Neural Processing Letters (2020) 52:75–96
https://doi.org/10.1007/s11063-018-9929-y

A Multi-resolution Approximation for Time Series

Heider Sanchez[1] · Benjamin Bustos[1]

Published online: 29 September 2018

Abstract
Time series is a common and well-known way for describing temporal data. However, most of the state-of-the-art techniques for analysing time series have focused on generating a representation for a single level of resolution. For analysing of a time series at several levels of resolutions, one would require to compute different representations, one for each resolution level. We introduce a multi-resolution representation for time series based on local trends and mean values. We require the level of resolution as parameter, but it can be automatically computed if we consider the maximum resolution of the time series. Our technique represents a time series using trend-value pairs on each segment belonging to a resolution level. To provide a useful representation for data mining tasks, we also propose dissimilarity measures and a symbolic representation based on the SAX technique for efficient similarity search using a multi-resolution indexing scheme. We evaluate our method for classification and discord discovery tasks over a diversity of data domains, achieving a better performance in terms of efficiency and effectiveness compared with some of the best-known classic techniques. Indeed, for some of the experiments, the time series mining algorithms using our multi-resolution representation were an order of magnitude faster, in terms of distance computations, than the state of the art.

Keywords Time series · Multi-resolution representation · Classification · Discord discovery

1 Introduction

Time series is a useful way for representing temporal data in a wide variety of real-world applications. Some applications of time series analysis are: economic indicators analysis, meteorological phenomena analysis, human movement and health monitoring, signal processing, etc. This has lead to numerous studies on data mining over time series.

This work was funded by the Millennium Institute for Foundational Research on Data.

✉ Benjamin Bustos
bebustos@dcc.uchile.cl

Heider Sanchez
hesanche@dcc.uchile.cl

1 Department of Computer Science, University of Chile, Santiago, Chile

For dealing with large amounts of time series data, there is a need of developing time series representations that are concise and indexable. This allows one to efficiently implement data mining tasks like clustering, classification, and anomaly detection. For this reason, there are many techniques for transforming a time series into a lower resolution representation. For instance, the most common spectral decomposition technique, Discrete Fourier Transform (DFT), is highly useful for periodic signals and feature-based systems. However, the main disadvantage of DFT over other techniques is its time complexity, that in the best case is $O(n \log n)$. Alternatively, Piecewise Approximation techniques address the segmentation problem with a time complexity of $O(n)$. These techniques require only one parameter, the number of segments or the sliding window size. Some of the better-known indexable techniques are: Piecewise Linear Approximation (PLA) [3,12], Piecewise Aggregate Approximation (PAA) [10] and Adaptive Piecewise Constant Approximation (APCA) [2].

In addition to these numeric representation techniques, new approaches based on symbolic representations have taken notoriety in this field. Symbolic representations are used to offer greater legibility of interpretation, simplicity, and efficient implementation. In this group, Symbolic Aggregate Approximation (SAX) [19] deserves special mention, as well as some of its extensions: Differential Evolution SAX for non-normalized time series [7] and SAX for multivariate time series [21]. Other techniques combine SAX with any additional feature of time series, for example the trend-based techniques [6,20].

All of the aforementioned techniques generate the representation of a time series for a single level of resolution. On the other hand, multi-resolution techniques operate the time series representation using different levels of granularity. The building algorithm of a multi-resolution representation has $O(cn)$ time complexity, where c is a constant of the resolution level. An example of this type of technique is the Discrete Haar Wavelet Transform (DWT) [26]. An advantage of the multi-resolution approach is that it provides us hierarchical access to data by means of resolution levels, for example: a multi-resolution index for binary SAX (iSAX) [23], and an iterative clustering algorithm using Haar Wavelets [17], or Multi-resolution PAA (MPAA) [18].

In this paper, we introduce a new multi-resolution representation based on local trends and mean values of the time series. It becomes a parameter-free technique when we use the maximum level of resolution which will be defined in this work. Unlike DWT and MPAA that only takes into account the average values, our proposed multi-resolution representation also considers the local trends on each level of resolution. We also propose a symbolic representation derived from the numeric representation by applying SAX quantization on the trend and value components, which allows us to provide a lower bounding function for indexing a time series collection. Finally, we propose a multi-resolution discord discovery for anomaly detection based on our proposed time series approximation.

Our proposed multi-resolution representation has several advantages with respect to the state of the art. First, it adds the trend value to the representation, which improves its discriminative power as shown in our experimental evaluation. Second, it provides lower-bounding distances for each resolution level, which can be used to early prune a similarity search thus saving distance computations. Finally, it takes advantage of a symbolic representation that allows us to build a hash table for quickly finding buckets with similar time series, which makes the technique efficient.

