Accurate and Fast Similarity Detection in Time Series

Francesco Gullo, Giovanni Ponti, Andrea Tagarelli, Sergio Greco

DEIS, University of Calabria
e-mail: {fgullo,gponti,tagarelli,greco}@deis.unical.it

Abstract. This paper addresses the problem of similarity detection in time series from both an effectiveness and efficiency viewpoint. A main motivation underlying our work is that, as we experimentally proved, no state-of-the-art method has capabilities for both fast and accurate similarity detection in time series. Viewed in this respect, we propose *DSA (Derivative time series Segment Approximation)*, a representation model for time series that suitably combines the notions of derivative estimation, segmentation and segment approximation to support effective and efficient similarity detection. Experiments conducted in a hierarchical clustering framework show that DSA based similarity detection is as good or better than both the most accurate and the fastest methods among the competing ones.

1 Introduction

A time series, or time sequence, T is a list of (real) numeric values upon which a total order based on timestamps is defined. The traditional form $T = [(x_1, t_1), \ldots, (x_n, t_n)]$ can be rewritten as $T = [x_1, \ldots, x_n]$ when, as usual, a fixed sampling period is assumed. Significant amounts of time series data are naturally available on several sources of different domains, such as speech recognition, biomedical measurement, financial and market data analysis, telecommunication and telemetry, sensor networking, motion tracking, and meteorology.

In recent years, all of these application domains have raised the demand for suitable solutions to the problem of identifying similarities among time series data. Addressing such a problem is significant for different tasks, such as indexing and query processing, change detection, rule discovery, and classification/clustering. In this context, the basic approach is dynamic time warping, and the most relevant methods are its extensions, possibly including techniques borrowed from string matching based on edit distance. Also, high dimensionality of time series has raised the demand for dimensionality reduction techniques to improve the efficiency of similarity searches.

The main contribution of this paper is twofold. At a first stage, we reviewed the state-of-the-art in time series data management focusing on existing solutions for two major issues, namely *similarity detection* and *dimensionality reduction*. Our empirical study pointed out that no existing technique can be used to perform high accuracy similarity detection while maintaining low the computational

effort. This finding led us to devise a representation scheme for time series that is able to support both effective and efficient similarity detection.

Within this view, we propose *DSA – Derivative time series Segment Approximation*, a time series representation model based on an original combination of the notions of derivative estimation, segmentation and segment approximation. DSA allows for modelling time series into a concise yet feature-rich representation, thus enabling fast and accurate time series matching and similarity detection.

Experimental results conducted in a hierarchical clustering framework show that DSA-based time series similarity detection is as good or better than both the most accurate and the fastest methods among the competing ones, thus guaranteeing the best trade-off between effectiveness and efficiency.

2 Related Work

2.1 Similarity detection

Similarity detection in time series in principle should meet the following requirements: handling local time shifting, high efficiency in computation, low sensitivity to noise, and support for indexing. The Euclidean distance (L_2), initially used in [1], is fast to compute and is a metric, but it is unable to deal with noisy sequences and sequences with different lengths or shifted in the time axis. Superior approaches are based on warping the time axis and on string matching measures.

Warping the time axis allows to achieve the best alignment between data points of two time series. The Dynamic Time Warping (DTW) algorithm has long been known in speech recognition [2], then was introduced to the data mining community as an effective solution to the sensitivity of the Euclidean distance to small distortions (i.e. fluctuations or phase shifts) in the time axis [3]. Given two sequences T_1 and T_2, DTW performs a non-linear mapping of one sequence to another by minimizing the total distance between them. Initially, a $(|T_1| \times |T_2|)$-matrix is built to contain the squared Euclidean distances between T_1's points and T_2's points. To find the best alignment between the two sequences, a warping path (i.e. a sequence of matrix elements) is computed in such a way that: it starts and ends in diagonally opposite corner cells of the matrix, all elements in the path are contiguous and monotonically spaced in time, and the total cumulative distance is minimized. The optimal path is retrieved by using a dynamic programming algorithm, whose complexity is $\mathcal{O}(|T_1| \times |T_2|)$.

