08	107(3)	0.352	32(11)	**0.102**	140(3)
18	0.217	0.177	0.154(4)	0.154	149(3)	0.257	16(29)	**0.177**	27(17)
19	0.087	0.093	0.087 (0)	0.02	38(4)	0.073	32(5)	**0.020**	38(4)
20	0.4	0.533	0.400 (0)	0.343	21(20)	**0.026**	32(13)	0.432	32(13)
21	0.630	0.623	0.588 (2)	0.549	328(3)	**0.588**	64(17)	0.594	948(1)
22	0.658	0.616	0.613 (14)	0.578	1770(1)	0.627	64(29)	**0.622**	171(11)
23	0.045	0.050	0.045 (0)	0.033	20(1)	0.043	8(3)	**0.033**	24(1)
24	0.133	0.167	0.133 (0)	0.1	191(3)	0.167	32(18)	**0.100**	234(2)
25	0.267	0.267	0.233 (1)	0.183	52(11)	**0.367**	8(72)	**0.367**	377(1)
26	0.038	0	0.000 (6)	*0*	35(4)	***0***	128(1)	***0***	135(1)
27	0.439	0.416	0.419 (2)	*0.398*	27(2)	**0.414**	32(2)	0.416	80(1)
28	0.121	0.165	0.134 (1)	0.135	14(6)	0.197	16(5)	**0.165**	84(31)
29	0.246	0.131	0.131 (6)	*0.082*	70(9)	0.246	16(40)	**0.180**	524(1)
30	0.479	0.409	0.388 (7)	0.364	31(14)	**0.372**	32(13)	0.409	35(12)
31	0.316	0.263	0.253 (20)	0.255	95(1)	**0.271**	64(2)	0.280	34(3)
32	0.292	0.263	0.263 (6)	0.24	75(1)	**0.288**	4(20)	**0.288**	4(20)
33	0.461	0.35	0.300 (3)	0.083	54(9)	0.239	64(7)	**0.122**	48(10)
34	0.305	0.275	0.305 (0)	0.206	37(2)	**0.208**	16(4)	0.304	26(3)
35	0.141	0.169	0.141 (0)	0.14	5(13)	0.197	16(4)	**0.178**	45(1)
36	0.211	0.208	0.154 (2)	0.165	59(2)	**0.195**	64(2)	0.208	55(2)
37	0.100	0.050	0.062 (8)	0.044	376(1)	**0.059**	32(12)	0.060	34(12)
38	0.120	0.007	0.017 (6)	0.007	60(1)	0.437	2(30)	**0.007**	60(1)
39	0.240	0.000	0.010 (3)	*0*	47(6)	***0***	64(4)	***0***	275(1)
40	0.090	0.000	0.002 (4)	*0*	21(6)	***0***	64(2)	***0***	128(1)
41	0.253	0.096	0.132 (5)	0.045	55(1)	0.073	32(3)	0.112	70(1)
42	0.005	0.020	0.005 (1)	0.007	109(1)	0.013	8(19)	**0.008**	95(2)
43	0.389	0.426	0.389 (0)	0.204	3(78)	0.463	20(11)	**0.37**	128(1)
44	0.382	0.351	0.252 (8)	0.337	133(2)	0.365	64(4)	**0.343**	135(2)
45	0.170	0.164	0.155 (2)	0.149	117(4)	0.158	128(3)	**0.154**	384(1)
$\overline{X}$	0.268	0.227	0.242	0.175		0.232		0.214	

Notes. DTW(r) is a constraint version of DTW where the number of consecutive data points that can be compared to a single point during the warping is bounded. r represents the size of the warping windows.