The efficacy and efficiency of our approach is experimentally evaluated in two data-mining tasks, classification and anomaly detection, over a diversity of data domains (UCR Time Series Archive [4,9]). We empirically show that our method outperforms conventional methods.

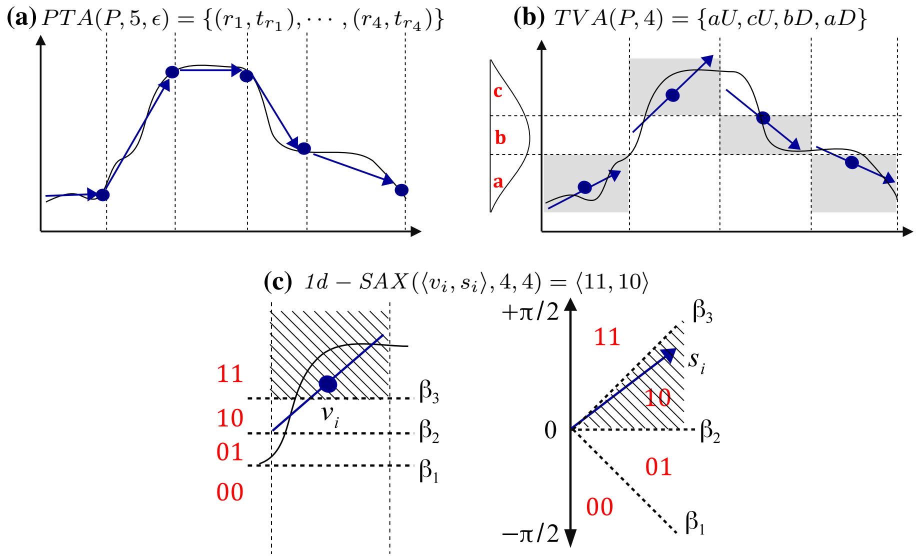
This paper is an extended version of a previous work presented at the 14th International Work-Conference on Artificial Neural Networks (IWANN'17) [22]. The main differences between this article and the conference paper are:

- The aim of the conference paper was improving the discord discovery task. This paper focuses on the multi-resolution proposal as a general technique that can be used for different time series mining tasks.
- We improved the presentation of the state-of-the-art techniques.
- We added a second data mining task, classification, for evaluating the efficiency and effectiveness of the multi-resolution representation.
- We added several experimental evaluations using more time series collections.

1.1 Background

Piecewise Approximation Method splits the time series into segments and builds a new time series with the representative values of each segment. This approach is widely used for dimensional reduction of time series. For example, two well-known piecewise approximation techniques are PAA [11] and SAX [16]. Other related methods are Piecewise Approximations based on Trend and Value features, that is, each segment of the time series is represented by both the local trend and a representative value. The trend can be denoted using the derivative estimation [8], the slope [6], or a radio-based function [5]. Figure 1 shows three recent techniques that use this approach, which are listed below.
Piecewise Trend Approximation (PTA) [5]: Given a time series $P = \{p_1, \cdots, p_n\}$, this algorithm partitions P into $m < n$ variable length segments and builds the new representation:

$$PTA(P, m, \epsilon) = \{(r_1, t_{r_1}), \ldots, (r_m, t_{r_m})\}, \quad (1)$$



(a) $PTA(P, 5, \epsilon) = \{(r_1, t_{r_1}), \cdots, (r_4, t_{r_4})\}$
(b) $TVA(P, 4) = \{aU, cU, bD, aD\}$
c
b
a
(c) $1d - SAX(\langle v_i, s_i \rangle, 4, 4) = \langle 11, 10 \rangle$
11
10
01
00
v_i
β_3
β_2
β_1
$+\pi/2$
$-\pi/2$
0
s_i

Fig. 1 Three techniques for time series representation based on trend and value: **a** Piecewise Trend Approximation, **b** Piecewise Trend-Value Approximation and (c) Extended Trend-Value Approximation

where t_{r_i} is the right endpoint and r_i is the ratio between both endpoints in the *ith* segment. This technique adapts the segments to the original shape of the time series, such that two conjunctive segments represent different trends. This adaptation requires tuning the threshold of ratios ϵ, indicating when two consecutive segments can be aggregated.
Piecewise Trend-Value Approximation (TVA) [6]: It is an extension of SAX by adding new symbols that represent the slope of each segment:

$$TVA(P, m) = \{(\hat{v}_1, \hat{s}_1), \ldots, (\hat{v}_m, \hat{s}_m)\}, \tag{2}$$