DTW can handle time series with local time shifting and different lengths, although it is not a metric, unlike the Euclidean distance. Pruning techniques proposing computationally cheap lower bounds (e.g. [4–7]) have been defined to make DTW able to support indexing. In particular, the lower bounding measure proposed in [4] has been extensively used in several application contexts (e.g. [8–12]). This measure uses a bounding envelope that encloses one of two series being compared and is defined by an upper bound sequence U and a lower bound sequence L. In general, such bound sequences are defined depending on the specific

domain. A similar lower bounding measure, named FTW (Fast search method for dynamic Time Warping), has been recently proposed in [7]. FTW approximates DTW for purposes of query processing (i.e. efficient k-nearest neighbor and range queries) and leverages the inability of exact DTW for long sequences, due to its quadratic complexity. For this purpose, FTW proposes to estimate the time warping distance by using a lower bounding distance measure on a coarse and compact version of the original sequences.

A major disadvantage of DTW is that it tends to produce "singularities", that are alignments of a single point in a sequence with multiple points of another sequence. This phenomenon becomes undesirable when unexpected singularities are produced. An effective variant of DTW able to reduce the phenomenon of singularities is Derivative Dynamic Time Warping (DDTW) [13]. The novelty of DDTW is that local derivatives of data points are estimated to capture information on slopes and trends in the sequences and find the correct warping.

An alternative approach to time series similarity detection is based on string matching measures. LCSS (*Longest Common SubSequence*) [14] is a variant of the edit distance that uses the length of the longest common subsequence of two sequences to define the distance between them. LCSS can handle time series with noise, but suffers from large-grained similarity.

A more refined measure based on edit distance is EDR (*Edit Distance with Real sequences*) [15], which performs the same distance quantization of LCSS (parametric with respect to a certain tolerance threshold) to remove noisy effects. In contrast to LCSS, EDR penalises the gaps between two matched subsequences according to the lengths of gaps. Unlike LCSS and EDR, ERP (*Edit distance with Real Penalty*) [16] is a metric and still supports local time shifting. ERP can be seen as a variant of EDR and DTW, although it does not require a noise-tolerance threshold like EDR, and does not replicate previous data points to add a gap like DTW. However, ERP shares with EDR and DTW the computational upper bound.

2.2 Dimensionality reduction

In order to improve the efficiency in time series data management, many research works have focused on *dimensionality reduction* to obtain a higher-level data representation. Typically, the goal is to approximate a (continuous) time series either with a piecewise discontinuous function or a low-order continuous function.

The first category includes Discrete Wavelet Transform (DWT) [17, 18], Piecewise Aggregate Approximation (PAA) [19, 20], and Adaptive Piecewise Constant Approximation (APCA) [21]. Using DWT, a time series is represented as a finite length, fast decaying oscillating waveform (mother wavelet), which is scaled and translated to match the original series. Unlike the continuous version of wavelet transform, the mother wavelet in DWT is discretely sampled.

PAA transforms a time series of n points in a new one composed by p segments (with $p \ll n$). All these segments have size equal to n/p. A segment is represented by a coefficient, which is the mean value of the data points falling

within the segment. Like PAA, APCA approximates a time series with a sequence of segments, each represented by the mean value of data points falling within it. A major difference is that APCA identifies segments of variable length. The $\mathcal{O}(n \log n)$ APCA algorithm is able to produce high-quality approximation by resorting to well-known solutions from the wavelet domain.

Dimensionality reduction techniques can be combined with existing similarity measures, in order to improve the computational cost in similarity searches. In particular, the use of DTW on the coefficients obtained by segmentation of time series has been investigated [22, 20].

Approaches that approximate a time series with a continuous polynomial include Discrete Fourier Transforms (DFT) [23, 24], splines, non-linear regression, and Chebyshev polynomials [25]. A very desirable requirement is the minimax approximation, i.e. an approximation that minimises the maximum deviation from the original data points. Although the optimal minimax polynomial is difficult to compute, it has been demonstrated that the Chebyshev approximation is very close to this polynomial and can be easily computed [26].

3 Derivative Time Series Segment Approximation

In this section we present a method for modelling time series into a compact representation, which suitably synthesizes the significant variations in the time series profile. The method is called *DSA (Derivative time series Segment Approximation)*, as it intuitively segments the derivative version of a time series before of approximating it into a high level representation.

