Adaptive Multiresolution and Dedicated Elastic Matching in Linear Time Complexity for Time Series Data Mining

Pierre-François Marteau and Gildas Ménier
VALORIA Université de Bretagne Sud, BP 573, 56017 Vannes, FRANCE
{pierre-francois.marteau, gildas.menier}@univ-ubs.fr

Abstract

We develop an adaptive multiresolution approach to the problem of multidimensional time series characterization. Furthermore we provide a dedicated elastic pseudo distance to support similarity search mechanisms for such characterization. We show theoretically and experimentally that our multiresolution decomposition of times series has a linear complexity in time and space. The pseudo elastic distance AMR-DTW that we develop to match multiresolution representations of time series is also based on iterative algorithms that show linear time and space complexity for some tuned parameters. We evaluate the proposed adaptive multiresolution algorithm and associated pseudo elastic distance in a classification experiments to demonstrate the efficiency and accuracy of the proposed representation and matching scheme for time series data mining.

1. Introduction

More and more applications require searching and extracting similar time series. Among numerous examples, we find financial data analysis [2], the identification of moving objects [28], the search for similar musical excerpt [6], etc. Research studies in this area address mainly two strongly interconnected questions: i) the choice of a representation space in which time series are embedded and the choice of a metrics to compare time series within this space (either a distance, a pseudo distance or a similarity measure) appropriate to the application domain; ii) the scalability of the search algorithms. A wide range of metrics have been proposed throughout the literature, in particular the *Lp* norms [2][8], dynamic time warping (*DTW*) [22] [27] [12] [10], the Longest Common Sub Sequence (*LCSS*) [4] and the edit distance (Symbolic Edit Distance *SED*, Edit Distance on Real Data *EDR*) [16][6]. The previous metrics are far to be equivalent : *Lp* norms are efficiently computed (in *O(N)* complexity) but their main drawbacks, at least for a wide range of applications (such as speech recognition, stock analysis, etc.) is their high sensitivity to local temporal compression, or dilatation. In the contrary, *DTW*, *LCSS* or *EDR* pseudo elastic distances have been precisely proposed to 'resist' to local time warp transformations, but suffer from a higher computational cost since they show to have a quadratic complexity. Several ways are considered in the literature to address the scalability of similarity search in large time series data bases. A first way considers reducing the dimension of the representation space in which time series are embedded. The search mechanisms are then exploited in the reduced dimensionality space. This general idea has been introduced in [2] with the use of the first Discrete Fourier Transform (*DFT*) coefficients to represent time series excerpts. Other techniques have been experimented, such as Singular Value Decomposition, (*SVD*) [11], Discrete Wavelet Transform, *DWT* [4]), Adaptive Piecewise Constant Approximation (APCA) [11], etc. A second way consists in reducing the complexity of the search mechanism. A tight lower-bounding measure for dynamic time warping (*DTW*) distances for univariate time series was introduced for instance in [12][27], [10] to avoid comparing the search template to a time series with *DTW* when the lower-bounding estimate (calculated in *O(N)*) indicates that the time series is a worse match than the current best match or outside a search neighbourhood. [24] proposed *FastDTW* algorithm based upon a multilevel approach that recursively projects a solution from a coarse resolution and refines the projected solution. They prove the linear time and space complexity of *FastDTW* both theoretically and empirically. In this context we tackle scalability of elastic matching through adaptive multiresolution polygonal approximation and matching of time series at the price of a linear time and space complexity. In the first part of this paper, we develop the adaptive multiresolution approach to the problem of approximating multidimensional time series. In the second part we detail a matching paradigm dedicated to the proposed adaptive multiresolution representation of time series. Finally, we use supervised classification paradigms on synthetic and real data to evaluate the efficiency and accuracy of our algorithms.

2. Adaptive Multiresolution simplification of times series using polygonal curves approximation

Approximation of multi dimensional discrete time series has been widely studied essentially to speed up data processing required by more and more resource demanding applications such as Computer Vision, Computer Graphics, Data Compression, etc. For polygonal approximation of discrete time series, the problem can be informally stated as follows: given a digitized curve X of $N \geq 2$ ordered samples, find k dominant samples among them that define a sequence of segments which most closely approximate the original curve. This problem is known as the *min-*ε *problem*. Numerous algorithms have been proposed for more than thirty years to solve efficiently this optimisation problem. Most of them belong either to graph-theoretic, dynamic programming or to heuristic approaches. In this section, we derive a new partially optimal algorithm that belongs to the dynamic programming category to solve the *min-*ε problem. This algorithm is essentially inspired from [18], [14]. It provides a polygonal multi resolution approximation of any discrete time series in linear time and space.