where $\hat{v}_i \in \{a = \text{low},\ b = \text{normal},\ c = \text{high}\}$ is the SAX symbol and $\hat{s}_i \in \{U = \text{up},\ D = \text{down},\ S = \text{straight}\}$ is the slope symbol. The algorithm uses linear regression to compute the slope. Testing was performed on a particular streaming multivariate time series to assign a label to each multivariate segment using classification tasks.
Extended Trend-Value Approximation (1d-SAX) [20]: It is a compact binary representation to improve the retrieval performance using the same quantity of information as SAX. Unlike TVA, it works with alphabets of different sizes:

$$1d - SAX(P, m, \alpha^v, \alpha^s) = \{(\hat{v}_1, \hat{s}_1), \cdots, (\hat{v}_m, \hat{s}_m)\}, \tag{3}$$

where $\hat{v}_i$ is the average value symbol from an alphabet of size α^v and $\hat{s}_i \in$ is the slope symbol from an alphabet of size α^s. In addition, the authors propose a lookup table with all distances between two different symbols to speed-up the approximate search on large databases. Testing was performed on seven univariate time series collections (UCR Archive) using classification tasks.

2 Multi-resolution Trend-Value Approximation

2.1 Motivation

Why a Trend-Based Representation? Esmael et al. claims that “using only the value approximation, causes a high possibility to miss some important patterns in some time series data. SAX does not pay enough attention to the shapes of the time subsequences and may produce similar strings for completely different time series” [6]. Figure 2 shows an example of this statement. Here, we use an agglomerative hierarchical clustering to group five time series in three different classes. The time series is split into four segments. SAX only takes the mean value while TVA considers the slope obtaining a better match between time series 2 and 3, which belong to the same class.

TVA and 1d-SAX are similar techniques that compress each trend-value pair of the time series by a quantization process. In this work, we extend the ability of the local trends to various levels of resolution. While the granularity parameter (number of segments) of the piecewise approximation like TVA and 1d-SAX produces a horizontal segmentation, we propose a hierarchical segmentation induced by the resolution level. We will discuss how segmentation provides greater advantages in design and optimization and use both representations, symbolic and numeric, to evaluate our proposed method. Moreover, the numeric representation serves us to further reduce the reconstruction error and obtain better results in information retrieval tasks.

Our time series representation technique is called *Multi-resolution Trend-Value Approximation* (MTVA). The technique generates trend-value pairs on each level of resolution of the time series, and then computes the similarity between two MTVA representations using

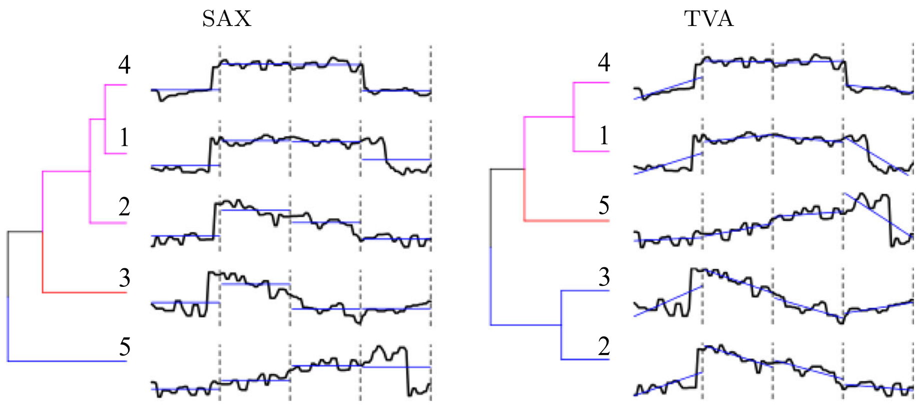


Fig. 2 A comparison of the ability of two time series representations to cluster five members of the CBF dataset [4] using the Euclidean distance

Table 1 Notation used in this paper

$P = \{p_1, \ldots, p_n\}$	A time series of length n (raw representation)
$P = \{(v_1, s_1), ..., (v_m, s_m)\}$	Numeric MTVA representation of P
$\hat{P} = \{(\hat{v}_1, \hat{s}_1), ..., (\hat{v}_m, \hat{s}_m)\}$	Symbolic MTVA representation of P
$p_i = (v_i, s_i)$	Trend-Value pair in the *ith* segment of P
$L \leq \lfloor \log_2(n/2) \rfloor + 1$	Level of resolution for P
$m = 2^L - 1$	Total number of trend-value pairs for P
$M_l = 2^{l-1}$	Number of segments in the level $l \in \{1, \ldots, L\}$

a distance measure. In addition, we design a symbolic representation to index a time series collection. Table 1 summarizes the notation used in this and subsequent sections.