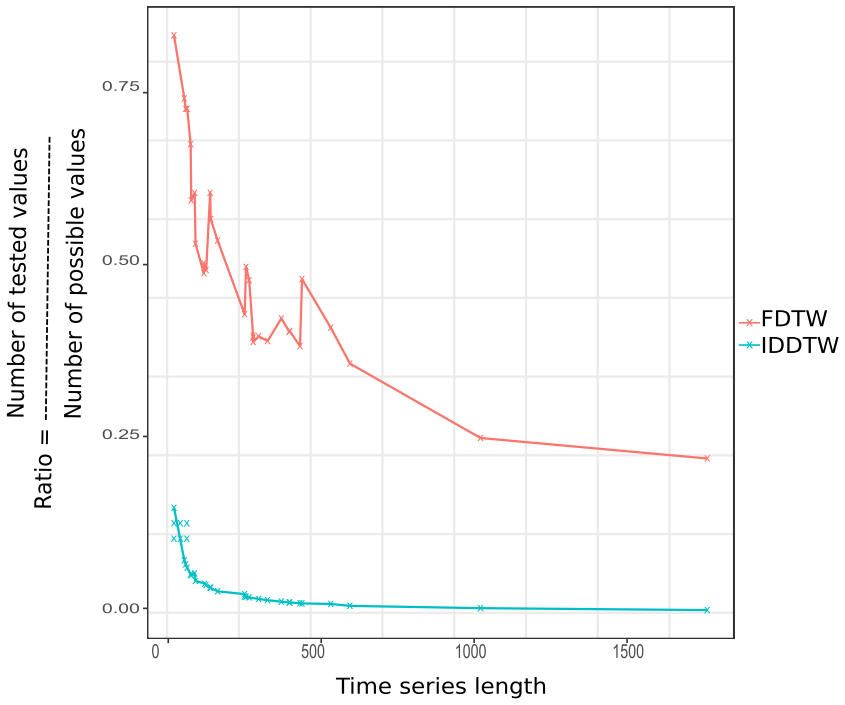


FIGURE 4. Comparison of the number of tested values of the parameter number of segments with the FDTW and IDDTW. x-axis datasets are sorted according to the length of the time series.

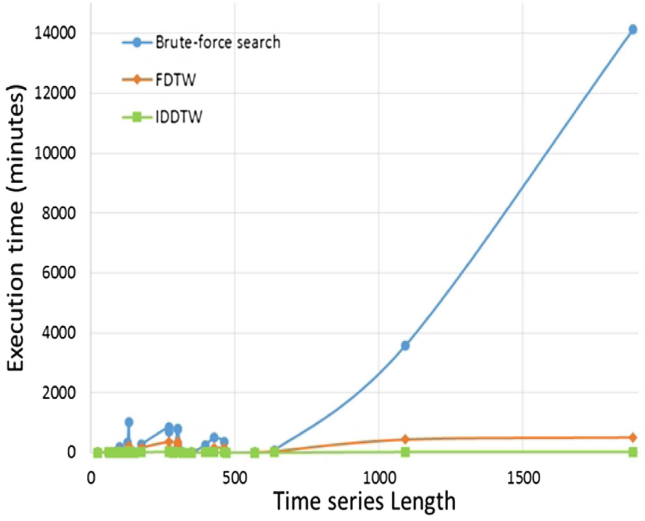


FIGURE 5. Comparison of the execution time of the Brute-force search algorithm, FDTW and IDDTW.

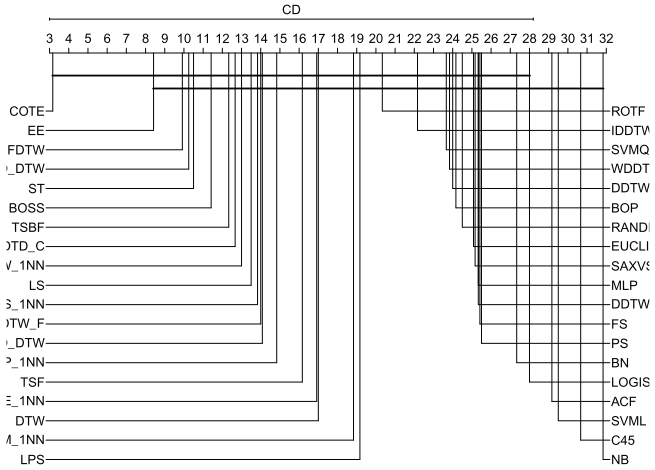


FIGURE 6. Critical difference diagram for FDTW and 36 other classification algorithms on six simulated datasets. FDTW is ranked 3rd/37 algorithms.