<u>*Notation:*</u>

- *X(m)*: a discrete time series
- *mr(X)*: the multiresolution of *X*
- *mr(X,i)*: the polygonal approximation of *X* for the resolution level *i*.
- *p*: the dimension of the space that embeds *X*; $\forall m, X(m) \in R^p$
- *N*: the number of samples or length of the multivariate time series
- k_N: number of segments of a polygonal approximation
- ρ_N: the ratio k_N/N : $\rho_N \in]0;1[$
- $ro_N = (1- \rho_N)$
- α_N: the corridor factor
- *Cradius*: the corridor radius, a parameter used to reduced the search space of *PyCA* algorithm: $Cradius = \alpha_N/\rho_N = \alpha_N . N/k_N$
- *cr:* the compression rate for a polygonal approximation
- *Lb(i) = CRadius –i*; Lower bound used to limit the search space of the *PyCA* algorithm
- $C_{inf}(j)$: Corridor lower bound for the j^{th} segment
- $C_{sup}(j)$: Corridor upper bound for the j^{th} segment
- C_N^{MR} : Complexity of the *PyCA* algorithm
- C_N : Complexity of the *AMR-PyCA* algorithm
- *R*: number of iterations for the multi resolution, equivalently the number of resolution levels.

We consider time series as a multivariate process $X(t)=[x_1(t), x_2(t),\dots, x_p(t)]$ where *X(t)* is a time-stamped spatial vector in R^p. In practice, we will deal with a discrete sampled time series *X(m)* where *m* is the time-stamp index ($m \in \{1,\dots,N\}$). Adopting a data modelling approach to handle the adaptive approximation of the time series, we are basically trying to find an approximation $X_{\hat{\theta}}$ of *X(m)* such as:

$\hat{\theta} = \underset{\theta}{ArgMin}\left(E(X, X_\theta)\right)$, where E is the *RMS* error between X and the model X_θ. In the case of polygonal curve approximation, we select the family $\{X_\theta(m)\}_{m\in\{1,\dots,N\}}$ as the set of piecewise linear and continuous functions (successive segments have to be contiguous, so that the end of a segment is the beginning of the next one). Numerous methods have been proposed for the problem of approximating multidimensional curves using piecewise linear simplification and dynamic programming in $O(k_N.N^2)$ time complexity (Perez et al. 1994). Some efficient algorithms [1] with complexity *O(Nlog(N))* have been proposed for planar curves, but none for the general case in R^d. Here, we have constrained the search of the segments by imposing that the extremities of the piecewise linear segments are vertices of time series *X(t).* Thus, θ is nothing but the set of discrete time location $\{m_i\}$ of the segments' endpoints. Since the end of a segment is the beginning of the following one, two successive segments share a common m_i at their interface. The selection of the optimal set of parameters $\hat{\theta} = \{\hat{m}_i\}$ is performed using a dynamic programming algorithm (Bellman 1957) as follows: we first define the compression rate of the piecewise approximation as:

$$cr = 1 - \frac{|\{m_i\}|}{|\{X(m)\}|} . \frac{p+1}{p} = 1 - \rho_n . \frac{p+1}{p} \quad (1)$$

where $|A|$ stands for the cardinal of set A and $X(m) \in \Re^p, \forall m$

Given a value for *cr* and the size of the trajectory window to sample $N = \left|\{X(m)\}_{n\in\{1,\dots,n\}}\right|$, the number $k = \left|\{m_i\}\right| -1$ of piecewise linear segments is known. Let us define $\theta(j)$ as the parameters of a piecewise approximation containing *j* segments, and $\delta(j,i)$ as the minimal error between the best piecewise linear approximation containing *j* segments and covering the discrete time window $\{1,..,i\}$:

$$\delta(j,i) = \underset{\theta(j)}{Min}\left\{\sum_{m=1}^{i} \left\| X_{\theta(j)}(m) - X(m)\right\|^2\right\} \quad (1)$$