2.2 Bottom-Up Construction Algorithm

Given the times series $P = \{p_1, \ldots, p_n\}$ and L as the level of resolution defined by the user, the MTVA representation of P is built following the next steps:

1. We start in the last resolution level L dividing the time series into $M = 2^{L-1}$ segments of size $w = n/M$.
2. Let $Y = \{y_1, \ldots, y_w\}$ be a segment of P in the time segment $X = \{x_1, \ldots, x_w\}$. We compute the linear regression on each segment by the function $lr(x) = ax + b$, where:
   - $a = \frac{\sum_{i=1}^{w}(x_j - \bar{X}) * y_j}{\sum_{i=1}^{w}(x_j - \bar{X})^2}$
   - $b = \bar{Y} - a * \bar{X}$
   - $\bar{X}$ and $\bar{Y}$ are the average value of X and Y, respectively.
   - The trend-value pair (v, s) of the segment Y is defined by:
     - $v = a * \frac{x_1 + x_w}{2} + b$ is the mean value.
     - $s = \arctan(a)$ is the slope,

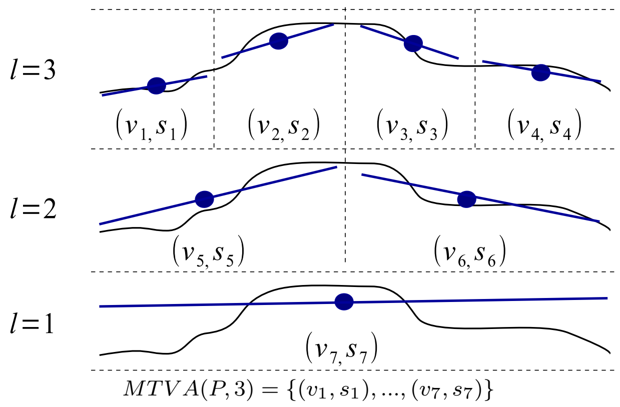


Fig. 3 Construction of the Multi-resolution Trend-Value Approximation

3. For the next resolution levels $M = 2^{\{L-2,L-3,...,0\}}$, compute the trend-value pair (v, s) for each segment as follows:
   - $v = \frac{v_i + v_{i+1}}{2}$
   - $s = \arctan\left(\frac{v_{i+1} - v_i}{x_{i+1} - x_i}\right)$.
   - v_i and x_i are, respectively, the mean value and the average time associated with a segment in the upper level (see Fig. 3).
4. The MTVA representation is an array of all the trend-value pairs: $MTVA(P, L) = \{(v_1, s_1), ..., (v_m, s_m)\}$.

Figure 3 shows the MTVA representation of the time series P up the third level of resolution ($L = 3$). Parameter L can be automatically computed so that the spatial complexity of the MTVA representation does not exceed the space of the original time series, that is, adjusting the total number of segments $m \leq n/2$. Alternatively, m can be defined in terms of the level of resolution $m = 2^L - 1$. Solving both equations, we conclude that the level of resolution for P is:

$$L_{max} = \lfloor \log_2(n/2) \rfloor + 1. \tag{4}$$

2.3 Distances Measures

First, we need cost functions to measure the distance between trend-value pairs. Given two pairs p_i and q_j, we define three cost functions:

- $cost_1(p_i, q_j) = |v_i^p - v_j^q| + |s_i^p - s_j^q|$
- $cost_2(p_i, q_j) = |v_i^p - v_j^q|^2 + |s_i^p - s_j^q|^2$
- $cost_{dot}(p_i, q_j) = (1 + |v_i^p - v_j^q|) * (1 + |s_i^p - s_j^q|) - 1$

For using $cost_1$ and $cost_2$, both value-domain and slope-domain should have similar ranges to avoid that the distance is governed by only one of them. The slope ranges are between $-\frac{\pi}{2}$ and $+\frac{\pi}{2}$. We therefore normalize the time series by a standard normalization procedure (e.g., Z-distribution). The cost function $cost_{dot}$ can also be used on time series that need to retain their value-domain.

Algorithm 1 *MDist*: Multi-resolution Distance with Anticipatory Pruning

```
Require: (Two MTVA representation $P$ and $Q$, Threshold $th$)
1: $L_{max}$ = the maximum common resolution of $P$ and $Q$
2: $d = 0$
3: for $l = 1$ to $L_{max}$ do
4:     $a = 2^{(l-1)}$
5:     $b = 2^l - 1$
6:     $d = d + Dist(P[a \cdots b], Q[a \cdots b])$
7:     if $d > th$ then                              ▷ Apply pruning
8:         Break
9:     end if
10: end for
11: return $d$
```

Second, we adapt two common techniques for measuring the distance between two sets of trend-value pairs:

– Linear Distance:

$$Dist(P, Q) = \sum_{i=1}^{M} cost(p_i, q_i). \tag{5}$$

– Dynamic Time Warping Distance: It is computed building a cumulative cost matrix $C \in \mathbb{R}^{M \times M}$ in the following way:

$$\begin{gathered} C[0,0] = 0 \quad C[i,0] = \infty \quad C[0,j] = \infty \\ C[i,j] = cost(p_i, q_j) + min \begin{cases} C[i-1, j] \\ C[i, j-1] \\ C[i-1, j-1] \end{cases} \\ \Rightarrow Dist(P, Q) = C[M, M]. \end{gathered}$$

Finally, our multi-resolution distance *MDist* measures the dissimilarity between two MTVA representations executing $Dist$ on all levels of resolution:

$$MDist(P, Q) = \sum_{l=1}^{L} Dist(P[a \dots b], Q[a \dots b]), \quad \text{where } a = 2^{l-1} \text{ and } b = 2^l - 1. \tag{6}$$

Algorithm 1 shows the pseudocode of *MDist*. Its time complexity using the Linear Distance corresponds to the sum of the time for each level of resolution:

$$T(L) = \sum_{l=1}^{L} M_l = \sum_{l=1}^{L} 2^{l-1} = 2^L - 1. \tag{7}$$

If we compute the distance in the worst case where L is exactly $\log_2(n/2) + 1$, the time complexity is $O(n)$, where n is the length of the original time series. For the DTW Distance, the time complexity is expressed as follows:

$$T(L) = \sum_{l=1}^{L} M_l^2 = \sum_{l=1}^{L} (2^{l-1})^2 = \frac{4^L - 1}{3}, \tag{8}$$

where in the worst case it is $O(n^2)$. Therefore, *MDist* in the worst case is theoretically as fast as the classic distances that work over raw time series.

To reduce the number of distance computations, we design an optimization strategy that applies a threshold th to break out of the loop when the partial distance is greater than th, thus avoiding having to compute the rest of levels (see Algorithm 1, lines 7–9). This threshold allows us to efficiently compute the similarity between two MTVA representations reducing the number of calls to $Dist$. When the algorithm receives two inputs of different resolution, it takes the maximum resolution level that both have in common (line 1). A practical application of the pruning is in time series classification when we use the Nearest Neighbor Search (see Algorithm 2, line 14). This optimization follows the spirit of the Anticipatory DTW [1], where the authors apply a threshold into the DTW distance to filter out comparisons.

2.4 Symbolic Representation

By using discretization techniques, we transform the numeric representation into a sequence of symbols. This symbolic representation provides us greater ease of interpretation and simplicity to manage time series collections.

Definition 1 According to Lin et al., "*Breakpoints* are a sorted list of numbers $\beta = \{\beta_1, \ldots, \beta_{\alpha-1}\}$, such that the area under a $N(0, 1)$ Gaussian curve from β_i to $\beta_{i+1} = 1/\alpha$ (β_0 and β_α area defined as $-\infty$ and $+\infty$, respectively)" [16]. For example, if $\alpha = 4$, the breakpoints are $\{\beta_1 = -0.67, \beta_2 = 0, \beta_3 = +0.67\}$.

2.4.1 Gaussian Assumption

To transform the numeric pair $p_i = (v_i, s_i)$ to a symbolic pair $\hat{p}_i = (\hat{v}_i, \hat{s}_i)$, we quantize both values separately using breakpoints that produce equal-size areas under the Gaussian curve $N(\mu, \sigma^2)$ (similar to 1d-SAX, see Fig. 1). Gaussian discretization is feasible for normalized time series, since statistically the mean value and the slope have a Gaussian distribution [19,20]. As in 1d-SAX, the breakpoints are determined by the curve $N(0, 1)$ for the mean value and $N(0, \sigma_L^2)$ for the slope. In this last case, we use the variance σ_L^2 in terms of the level of resolution L because each level of resolution generates different slope distributions (see Fig. 4), unlike the 1d-SAX that uses a slope variance in terms of the size of the segment. Additionally, to apply the linear regression between X and Y, we recommend that both variables have a similar range. If the time series is normalized in $N(0, 1)$, the temporal component X must fit in this interval size. In this work, we normalize the length of each segment $X = [1, w] \rightarrow X = [-1, 1]$. Thus, the variance σ_L^2 is only in terms of the level of resolution and independent of the size of the time series.