Let $T = [x_1, \ldots, x_n]$ be a time series, where the timestamp associated to x_1 is assumed to be zero. DSA computes in $\mathcal{O}(n)$ a new sequence τ of p values, with $p \ll n$, in three main steps: *(i)* derivative estimation, *(ii)* segmentation, and *(iii)* segment approximation.

3.1 Derivative estimation

Given a time series $T = [x_1, \ldots, x_n]$, the derivative estimation step yields a sequence $\dot{T} = [\dot{x}_1, \ldots, \dot{x}_n]$, whose elements $\dot{x}_i$ are first derivative estimates. A simple derivative estimation model, exploited in [13] and hereinafter referred to as DDTW model, is the following:

$$\dot{x}_i = \begin{cases} \dot{x}_{i+1} & \text{if } i = 1 \\ \frac{1}{2}[(x_i - x_{i-1}) + \frac{1}{2}(x_{i+1} - x_{i1})] & \text{if } i \in [2..n\text{-}1] \\ \dot{x}_{i1} & \text{if } i = n. \end{cases}$$

This estimation model computes for each point (except the first and the last one in the series) the mean value between the slope of the line from the left neighbor to the point and the slope of the line from the left neighbor to the right neighbor.

We slightly modify the above model by considering also the slope of the line from the point to the right neighbor; this modification leads to an algebraic simplification producing an expression which is equivalent to consider only the

slope of the line from the left neighbor to the right neighbor. Neighbors are also considered when computing the derivatives of the first and last points:

$$\dot{x}_i = \begin{cases} x_{i+1} - x_i & \text{if } i = 1 \\ \frac{1}{2}(x_{i+1} - x_{i1}) & \text{if } i \in [2..n\text{-}1] \\ x_i - x_{i1} & \text{if } i = n. \end{cases}$$

As experimentally founded, our derivative estimation model surprisingly gives approximation errors lower than the DDTW model.

3.2 Segmentation

The segmentation of a time series of length n consists in identifying $p-1$ points ($p \ll n$) to partition it into p contiguous subsequences of points, i.e. segments, having similar features.

In our approach, the novel idea is that the segmentation is computed on derivative versions of time series. In particular, the derivative time series $\dot{T} = [\dot{x}_1, \ldots, \dot{x}_n]$ is transformed into a sequence $S_{\dot{T}} = [s_1, \ldots, s_p]$ of variable-length segments $s_i = [s_{i,1}, \ldots, s_{i,k_i}] = [\dot{x}_{i_1}, \ldots, \dot{x}_{i_{k_i}}]$, such that:

- $s_{1,1} = \dot{x}_1$,
- $s_{p,k_p} = \dot{x}_n$,
- for each $i \in [1..p\text{-}1]$, s_{i,k_i} immediately precedes $s_{i+1,1}$ in the time axis.

The critical aspect in segmentation is to determine the segment delimiters. Our approach falls into the sliding windows category: a segment is grown until it exceeds an error threshold, and the process repeats starting from the next point not yet considered. The key idea in our method is simply to break a series according to the first point such that the absolute difference between it and the mean of the previous points is above a certain threshold; this point becomes the anchor for the next segment to be identified in the rest of the series.

Formally, let $\mu(s_i)$ denote the average of the points in a potential segment s_i, defined as $\mu(s_i) = \frac{1}{k_i} \sum_{j=1}^{k_i} \dot{x}_{i_j}$. The sequence s_i, for each $i \in [1..p\text{-}1]$, is identified as a segment if and only if $|\mu([s_{i,1}, \ldots, s_{i,j}]) - s_{i,j+1}| \leq \epsilon$, for all $j \in [1..k_{i-1}]$, and $|\mu([s_{i,1}, \ldots, s_{i,k_i}]) - s_{i+1,1}| > \epsilon$.

Intuitively, this condition allows for aggregating subsequent data points having very close derivatives. In such a way, the growth segment represents a subsequence of points with a specific trend.

Parameter ϵ can be estimated in principle by considering an index of dispersion of the (derivative) data points within the same sequence around the respective mean value. Estimating ϵ might be accomplished based on one of three different contexts: globally to a given collection of time series, globally to a given time series, or locally to a given time series.