According to the Bellman optimality principle (Bellman, 1957), $\delta(j,i)$ can be decomposed as follows:

$$\delta(j,i) = \underset{m \le i}{Min}\{d(m,i) + \delta(j-1,m)\} \tag{2}$$

$$\text{where} \quad d(m,i) = \sum_{l=m}^{i} \left\| \tilde{X}_{m,i}(l) - X(l) \right\|^2$$

$$\text{and} \quad \tilde{X}_{m,i}(l) = (X(i) - X(m)).\frac{l-m}{i-m} + X(m)$$

is the linear segment between *X(i)* and *X(m)*. The initialization of the recursion is obtained given that *δ(1,1)=0*. The end of the recursion gives the optimal piecewise linear approximation, e.g. the set of discrete time locations of the extremity of the linear segments:

$$\hat{\theta}(k_N) = \underset{\theta(k_N)}{ArgMin}\left\{ \sum_{m=1}^{N} \left\| X_{\theta(k_N)} - X(m) \right\|^2 \right\}$$

with the minimal error: $\delta(k_N,n) = \sum_{m=1}^{N} \left\| X_{\hat{\theta}(k_N)}(m) - X(m) \right\|^2$

It is easy to show that the complexity of the previous algorithm that implements a "Full Search" (FS) is in $O(k_N.N^2)$, which prevents the use of such an algorithm for large *N*. In the scope of dynamic programming search algorithm the only way to reduce time complexity is to reduce the search space itself. Sakoe and Shiba [22] have managed to reduce the complexity of the Dynamic time Warping algorithm down to *O(N)* while defining fixed constraints that define a 'corridor' inside the search space. Following Sakoe and Shiba's work, we developed a dynamic programming solution (*PyCA)* that implements a fixed size 'corridor' [17]: the search space is thus reduced using two fixed constraints. The first one limits the search of the j^{th} segment upper extremity *i* around the mean value $j.N/k_N$ namely the limit of the j^{th} segment as to be chosen inside the interval:

$$[c_{\inf}(j); c_{\sup}(j)[, \text{where:}$$
$$c_{\inf}(j) = Max\{1, j.N/k_N - Cradius\}; \tag{3}$$
$$c_{\sup}(j) = Min\{N, j.N/k_N + Cradius\}$$

The second constraint limits the search of the j^{th} segment lower extremity *m* in the interval $[Max\{1, Lb(i); i\}[$ where *lb(i)=i-Cradius*. Thus, the first constraint defines search bounds for the upper extremity of the j^{th} segment (the *i* index) while the second constraint defines a search bound for the lower limit of the j^{th} segment $m \in [\max(1, i - Cradius); i[$, where *CRadius* is fixed by the user through parameter α_N. The recursive equations for the *PyCA* algorithm are:

$$\delta(j,i) = \underset{lb(i) \le m \le i}{Min}\{d(m,j) + \delta(j-1,m)\}$$
$$\text{with } i \in [c_{\inf}(j); c_{\sup}(j)[, \tag{4}$$
$$c_{\inf}(j) = Max\{1, j.N/k_N - Cradius\};$$
$$c_{\sup}(j) = Min\{N, j.N/k_N + Cradius\}$$

The initialization of the recursion is still obtained observing that *δ(1,1)=0* and the end of the recursion gives the **sub-optimal** (it is optimal on the constrained search space) piecewise linear approximation, e.g. the set of discrete time locations of the extremity of the linear segments:

$$\delta(k_N,i) = \underset{lb(i) \le m \le i}{Min}\{d(m,k_N) + \delta(k_N - 1, j)\} \tag{7}$$
$$\text{where } i \in [Max\{1, N - Cradius\}; N[$$

For *CRadius* = $2.N/k_N$, this leads to an $O(N^2/k_N)$ complexity.

3. Adaptive Multi Resolution approach to Polygonal Curve Approximation (MR-PyCA algorithm)

Basically, the idea behind the adaptive multiresolution approach to polygonal curve simplification is to successively approximate previous approximations obtained by using some given simplification algorithm, this process being initiated from an original discrete time series. Following Kolesnikov's notation [14], we take a sequence of polygonal curves $\{X_1, X_2, \ldots, X_R\}$ as a multiresolution (multiscale) approximation of a *N*-vertex input curve *X*, if the set of curves X_r satisfies the following conditions:

i) A polygonal curve X_r is an approximation of the curve *X* for a given number of segments k_r (*min-ε problem*) or error tolerance ε_r (*min-# problem*), where *r* is the number of resolution level (*r*=1,2,…, *R*).