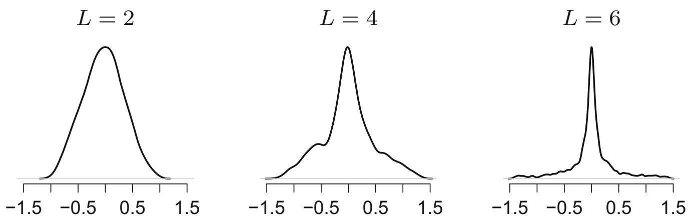


Fig. 4 Density of the slope varying the level of resolution in ECG time series

2.4.2 Alphabet Size

The alphabet size is delimited by the number of breakpoints (Definition 1) and strongly influences the compression ratio and the reconstruction error. To quantize the trend-value pair, we need two alphabets with size α_v and α_s for the mean value and the slope respectively. We use binary symbols where α is a power of two [23]. For example, to compress the numeric MTVA representation up the level 3 using $\alpha_v = 4$ and $\alpha_s = 4$, we need $(2 + 2) * (2^3 - 1)$ bits, which is less than 4 bytes for time series. The symbolic MTVA representation is useful for different applications like indexing and anomaly detection.

2.5 Indexing

We propose a simple bucket index to efficiently manage MTVA time series datasets (see Fig. 5). We use the symbolic representation to build a hash table, where each bucket $\hat{P}$ envelops a set of similar MTVA time series. Moreover, we design a lower bounding function called LB_MDist to measure the distance between the query object Q and a bucket $\hat{P}$, so that it is lower or equal than the real distance between Q and any object $P \in \hat{P}$.

Before defining *LB_MDist*, we first need to define the lower bounding function of the trend-value cost, which is denoted as follows:

$$LB\_cost(\hat{p}_i, q_i) \le cost(p_i, q_i). \tag{9}$$

The symbol $\hat{p}_i$ derives from a trend-value pair p_i that is located between two breakpoints $\beta_{Ui} < p_i \le \beta_{Li}$, independently for each pair value (see Fig. 6). Either of these equations then computes LB_cost:

- $LB\_cost_1(\hat{p}_i, q_i) = \Delta v + \Delta s$
- $LB\_cost_2(\hat{p}_i, q_i) = (\Delta v)^2 + (\Delta s)^2$
- $LB\_cost_{dot}(\hat{p}_i, q_i) = (1 + \Delta v) * (1 + \Delta s) - 1$

where

$$\Delta v = \left\{ \begin{array}{cl} |v_i^q - \beta_{Ui}| & v_i^q > \beta_{Ui} \\ |\beta_{Li} - v_i^q| & v_i^q < \beta_{Li} \\ 0 & \text{else,} \end{array} \right. \quad : \beta_{Li} \le v_i^p < \beta_{Ui},$$

$$\Delta s = \left\{ \begin{array}{cl} |s_i^q - \beta_{Ui}| & s_i^q > \beta_{Ui} \\ |\beta_{Li} - s_i^q| & s_i^q < \beta_{Li} \\ 0 & \text{else,} \end{array} \right. \quad : \beta_{Li} \le s_i^p < \beta_{Ui}.$$

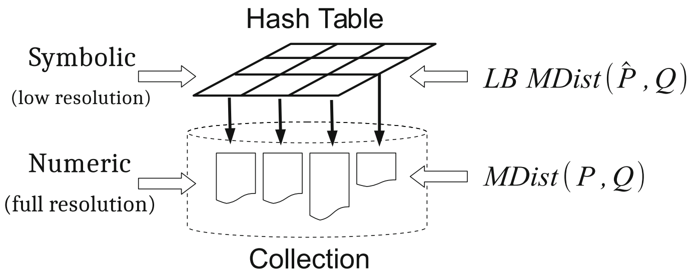


Fig. 5 Index model for the MTVA representation

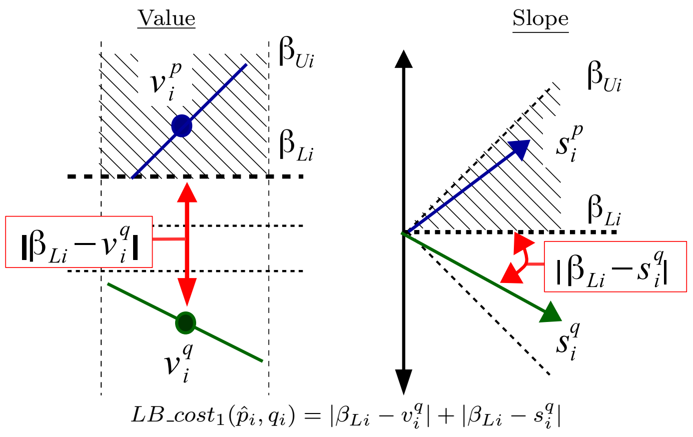


Fig. 6 Lower bounding trend-value cost, the blue line represents a trend-value pair stored in our database and the green line is the query. (Color figure online)

Equation 10 describes the lower bounding function for the multi-resolution linear distance:

$$LB\_MDist(\hat{P}, Q) = \sum_{l=1}^{IL} \sum_{i=2^{(l-1)}}^{2^l-1} LB\_cost(\hat{p}_i, q_i). \tag{10}$$

The Indexing Level $IL \leq L$ is the level of resolution with which we build the hash table. IL determines the amount of bits for the index structure.