Given a dataset of N time series, the first way of computing ϵ may lead to the following definition: $\epsilon = \frac{1}{N} \sum_{k=1}^{N} \frac{|\dot{T}_k|}{\max_{i=1}^{N}\{|\dot{T}_i|\}} \sigma^2(\dot{T}_k)$ where $\sigma^2(\dot{T}_k)$ denotes the variance over the points in the k-th derivative series. The normalisation of the lengths is significant if the assumption of equally-sized series does not hold. The above definition can be adequate for the purpose provided that the time

series of a collection are quite equally sized. This cannot necessarily hold in some real domains (e.g. sensor network measurements) in which the time series generated may have varying lengths.

An opposite solution consists in estimating threshold ϵ locally to each time series, and in particular as function of the segment s_i which is currently being identified, that is, e.g. $\epsilon(s_i) = \sigma^2(s_i)$. However, although intuitively more accurate, this way of computing ϵ might be expensive.

A good trade-off between a collection-global and a series-local computation of ϵ is represented by the definition of a series-global computation: $\epsilon(\dot{T}_i) = \sigma^2(\dot{T}_i)$. We hereinafter refer to that as the definition of ϵ adopted in the DSA model.

While the main dimensionality reduction methods (Chebyshev polynomials, PAA and APCA) require in input the number of segments or coefficients which have to be identified, DSA does not need any input parameter. This is an important advantage of our method.

3.3 Segment approximation

All individual segments of a derivative time series are approximated with a synthetic information capturing their respective main features. More precisely, each segment s_i is mapped to a pair formed by the timestamp t_i of the last point ($\dot{x}_{i_{k_i}}$) of s_i and an angle that explains the average slope of the portion of time series bounded by s_i. This is mathematically expressed by the notion of arctangent applied to the mean of the (derivative) points in each segment.

Given a segmented derivative time series $S_{\dot{T}} = [s_1, \ldots, s_p]$, a sequence $\tau = [(\alpha_1, t_1), \ldots, (\alpha_p, t_p)]$ is computed, where

$$\alpha_i = \arctan(\mu(s_i)), \quad \text{for } i \in [1..p],$$
$$t_i = \begin{cases} k_i & i = 1 \\ t_{i-1} + k_i & i \in [2..p]. \end{cases}$$

4 Experiments

Experiment devised to assess both effectiveness and efficiency of the previously presented methods, including our DSA, in a clustering framework. Specifically, we tested LCSS, EDR, ERP DTW, DDTW and FTW directly as distance measures. Moreover, in order to include Chebyshev, PAA, APCA and our DSA in the comparative evaluation of distance measures, we chose to apply DTW over the sequences computed by each approximation scheme. The goal of the evaluation was to demonstrate the superiority of DSA to achieve fast and accurate similarity detection of time series in comparison with competing methods.

Input parameters required by LCSS, EDR, ERP, FTW and Chebyshev have been chosen as suggested in their respective works: constant gap for ERP has been set to 0, time interval for FTW has been set to 4, number of coefficients for Chebyshev has been set to 20 and the matching thresholds for LCSS and EDR have been assumed to be equal to $\frac{1}{4}\max\{\sigma(T_i)\}$ and $\min\{\sigma(T_i)\}$, for all T_i in the target dataset, respectively. As concerns PAA and APCA, since no indication about how to set the number of segments (p) is provided in their respective

works, we chose p as follows: for effectiveness evaluation, p was set as the number of segments produced by DSA, in order to evaluate accuracy reached at the same compression level; instead, for efficiency evaluation, we firstly measured effectiveness for different compression levels by varying p and, finally, chose p according to the best trade-off between accuracy and computational time.

4.1 Data description

We selected well-known datasets in the time series domain according to two main aspects: heterogeneity of the series profiles and significance of the series lengths. Table 1 summarises the main characteristics of the datasets, which are mostly available at http://www.cis.temple.edu/∼latecki/TestData/TS_Koegh/.

dataset	size	classes	time steps
GunX	200	2	150
Trace	200	4	275
ControlChart	600	6	60
CBF	300	3	128
Twopat	800	4	128

Table 1. Main characteristics of the test datasets

GunX comes from the video surveillance domain, whereas Trace simulates signals representing instrumentation failures. In CBF, each class is characterized by a specific pattern, namely a plateau (C), an increasing ramp followed by a sharp decrease (B), a sharp increase followed by a decreasing ramp (F). ControlChart contains synthetically generated control charts which are classified into 6 classes. In Twopat, two different patterns (upward step and downward step) are used to define the classes.