ii) The set of vertices of curve X_r for resolution level *r* is a subset of vertices of curve X_{r-1} for the previous (higher) resolution level (*r*-1). The lowest resolution level *r*=*R* is represented by the most coarse approximation of *X*. The highest resolution level *r*=1 is represented by the most detailed approximation with the largest number of segments k_r ($k_0 > k_1 > \ldots > k_R$) or smallest error tolerance ε_R for some distance measure (e.g.L_∞) ($\varepsilon_0 < \varepsilon_1 < \ldots < \varepsilon_R$). Thus, an approximation curve X_r is either obtained by inserting new points into the approximation curve X_{r+1}, or, conversely, X_r is obtained by deleting points from the approximation curve X_{r-1}. These two approaches have led to the development of two very popular heuristic approaches: *Split* and *Merge* methods respectively. A famous split method is the Douglas-Peucker algorithm [7]; it has been used for multiresolution approximation in [15]. In the Merge approach [19], [26], the approximation is performed by using a cost function that allows sequential elimination of the vertices with the smallest cost value, and the two adjacent segments are merged into one segment.

Our algorithm relates to the Merge approach: we initiate the simplification process from the finest resolution level and iterate to obtain the crudest, while discarding some vertices during each iteration using the *PyCA* algorithm. The parameters of this algorithm are the corridor factor α (*alpha*), the *ro* factor (*ro=1-ρ*) between two successive resolution levels. If *K,* the number of segments of the crudest approximation, is an input, the number of resolution levels *R* (the number of iterations) is calculated given *K* and *ro*. In that case we chose *r* such that: $N.\rho^{R+1} < K \leq N.\rho^{R}$. As potentially $K < N.\rho^{R}$, a residual iteration is required to simplify the R^{th} approximation (corresponding to resolution level *R*) that has $N.\rho^{R}$ segments to an approximation having exactly *K* segments.

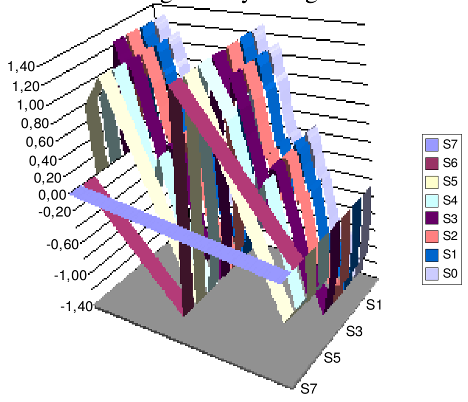

FIG. 1 – Adaptive multiresolution of a signal composed with 7 sinus functions, with *ρ=.25*.

The adaptive multiresolution is the sequence of nested approximations provided as outputs. By construction this algorithm maintains partial optimality between two successive resolution levels, since a constrained dynamic algorithm is used to search inside a fixed size 'corridor' for which segment extremities should be discarded and which should be kept. An example of adaptive multiresolution is given in Fig.1 for the signal: $f(t)=\sum_{k=0}^{6}(1/2)^k \sin(2\pi t/2^{7-k})$.

Complexity of PyCA

According to the previous notations, the complexity of the *PyCA* algorithm evaluates to:

$$\mathcal{C}_N = 2.Cradius^2.k_N = \frac{2.\alpha^2.N^2}{k_N} \quad (5)$$

Two cases can be considered:

i) ρ_N is constant, meaning that the compression rate is constant, then: ∀*N*, $\rho_N=\rho_0$ in that case k_N is linearly increasing with *N*: $k_N=N.\rho_0$ and $\mathcal{C}_N = \frac{2.\alpha^2.N}{\rho_0}$ is linear.

ii) K_N is constant: meaning that the number of segments composing the approximation is fixed, whatever the length of the time series, then: ∀*N*, $k_N=k_0$ leading to a decreasing of ρ_N according to $\rho_N = k_0/N$. With that condition, $\mathcal{C}_N = \frac{2.\alpha^2.N^2}{k_0}$ becomes a quadratic function of *N*.