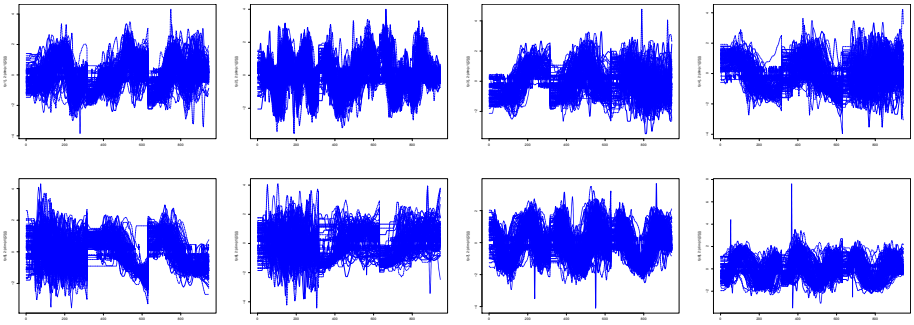


FIGURE 7. Eight types of time series corresponding to the vocabulary of eight gestures.

These results demonstrate the competitiveness of FDTW. Moreover, this algorithm outperforms the best result reported in the literature on UWaveGestureLibraryAll dataset (Fig. 7). The challenge with this dataset is to recognize the gesture made by a user from measurements made by accelerometers. As reported in [1] the best accuracy obtained on this dataset is 83.44% with TSBF algorithm; FDTW outperforms this result and allows to obtain *91.87%* of accuracy.

Additional experiments are available here [24].

5. Conclusion and perspective

This paper deals with the problem of choosing an appropriate number of segments to compress time series with PAA in order to improve the alignment with DTW. In this aim, we proposed a parameter Free heuristic named FDTW, which approximates the optimal number of segments to use. The experiments showed that FDTW increased the quality of alignment of time series especially on synthetic datasets where DTW associated with PAA performed better than any other variant of DTW on a classification task and was rank 3rd/37 behind two ensemble classification algorithms COTE and EE. This algorithm allows reducing the storage space and the processing time of time series while increasing the quality of the alignment of DTW. As a perspective, the

problem we have dealt with in this paper could be modeled as a multi-objective optimization problem where one objective function would be compression and the other the classification of time series.

Acknowledgements. We thank the French Ministry of Higher Education, Research, and Innovation, for funding this work. This work was also partly supported by the Labex project IMobS3. We finally thank the reviewers who allowed us to improve the quality of this work through their comments, and suggestions.

References

[1] A. Bagnall, E. Keogh, J. Lines, A. Bostrom, J. Large, Time Series Classification Website. Available at: http://timeseriesclassification.com (2016).

[2] A. Bagnall, J. Lines, A. Bostrom, J. Large, E. Keogh, The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Min. Knowl. Discov.* **31** (2017) 606–660.

[3] A. Camerra, T. Palpanas, J. Shieh, E. Keogh, isax 2.0: Indexing and mining one billion time series. In: 2010 IEEE 10th International Conference on Data Mining – ICDM (2010) 58–67.

[4] K.S. Candan, R. Rossini, X. Wang, M.L. Sapino, sdtw: computing dtw distances using locally relevant constraints based on salient feature alignments. *VLDB Endowment* **5** (2012) 1519–1530.

[5] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, The UCR time series classification archive. Available at: http://www.cs.ucr.edu/~eamonn/time_series_data/ (2015).