4.2 Clustering method and quality measures

In our work the choice of a clustering scheme is functional to a comparative evaluation of methods for similarity detection. In this paper we employ a traditional agglomerative hierarchical clustering (AHC) algorithm [27] into our clustering framework.

Since the availability of reference classifications for the test datasets, the desired number of clusters at a hierarchy level is used as AHC termination criterion. Also, evaluating clustering effectiveness can be accomplished by adopting an external validity criterion; we use the well-known F-measure (ranging within [0..1]), which is defined in terms of classic Information Retrieval notions' precision and recall [28]. The objective is to assess how well a clustering fits a predefined scheme of known classes (natural clusters).

4.3 Preprocessing time series

A preliminary step may consists in preprocessing raw time series in order to dampen the effect due to noise in data and to improve both effectiveness and efficiency in the clustering task. This is usually accomplished by using some

smoothing models. Moving average represents the simplest family of smoothing models, as it is a compromise between the mean model and the random walk model. Given an original series $T = [x_1, \ldots, x_n]$, a centered q-point moving average recomputes the data points as follows:

$$x_i^{smoothed} = \begin{cases} \mu([x_1, \ldots, x_{i+r}]) & \text{if } i-r \leq 0 \\ \mu([x_{i-r}, \ldots, x_{i+r}]) & \text{if } i-r > 0 \text{ and } i+r \leq n \\ \mu([x_{i-r}, \ldots, x_n]) & \text{if } i+r > n \end{cases}$$

where q is the smoothing degree (i.e. the maximum width of the moving average) and $r = (q-1)/2$ denotes the maximum number of back and forward points considered for smoothing the i-th point. The centered q-point version considers both previous and next observations around a center, unlike simple moving average, although it still treats these points equally.

More refined models, such as exponential smoothing models, compute the weighted average of past observations with more weight given to the more recent values and decreasing weights for earlier values. The formula for simple exponential smoothing is as follows:

$$x_i^{smoothed} = \begin{cases} x_i & \text{if } i = 1 \\ wx_i + (1-w)x_{i-1}^{smoothed} & \text{if } i > 1 \end{cases}$$

where w is a smoothing weight with values in [0..1].

4.4 Effectiveness evaluation

In this section we aimed at checking the ability of DSA and the competing methods in producing high-quality clustering. To accomplish this, we explored how clustering results can be influenced by choosing different alternatives for data preprocessing and setting the parameters therein involved. Then, we compared DSA with the competing methods according to their respective best settings.

Tuning preprocessing parameters. We employed two schemes of preprocessing, based on centered moving average and simple exponential smoothing. In the first case, we had to set the smoothing degree $q(= 2r+1)$, whereas in case of exponential smoothing we tried different values for the smoothing weight w. In particular, q was set to typical values, i.e. 5 and 9, and w was varied within [0..1] by a 0.1 step. We performed multiple iterations of smoothing (up to 5) in order to handle possibly excessive noise. On the other hand, we also tried to not perform any smoothing operation in order to prevent unnecessary loss of information for series with very low noise.

We performed tuning tests on DSA as well as on the competing methods. For the sake of brevity, Table 2 summarises the best preprocessing setups for DSA and the other methods on the selected datasets. Term MA (resp. EXP) stands for moving average (resp. exponential smoothing) and is followed by the value set for q (resp. w) and the number of iterations.