Complexity of Multi-Resolution MR-PyCA

If we consider that for all *N* and a given k_0 there exist a natural number *r* and a constant ρ_0 in]0;1[such that $\rho_N = \frac{k_0}{N} = \rho_0^r$, and defining $k_j = \rho_0^j.N$, then, using the *PyCA* algorithm, we can obtain from an original curve of size *N* a polygonal curve approximation having k_1 (k_1-*PyCA*) segments with complexity:

$$\mathcal{C}^{MR}_{N,1} = \frac{2.\alpha^2.N^2}{k_1} = \frac{2.\alpha^2.N}{\rho_0}$$

If we consider as a second step the simplification of the k_1-*PyCA* curve still using the *PyCA*, then we get a polygonal curve approximation having k_2 segments from the k_1-*PyCA* curve with complexity

$$\mathcal{C}^{MR}_{N,2} = \frac{2.\alpha^2.k_1^{\,2}}{k_2} = \frac{2.\alpha^2.(\rho_0.N)^2}{\rho_0^{\,2}.N} = 2.\alpha^2.N$$

Iterating the process, we get:

$$\mathcal{C}^{MR}_{N,3} = \frac{2.\alpha^2.k_2^{\,2}}{k_3} = \frac{2.\alpha^2.(\rho_0^{\,2}.N)^2}{\rho_0^{\,3}.N} = 2.\alpha^2.\rho_0.N$$

By induction, it is easy to show that for all *j* in *{1,..,R}* we have :

$$\mathcal{C}^{MR}_{N,j} = \frac{2.\alpha^2.k_{j-1}^2}{k_j} = \frac{2.\alpha^2.(\rho_0^{\,j-1}.N)^2}{\rho_0^{\,j}.N} = 2.\alpha^2.\rho_0^{\,j-2}.N \quad (6)$$

Finally, from the original time series of size *N*, we get after *r* iterations of the previous process a polygonal approximation having $k_i = \rho_0^r.N$ segments with a complexity:

$$\mathcal{C}^{MR}_{N} = \sum_{j=1}^{R} C_{N,j} = 2.\alpha^2.N.(\frac{1}{\rho_0}+1+\rho_0+\rho_0^{\,2}+...+\rho_0^{\,R-2})$$

$$= \frac{2.\alpha^2.N}{\rho_0}.\frac{1-\rho_0^{\,R}}{1-\rho_0} \quad (7)$$

as $\rho_0 \in]0;1[$, we get the following upper bound showing that the *AMR-PyCA* complexity is *O(N)*:

$$\mathcal{C}^{MR}_{N} \leq \frac{2.\alpha^2.N}{\rho_0.(1-\rho_0)} \quad (8)$$

We note that this upper bound is minimized for $\rho_0 = 1/2$.

4. Elastic Alignment of Adaptive Multiresolution Representation of Time Series(*AMR-DTW*)

The objectives behind the development of an elastic alignment algorithm dedicated to the matching of adaptive multiresolution representations of time series are two folds. The first one is related to computation efficiency: we want to break down the quadratic complexity of elastic pseudo distances based on dynamic time warping that are commonly used when similarity between time series is intensively required. The second one is related to the multiresolution representation space itself: if we can derive a suitable pseudo-metric for this space, we will be able to handle time series directly within this space with the capability to work at any resolution level in a very efficient way. Our work is similar than the one proposed by [24], in the sense that we also tackle linear complexity through a multi level approach. The main difference relies in the adaptive feature of the developed multiresolution representation of time series: while in [24] the authors proposed a coherent down sampling of time series, we use an adaptive and recursive down sampling approach based on the polygonal curve simplification procedure described above that keeps more samples when local bandwidth is high and less samples when the local bandwidth is low. This allows adopting a very efficient matching algorithm similar to the one described in *FastDTW* but which implements a refinement procedure that constrains the search around the local artefacts of time series.

Basically, the matching algorithm is initialized at resolution *R*, e.g. the crudest resolution level. Using a fixed size corridor, the pseudo distance between two multiresolutions *mr(X)* and *mr(Y)* is initialized as the standard "constrained search" dynamic time warping between time series *mr(X,R)* and *mr(Y,R)* the samples of which are formed with the segment extremities of each respective polygonal approximation at resolution level *R*. This elastic alignment provides a path *P(X,Y,R)* or a mapping between the time stamps associated to the *mr(X,R)* samples and the time stamps associated to the *mr(Y,R)* samples. We then define an adaptive corridor *c(X,Y,R-1)* that limits the search space at resolution level *R-1* using a fixed corridor radius around the resampled path $P^{R-1}(X,Y,R)$ at level *R-1* and a constrained dynamic time warping matching (CDTW) exploiting this adaptive corridor. This leads to the following recursive equation:

$$\delta_j(X,Y) = CDTW_j(mr(X,j), mr(Y,j)),\ \ R > j \geq 0 \quad (9)$$

where $CDTW_i$ is the constrained time warping match between *mr(X,j)* and *mr(Y,j)* using the corridor constraints:

$$C_{\min}(X,Y,j) = P^j(X,Y,j+1) - Cradius$$
$$C_{\max}(X,Y,j) = P^j(X,Y,j+1) + Cradius \quad (10)$$

$P^j(X,Y,j+1)$: matching path obtained at resolution level $j+1$, resampled for level j.