For similarity searches, on the one hand, we can use LB_MDist to find approximated buckets in which the objects are most similar to a query object Q. On the other hand, we can use LB_MDist to efficiently search the nearest neighbor of Q (see Algorithm 2). We end up with the same result as applying a linear scan over the numeric MTVA time series. This algorithm orders the buckets in such a way that those closest to Q, in terms of LB_MDist, are visited first (lines 2–5). Afterwards, we refine the candidates objects, applying a post-processing search with the anticipatory pruning distance $MDist$ (lines 12–19) that receives as threshold the best match distance so far. The algorithm finishes immediately when the current visited bucket has a lower bounding distance greater than the best match distance (lines 9–11).

3 Multi-resolution Discord Discovery

Keogh et al. [14] introduced an unsupervised method—called discord discovery—for finding the most unusual subsequence (the one with the largest distance from its nearest non-self match) in a streaming time series. The basic algorithm has a computational complexity of $O(N^2)$, where N is the total number of subsequences. Therefore, the main challenge of the discord discovery techniques is to face this quadratic complexity.

In this sense, our MTVA representation together with the proposed efficient search techniques and the discord discovery heuristics can be used to solve the anomaly detection on normalized subsequences. This is called HOT MTVA. However, we observed that the index method designed above uses only one hash table for the whole SAX space (similar to HOT

Algorithm 2 Nearest Neighbor Search for the MTVA Index

```
Require: (Index $\mathbb{R}$ , Query $Q$)
1: $list = \emptyset$
2: for $\hat{P} \in \mathbb{R}$ do
3:     $d_i = LB\_MDist(\hat{P}, Q)$
4:     $list.add([\hat{P}, d_i])$
5: end for
6: $sorted\_list = argsort(list)$
7: $best\_so\_far = \infty$;
8: for $[\hat{P}, min\_d] \in sorted\_list$ do
9:     if $min\_d > best\_so\_far$ then
10:        Break
11:    end if
12:    $objects = readBucket(\hat{P})$
13:    for $P \in objects$ do
14:        $d = MDist(P, Q, best\_so\_far)$                    ▷ Alg. 1
15:        if $d < best\_so\_far$ then
16:            $best\_so\_far = d$
17:            $best\_match\_object = P$;
18:        end if
19:    end for
20: end for
21: return ($best\_match\_object$);
```

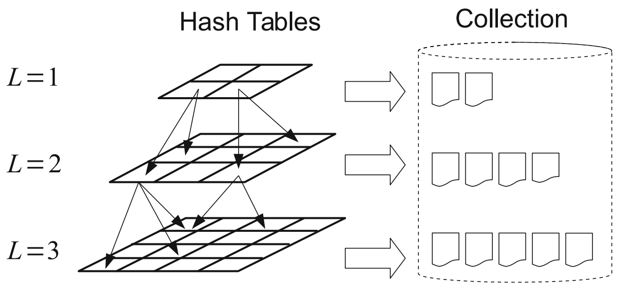


Fig. 7 Multi-resolution Index Model for the MTVA representation

SAX), and this has a disadvantage: the size of the hash table grows considerably when using a higher resolution level. This may result in a high searching cost, which could be as expensive as applying a linear scan over the data. Therefore, we propose a multi-resolution method which increases the level resolution when a hash-bucket is overflowed (see Fig. 7). Such an indexing structure is the perfect fit for our MTVA representation. Moreover, it allows one to control the level of resolution of the detected anomaly.

3.1 Building Algorithm

Let $P = \{p_1, \ldots, p_n\}$ be a time series, and let w be the size of the sliding window for extracting all subsequences. Algorithm 3 describes the insertion procedure of a MTVA subsequence C_p into the indexing structure $\mathbb{R}$. Unlike HOT iSAX, we apply hierarchical quantization (line 2) to access the hash tables (where each slot is a node) from the lowest resolution to the maximum resolution. If a terminal node is full, we re-insert all its associated objects into a hash table of higher level to provide additional differentiation (lines 10–16), so we create new nodes with the next resolution level of the current node (lines 20–22). We use a size