DSA vs. competing methods. Table 3 gives a summary of results obtained by DSA and the competing methods from an accuracy viewpoint. The reported results correspond to F-measure values obtained by algorithm AHC, and they

	GunX	Trace	ControlChart	CBF	Twopat
DTW	MA q=9 it=1	No preproc.	EXP w=0.8 it=1	EXP w=0.5 it=1	EXP w=0.8 it=3
LCSS	MA q=5 it=1	No preproc.	MA q=9 it=4	MA q=9 it=4	MA q=5 it=1
EDR	MA q=9 it=4	MA q=5 it=3	EXP w=0.5 it=2	EXP w=0.8 it=2	MA q=5 it=2
ERP	EXP w=0.6 it=1	No preproc.	MA q=5 it=1	EXP w=0.15 it=5	EXP w=0.5 it=3
DDTW	EXP w=0.1 it=1	MA q=9 it=1	EXP w=0.2 it=3	MA q=9 it=3	MA q=9 it=1
FTW	EXP w=0.1 it=1	No preproc.	EXP w=0.8 it=1	EXP w=0.1 it=2	EXP w=0.3 it=3
DTW on Chebyshev	EXP w=0.2 it=1	EXP w=0.9 it=1	MA q=5 it=4	EXP w=0.5 it=2	MA q=9 it=5
DTW on PAA	EXP w=0.5 it=1	EXP w=0.1 it=1	EXP w=0.6 it=4	EXP w=0.6 it=2	EXP w=0.8 it=5
DTW on APCA	MA q=9 it=3	MA q=9 it=1	MA q=9 it=2	MA q=5 it=3	MA q=9 it=4
DTW on DSA	**EXP w=0.3 it=3**	**MA q=9 it=4**	**EXP w=0.5 it=5**	**EXP w=0.3 it=3**	**EXP w=0.3 it=3**

Table 2. Summary of best preprocessing setups

	LCSS	EDR	ERP	DTW	DDTW	FTW	DTW on Chebyshev	DTW on PAA	DTW on APCA	**DTW on DSA**
GunX	0.60	0.68	0.67	0.67	0.64	0.67	0.67	0.66	0.64	**0.68**
Trace	0.36	0.58	0.77	0.75	1	0.77	0.65	0.77	0.74	**1**
ControlChart	0.66	0.83	0.81	0.86	0.86	0.83	0.80	0.86	0.71	**0.86**
CBF	0.66	0.84	0.65	0.72	0.99	0.67	0.69	0.72	0.82	**0.99**
Twopat	0.39	0.41	0.39	0.81	1	0.49	0.45	0.66	0.72	**1**

Table 3. Summary of best clustering quality results

refer to the best preprocessing setups respectively chosen for each similarity method on each dataset (see Table 2).

Table 3 shows that DSA fares as good as DDTW, which turns out to be mostly the best performing method among the remaining ones. Notice also the relative better performance of DDTW against FTW, Chebyshev, PAA and APCA. EDR and ERP perform similarly and both outperform LCSS. Yet, DDTW behaves as good or far better than DTW on all datasets but GunX, whose 2-class series have trends very similar but shifted in the time axis. The 2-class composition of this dataset is probably a reason why clustering results are relatively poor for all methods, suggesting that AHC fails in distinctly separating the two classes of series. We can also observe that DSA drastically outperforms the other dimensionality reduction techniques (Chebyshev, PAA and APCA).

4.5 Efficiency evaluation

We present here the time performances of DSA based clustering, and a comparison with respect to DTW, DDTW, FTW, Chebyshev, PAA and APCA based clustering. We left string matching based approaches out of the presentation since they turned out to be significantly slower than the other methods.[1]

[1] Experiments were conducted on a platform Intel Pentium IV 3GHz with 2GB memory and running on Microsoft Windows XP Pro.

	DTW	DDTW	FTW	DTW on Chebyshev	DTW on PAA	DTW on APCA	**DTW on DSA**
GunX	280,078	330,281	64,500	12,328	56,124	117,125	**62,281**
Trace	947,516	1,420,969	163,347	42,860	200,684	181,750	**87,719**
ControlChart	559,063	478,235	82,656	147,984	163,249	226,016	**104,875**
CBF	643,797	573,547	56,766	36,203	152,462	199,265	**155,968**
Twopat	5,068,859	5,959,391	946,484	315,906	1,260,537	1,630,344	**356,609**

Table 4. Summary of best time performances

Table 4 reports the time performances (in milliseconds) on each dataset, using setups as in Table 2. As we expected, DTW and DDTW are drastically outperformed by the other measures, while APCA is slower than FTW, Chebyshev, PAA and DSA. DSA is comparable to FTW and PAA on all datasets but in Trace and Twopat, in which DSA is faster: this is particularly significant since Twopat and Trace are the largest datasets among the competing ones regards to the number of series and the mean series length, respectively. Chebyshev is found as the faster method; this is mainly due to the choice done for the number of coefficients (20). Nevertheless, DSA remains comparable while achieving drastically higher accuracy.