Finally, we define the pseudo distance between two multiresolution representations associated to the *X* and *Y* time series as: $AMR-DTW(X,Y) = \sum_{j=0}^{R-1} \delta_j(X,Y)$.

<u>*Complexity analysis of the AMR-DTW computation*</u>

We define N_X and N_Y as the lengths of the two time series being compared *X* and *Y* respectively. We furthermore consider that given the multiresolution representation *mr(X)* (resp. *mr(Y)*), the sample compression factor $\rho^j_{0,X}$ (resp. $\rho^j_{0,Y}$) is constant between each resolution levels. For any resolution level $j \in \{0,1,2,...,r-1\}$, the number of samples contained inside the j^{th} polygonal approximation of *X* (resp.*Y*) is $k_{j,X} = \rho^j_{0,X} N_X$ (resp. $k_{j,Y} = \rho^j_{0,Y} N_Y$).

The complexity of the matching at resolution level *R* verifies: $\mathcal{C}\delta_R(X,Y) \leq 2.Cradius.\rho_0^R.N$, where $\rho_0 = Max\{\rho_{0,X}, \rho_{0,Y}\}$ and *N=Max{N_X, N_Y}*. Up to resolution level *j* the matching cost verifies:

$$\mathcal{C}\delta_R(X,Y) \leq 2.Cradius.\rho_0^j.N + \sum_{k=j-1}^{R} 2.Cradius.\rho_0^k.N$$

By straightforward induction we get:

$$\mathcal{C}\delta_0(X,Y) \leq \sum_{k=0}^{R} 2.Cradius.\rho_0^k.N = 2.Cradius.N.\frac{1-\rho_0^{R+1}}{1-\rho_0}$$
$$\leq \frac{2.Cradius.N}{1-\rho_0} \quad (11)$$

showing the linear complexity of *AMR-DTW.*

5. Experimentations

For various *N* values, 100 couples of time series are randomly computed, the adaptive multiresolution representation of the time series of each couple are processed then matched using *AMR-DTW.* The empiric scalability of the *AMR-PyCA* and *AMR-DTW* algorithms is shown in Fig. 2. The experimental time complexity shown in Fig.2 for the whole process is linear with *N* as expected for various *ro* values.

The classification tasks we consider consist in affecting to an unknown time series one of the possible categories for the 16 data sets available at UCR repository [25].

Dataset	Nbr of classes | Size of testing set	1-NN Euclid. Dist.	1-NN DTW Best Warp. Window	1-NN DTW no Warp. Window	1- NN Fast-DTW	1-NN AMR-PyCA + AMR-DTW
Synthetic Control	6|300	0.12	0.017	**0.007**	0.04	0.02
Gun-Point	2|150	0.087	0.087	0.093	0.087	**0.027**
CBF	3|900	0.148	0.004	0.003	0.002	**0.001**
Face (all)	14|1690	0.286	0.192	0.192	**0.183**	**0.197**
OSU Leaf	6|242	0.483	0.384	0.409	0.347	**0.264**
Swedish Leaf	15|625	0.213	0.157	0.210	0.198	**0.115**
50Words	50|455	0.369	**0.242**	0.310	0.343	0.299
Trace	4|100	0.24	0.01	**0.0**	0.06	0.01
Two Patterns	4|4000	0.09	0.0015	**0.0**	0.021	0.003
Wafer	2|6174	**0.005**	**0.005**	0.020	0.022	0.014
Face (four)	4|88	0.216	**0.114**	0.170	0.125	0.148
Lighting2	2|61	0.246	0.131	0.131	0.197	**0.082**
Lighting7	7|73	0.425	0.288	0.274	0.274	**0.247**
ECG	2|100	**0.12**	**0.12**	0.23	0.24	0.22
Adiac	37|391	**0.389**	0.391	0.396	0.437	0.414
Yoga	02|3000	0.170	0.155	0.164	0.155	**0.137**
Fish	7|175	0.267	0.233	0.267	0.2	**0.097**

Tab.1: Comparative study using the UCR datasets[25]