[6] S. Chu, E.J. Keogh, D.M. Hart, M.J. Pazzani, *et al.*, Iterative deepening dynamic time warping for time series. In: *Proc. of the 2002 SIAM International Conference on Data Mining*. SIAM (2002) 195–212.

[7] J. Cuřín, P. Fleury, J. Kleindienst, R. Kessl, Meeting state recognition from visual and aural labels. In: *International Workshop on Machine Learning for Multimodal Interaction*. Springer (2007) 24–25.

[8] T.A. Feo, M.G. Resende, Greedy randomized adaptive search procedures. *J. Glob. Optim.* **6** (1995) 109–133.

[9] O.H. Ibarra, C.E. Kim, Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM (JACM)* **22** (1975) 463–468.

[10] Itakura, F., Minimum prediction residual principle applied to speech recognition. *IEEE Trans. Acoust. Speech Signal Process.* **23** (1975) 67–72.

[11] Y.S. Jeong, M.K. Jeong, O.A. Omitaomu, Weighted dynamic time warping for time series classification. *Pattern Recogn.* **44** (2011) 2231–2240.

[12] R.J. Kate, Using dynamic time warping distances as features for improved time series classification. *Data Min. Knowl. Discov.* **30** (2016) 283–312.

[13] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, Dimensionality reduction for fast similarity search in large time series databases. *Knowl. Inform. Syst.* **3** (2001) 263–286.

[14] E.J. Keogh, M.J. Pazzani, Scaling up dynamic time warping for datamining applications. In: Sixth ACM SIGKDD. ACM (2000) 285–289.

[15] E.J. Keogh, M.J. Pazzani, Derivative dynamic time warping. In: *1st SIAM International Conference on Data Mining*. SIAM (2001) 1–11.

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

[17] B. Lkhagva, Y. Suzuki, K. Kawagoe, Extended SAX: Extension of Symbolic aggregate approximation for financial time series data representation. DEWS2006 4A-i8 **7** (2006).

[18] J. Longin, M. Vasilis, W. Qiang, L. Rolf, A. Chotirat, E. Keogh, Elastic partial matching of time series. In: *9th European Conference on Principles and Practice of Knowledge Discovery in Databases, Porto, Portugal* (2005).

[19] C. Myers, L. Rabiner, A. Rosenberg, Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Trans. Acoust. Speech Signal Process.* **28** (1980) 623–635.

[20] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, E. Keogh, Searching and mining trillions of time series subsequences under dynamic time warping. In: *18th ACM SIGKDD* (2012) 262–270.

[21] C.A. Ratanamahatana, E. Keogh, Making time-series classification more accurate using learned constraints. In: *Proc. of the 2004 SIAM International Conference on Data Mining*. SIAM (2004) 11–22.

[22] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoust. Speech Signal Process.* **26** (1978) 43–49.

[23] S. Salvador, P. Chan, Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal.* **11** (2007) 561–580.

[24] V.S. Siyou Fotso, E. Mephu Nguifo, P. Vaslin, Comparaison des Algorithmes de classification FDTW. Available at: http://fc.isima.fr/~siyou/fdtw (2016).

[25] Y. Sun, J. Li, J. Liu, B. Sun, C. Chow, An improvement of Symbolic aggregate approximation distance measure for time series. *Neurocomputing* **138** (2014) 189–198.

[26] L. Ulanova, N. Begum, E. Keogh, Scalable clustering of time series with u-shapelets. In: *2015 SIAM International Conference on Data Mining*. SIAM (2015) 900–908.

[27] 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** (2013) 275–309.
[28] D. Yu, X. Yu, Q. Hu, J. Liu, A. Wu, Dynamic time warping constraint learning for large margin nearest neighbor classification. *Inform. Sci.* **181** (2011) 2787–2796.
[29] Z. Zhang, P. Tang, R. Duan, Dynamic time warping under pointwise shape context. *Inform. Sci.* **315** (2015) 88–101.
[30] J. Zhao, L. Itti, Shapedtw: shape dynamic time warping. Preprint arXiv:1606.01601 (2016).