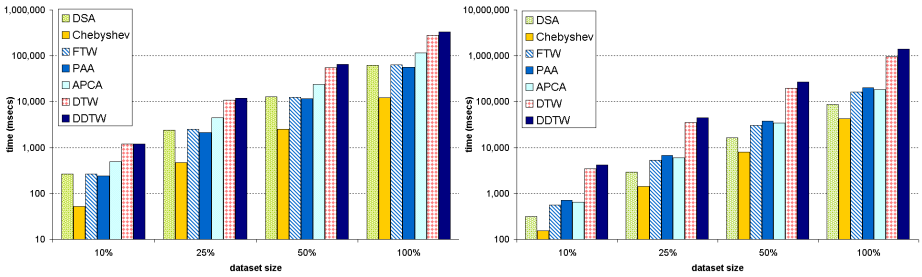
We also carried out experiments to test the time performances by varying the dataset size. Each dataset was sampled in portions with size 10%, 25%, 50%, and 100%; then, for each dataset and for each method, we carried out one run of AHC over each of these portions. Figure 1 shows that the advantage of DSA over DTW and DDTW absolutely increases with the dataset size.

Finally, we evaluated the effect of dimensionality reduction due to the segmentation performed by DSA. Experiments showed that DSA reduces the series lengths from 54% up to 77%, with noteworthy advantages in time performance.

5 Conclusion

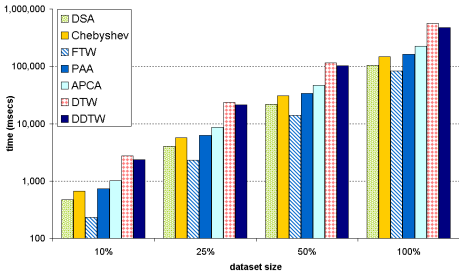
The presented work originally arises from our review of the state-of-the-art of similarity detection in time series, and from the finding that no existing method is able to provide both high effectiveness and efficiency. This prompted us to devise a solution to the identified challenge by combining the notions of derivative estimation and segmentation into a concise yet feature-rich representation model, called *DSA (Derivative time series Segment Approximation)*. Experiments highlight that DSA-based time series similarity detection is as good or better than both the most accurate and the fastest methods among the competing ones, thus guaranteeing the best trade-off between effectiveness and efficiency.

We are currently working on making DSA able to deal with amplitude translation and scaling, which may be significant in some application domains. Moreover, we shall provide the beneficial of DSA in summarising sets of time series to compute cluster prototypes that enable high cluster compactness and separation. Finally, we would like to extend DSA for multidimensional time series (i.e. moving object trajectories), to which many research work has recently paid attention.

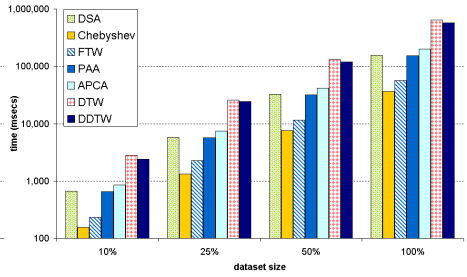


(a) GunX

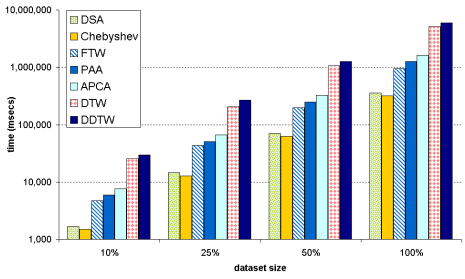
(b) Trace



(c) ControlChart



(d) CBF



(e) Twopat

Fig. 1. Time performances

References

1. Agrawal, R., Faloutsos, C., Swami, A.N.: Efficient Similarity Search in Sequence Databases. In: Proc. FODO Conf. (1993) 69–84
2. Rabiner, L., Juang, B.H.: Fundamentals of Speech Recognition. Englewood Cliffs, N. J. (1993)
3. Berndt, D.J., Clifford, J.: Using Dynamic Time Warping To Find Patterns in Time Series. In: Proc. AAAI Workshop on Knowledge Discovery in Databases. (1994) 359–370
4. Keogh, E.: Exact Indexing of Dynamic Time Warping. In: Proc. VLDB Conf. (2002) 406–417