For each datasets, a train subset is defined as well as a test subset. The classification is based on the simple nearest neighbour decision rule: we select first a reference data set containing time series for which the correct category is known. To affect a category to an unknown time series, we select from the reference data set the closest time series (in the sense of a distance or pseudo distance) to the unknown one, then affect to the unknown time series the category associated to its nearest neighbour. *Tab.1* shows the results obtained for the tested methods, e.g. Euclidian distance on the original time series, *DTW* with best warping windows as defined in [20], classical DTW with no warping window, *FastDTW* as defined in [24] and *AMR-DTW* applied on the *AMR-PyCA* representation of time series. For this last method, the main parameters values, namely (ρ, the number of resolution levels R, and the size of the corridors for *AMR-PyCA* and the *AMR-DTW*) are selected such as to minimize the classification errors estimated on the train datasets. More precisely, we set the number of segments of the crudest approximation $K=2$, R varies from 3 to 9 resolution levels and α, the corridor factor for *AMR-PyCA* varies in {1,2}. ρ is calculated from K and R: $\rho = (K/N)^{1/R}$, the corridor radius for *AMR-PyCA* and *AMR-DTW* are respectively *Max{1,α/ρ,} Min{48, Max{1,2.α/ρ}}*. For *FastDTW*, the corridor radius is set to *48* and *K=2*. For this experiment *AMR-DTW* shows to be globally more accurate than *FastDTW* and *DTW* and globally almost as accurate as *DTW* with learned constraints [20].

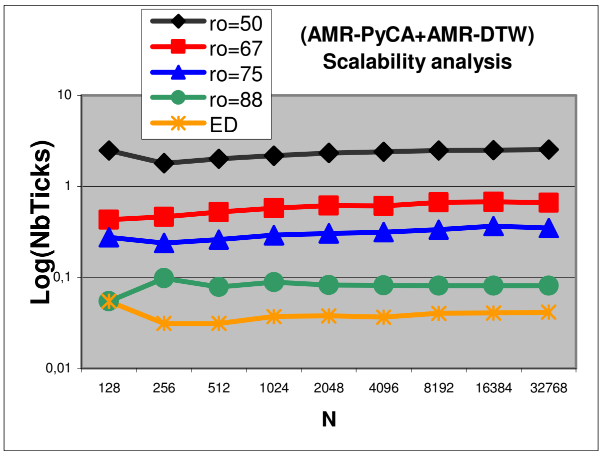

FIG. 2 Time complexity on a logarithmic scale (expressed as the number of clock ticks) for various ρ values. The experimental time complexity for the Euclidian Distance (ED) is given as a baseline (bottom curve).

Since the corridor radius is in the worse case 48 in *AMR-DTW* it is in average faster than *FastDTW* for which the corridor radius is fixed to 48.

6. Conclusion

To our knowledge the proposed *AMR-PyCA* algorithm and dedicated *AMR-DTW* elastic distance are original. They consist in iteratively applying a constraint dynamic programming search algorithm to find (and resp. match) successive approximations of time series using polygonal curve approximation. We have shown both theoretically and practically that these algorithms have linear time complexities (*O(N)*), whatever the chosen number of resolution levels. The *AMR-PyCA* algorithm does not provide a single approximation, but a family of nested approximations from the finest to the crudest approximating levels with increasing distance between the original curve and the successive approximations. This algorithm is sub optimal but maintains partial optimality between each resolution levels. It offers good approximating solutions in cases where real time and storage space are issues, namely each time the optimal solution cannot be calculated due to the size of *N*. Furthermore, the multiresolution aspect of the method allows managing simultaneously or successively various resolution levels, a functionality that is exploited in the *AMR-DTW* elastic pseudo distance that in some way generalizes the *FastDTW* proposed in [24]. could be very useful in time series information retrieval tasks for example. For the considered classification tasks, *AMR-PyCA* associated to *AMR-DTW* outperform *FastDTW* and show similar level of accuracy than *DTW* associated with the best warping window [20]. The *O(N)* complexity as well as the adaptive capability of *AMR-DTW* make this pseudo elastic distance for adaptive multiresolution a potential candidate for time series data mining applications.

7. References

[1] Agarwal, P. K., Varadarajan, K. R., 2000. Efficient Algorithms for Approximating Polygonal Chains. Discrete & Computational Geometry 23(2): 273-291

[2] Agrawal, R., Psaila, G., Wimmers, E. L. and Zaït, M., 1995. Querying shapes of histories. In Proc. 21th Int. Conf. on Very Large Data Bases, pages 502–514.

[3] Bellman, R. 1957. Dynamic Programming. Princeton University Press.

[4] Boreczky, J. S., Rowe, L. A., 1996. Comparison of video shot boundary detection techniques. In Proc. 8th Int. Symp. Storage and Retrieval for Image and Video db, pp 170–179.

[5] Chan, K., Fu, A. W., 1999. Efficient time series matching by wavelets. In proceedings of the 15th IEEE Int'l Conference on Data Engineering. Sydney, Australia, Mar 23-26. pp 126-133. Science, University of Waterloo.

[6] Chen, L., Ng, R., 2004. "On the Marriage of Lp-Norm and Edit Distance", In *Proceedings of 30th International Conference on Very Large Data Base*, Toronto, pp 792-803.

[7] Douglas, D.H., Peucker, T.K., 1973. Algorithm for the reduction of the number of points required to represent a line or its caricature. The Canadian Cartographer 10(2), 112–122.

[8] Faloutsos, C., Ranganathan, M., Manolopoulos, Y., 1994. Fast subsequence matching in time-series databases. In Proc. ACM SIGMOD Int. Conf. on Management of Data, pages 419–429.

[9] Fink, E., Pratt, B., 2003. "Indexing of compressed time series, *Data Mining in Time Series Databases*, pp.51-78.

[10]Keogh, E. J., Pazzani, M. J., 2000: Scaling up dynamic time warping for datamining applications. KDD pp 285-289

[11] Keogh, E. J., Chakrabarti, K., Mehrotra, S., Pazzani, M. J., 2001. Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. SIGMOD Conf.

[12] Kim, S., Park, S., Chu, W., 2001. An indexed-based approach for similarity search supporting time warping in large sequence databases. In Proc. 17th Int. Conf. on Data Engineering, pages 607–614.

[13] Kolesnikov A. and Fränti P., 2003. Reduced-search dynamic programming for approximation of polygonal curves Pattern Recognition Letters 24 (2003) 2243–2254

Kolesnikov, A., Fränti, P., Wu, X., 2004. Multiresolution

[14] Polygonal Approximation of Digital Curves, Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04), 1051-4651.

[15] Le Buhan Jordan, C. , Ebrahimi, T., Kunt, M. , 1998. Progressive content-based shape compression for retrieval of binary images, *Computer Vision and Image Understanding*, 71(2): 198-212.

[16] Levenshtein, V. I., 1966. Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys. Dokl.*, 6:707-710.

[17] Marteau, P.-F., Gibet, S., 2005. Adaptive Sampling of Motion Trajectories for Discrete Task-Based Analysis and Synthesis of Gesture. In *LNCS/LNAI Proc. of Int. Gesture Workshop*, pp. 224 - 235 Publisher.

[18] Perez, J.C., Vidal, E., 1994. Optimum polygonal approximation of digitized curves, *Pattern Recognition Letters*, 15: 743-750.

[19] Pikaz, A., Dinstein, I., 1995. An algorithm for polygonal approximation based on iterative point elimination, *Pattern Recog. Let.*, 16 (6): 557-563.

[20] Ratanamahatana, C. A. and Keogh. E. 2004. *Making Time-series Classification More Accurate Using Learned Constraints_ SDM '04, Florida, April. pp. 11-22.*

[21] Rosin, R.L., 1997. Techniques for assessing polygonal approximations of curves. IEEE Trans PAMI 14, 659–666.

[22] Sakoe, H., Chiba, S., 1978. Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoustics Speech Signal Process ASSP 26:43–49

[23] Salotti, M., 2001; An efficient algorithm for the optimal polygonal approximation of digitized curves, *Pattern Recognition Letters*, vol. 22, pp. 215-221.

[24] S. Salvador and P. Chan., 2004. FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. In Proc. KDD Workshop on Mining Temporal and Sequential Data,.

[25] UCR Time Series Classification/Clustering Page, Keogh & al., http://www.cs.ucr.edu/~eamonn/time_series_data/

[26] Visvalingam M., Whyatt J., 1993. Line generalization by repeated elimination of points, *Cartographic Journal*, 30(1): 46-51.

[27] Yi, B.-K., Jagadish, H. Faloutsos, C., 1998. Efficient retrieval of similar time sequences under time warping. In Proc. 14th Int. Conf. on Data Engineering, pages 23–27.

[28] Zhu, Y. Shasha, D., 2003. Warping indexes with envelope transforms for query by humming. In Proc. ACM SIGMOD Int. Conf. on Management of Data, pp 181–192.