5. Kim, S.W., Park, S., Chu, W.W.: An Indexed-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases. In: Proc. ICDE Conf. (2001) 607–614
6. Yi, B.K., Jagadish, H.V., Faloutsos, C.: Efficient Retrieval of Similar Time Sequences Under Time Warping. In: Proc. ICDE Conf. (1998) 201–208
7. Sakurai, Y., Yoshikawa, M., Faloutsos, C.: FTW: Fast Similarity Search under the Time Warping Distance. In: Proc. ACM PODS. (2005) 326–337
8. Keogh, E., Palpanas, T., Zordan, V., Gunopulos, D., Cardle, M.: Indexing Large Human-Motion Databases. In: Proc. VLDB Conf. (2004) 780–791
9. Vlachos, M., Hadjieleftheriou, M., Gunopulos, D., Keogh, E.: Indexing Multi-Dimensional Time-Series with Support for Multiple Distance Measures. In: Proc. ACM KDD Conf. (2003) 215–225
10. Wei, L., Keogh, E., Herle, H.V., Mafra-Neto, A.: Atomic wedgie: Efficient query filtering for streaming times series. In: Proc. IEEE ICDM Conf. (2005) 490–497
11. Fung, W., Wong, M.: Efficient Subsequence Matching for Sequences Databases under Time Warping. In: Proc. IDEAS Conf. (2003) 139–148
12. Keogh, E., Wei, L., Xi, X., Lee, S., Vlachos, M.: LB_Keogh Supports Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures. In: Proc. VLDB Conf. (2006) 882–893
13. Keogh, E., Pazzani, M.: Dynamic Time Warping with Higher Order Features. In: Proc. SIAM Int. Conf. on Data Mining. (2001)
14. Vlachos, M., Gunopulos, D., Kollios, G.: Discovering Similar Multidimensional Trajectories. In: Proc. ICDE Conf. (2002) 673–684
15. Chen, L., Özsu, M.T., Oria, V.: Robust and Fast Similarity Search for Moving Object Trajectories. In: Proc. ACM SIGMOD Conf. (2005) 491–502
16. Chen, L., Ng, R.: On The Marriage of Lp-norms and Edit Distance. In: Proc. VLDB Conf. (2004) 792–803
17. Chan, K., Fu, A.: Efficient Time Series Matching by Wavelets. In: Proc. ICDE Conf. (1999) 126–133
18. Wu, Y., Agrawal, D., Abbadi, A.: A Comparison of DFT and DWT Based Similarity Search in Time-Series Databases. In: Proc. ACM CIKM Conf. (2000) 488–495
19. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. Knowledge and Information Systems **3** (2000) 263–286
20. Keogh, E., Pazzani, M.: Scaling up Dynamic Time Warping for Datamining Applications. In: Proc. ACM KDD Conf. (2000) 285–289
21. Keogh, E., Chakrabarti, K., Mehrotra, S., Pazzani, M.: Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. In: Proc. ACM SIGMOD Conf. (2001) 151–162
22. Keogh, E., Pazzani, M.: Scaling up Dynamic Time Warping to Massive Datasets. In: Proc. PKDD Conf. (1999) 1–11
23. Rafiei, D., Mendelzon, A.O.: Efficient Retrieval of Similar Time Sequences Using DFT. In: Proc. FODO Conf. (1998)
24. Rafiei, D., Mendelzon, A.: Similarity-based queries for time series data. In: Proc. ACM SIGMOD Conf. (1997) 13–25
25. Mason, J.C., Handscomb, D.: Chebyshev Polynomials. Chapman & Hall (2003)
26. Cai, Y., Ng, R.: Indexing Spatio-Temporal Trajectories with Chebyshev Polynomials. In: Proc. ACM SIGMOD Conf. (2004) 599–610
27. Jain, A., Dubes, R.: Algorithms for Clustering Data. Prentice-Hall (1988)
28. van Rijsbergen, C.J.: Information Retrieval. Butterworths (1979)