Algorithm 3 Insert Objects into the Multi-resolution Index

```
Require: (Object $C_p$)
1: let $\mathbb{R}$ be the multi-resolution index
2: $str = generateSAX(C_p, this.level)$
3: if $\mathbb{R}.containsKey(str)$ then
4:     $node = \mathbb{R}.getNode(str)$          ▷ get the child node associated with the key $str$
5:     if $R$ is internal then
6:         $node.insert(C_p)$
7:     else if $node$ is terminal then
8:         $writeObject(str, C_p)$
9:         if $R$ is full then
10:            $objects = readBucket(str)$          ▷ retrieval all the objects associated with $str$
11:            $new\_node = newInternalNode(node.level)$
12:            for $C_i \in objects$ do
13:                $new\_R.insert(C_i)$
14:            end for
15:            $\mathbb{R}.remove(str, node)$
16:            $\mathbb{R}.add(str, new\_node)$
17:        end if
18:    end if
19: else
20:    $new\_node = newTerminalNode(this.level + 1)$          ▷ increase the resolution level
21:    $\mathbb{R}.add(str, new\_node)$
22:    $writeObject(str, C_p)$
23: end if
```

threshold th_{max} to control the maximum number of objects in a terminal node (the so-called bucket). It follows that the indexing level has dynamic behavior, as its incremental value depends on the size of the dataset and the maximum level of resolution (L_{max}).

3.2 Discord Discovery Heuristics

The discordant subsequence is found by applying the optimal discord discovery procedure [14] using the following heuristics:

Outer Loop Heuristic We first visit all subsequences belonging to the bucket that contains the minimum number of subsequences starting from the lowest resolution level ($L = 1$). Afterwards, we visit the rest of buckets in random order. This heuristic ensures that the subsequences that are most isolated, at each resolution level, will be visited at the beginning of the search as potential candidates.

Inner Loop Heuristic We then use an inner loop to search the best non-self match of each selected candidate C_q. We first visit all subsequences contained in the bucket from which C_q is retrieved. Afterwards, we apply the nearest non-self match search algorithm (NNM-Search) to visit the rest of the buckets. This heuristic allows us to first visit all the subsequences most similar to C_q, increasing the probability of an early termination of the loop.

The NNM-Search algorithm (see Algorithm 4) performs a hierarchical search across the internal nodes using a stack to maintain the nodes ordered by MINDIST (lines 10–17). This MINDIST is a lower bounding function of the MTVA distance, which measures the minimum distance between the query and the current node. MINDIST is calculated as:

$$MINDIST(\hat{P}, Q, l, \alpha) = \alpha + \sum_{i=2^{(l-1)}}^{2^l-1} LB\_cost(\hat{p}_i, q_i), \tag{11}$$

Algorithm 4 NNM-Search for the Multi-resolution MTVA Index

```
Require: (Index $\mathbb{R}$, Query $C_q$, Window Size $w$, Threshold Distance $th_{dist}$)
1: $stack.push([\mathbb{R}.getNodeRoot(), 0])$
2: $best\_dist = \infty$
3: $best\_post = -1$
4: while $stack \neq \emptyset$ do                                    ▷ inner loop
5:     $[node, min\_d] = stack.pop()$
6:     if $min\_d > best\_dist$ then
7:         Break                                                   ▷ break out of inner loop
8:     else if $node$ is internal then
9:         $list = \emptyset$
10:        for $child\_node \in node.children$ do
11:            if $child\_node$ was not visited then
12:                $d = MINDIST(child\_node.str, C_q, node.level, min\_d)$
13:                $list.add([child\_node, d])$
14:            end if
15:        end for
16:        $sorted\_list = argsort(list)$
17:        $stack.push(sorted\_list)$
18:    else if $node$ is terminal then
19:        $objects = readBucket(node.str)$
20:        for $C_p \in objects$ do
21:            if $|p - q| \geq w$ then                               ▷ non-self match?
22:                $d = MDist(C_p, C_q, best\_dist)$                  ▷ Alg. 1
23:                if $d < best\_dist$ then
24:                    $best\_dist = d$
25:                    $best\_post = p$
26:                end if
27:                if $d < th_{dist}$ then
28:                    Break                                       ▷ break out of inner loop
29:                end if
30:            end if
31:        end for
32:    end if
33: end while
34: Return $(best\_dist, best\_pos)$
```

where l is the current resolution level and α is the accumulated distance of the previous levels. The algorithm also applies two breaking statements to break the inner loop as early as possible: one corresponds to MINDIST (line 6) and the other one to the best-so-far discord distance (line 27).

4 Experimental Results

In this section, we evaluate the performance of our approach for classification and anomaly detection in different datasets. An Intel Core i7 3.4GHz with 8GB RAM is used for conducting all our experiments. All algorithms are implemented in C++.

4.1 Classification

For this experimental evaluation, we use 44 univariate time series collections from the UCR Archive (see Table 2). We first evaluate the quality of our proposed representation with other state-of-the-art techniques for representing time series. We next compare our approach with two classic distances that work over the raw representation. The nearest neighbor search (Algorithm 2) gives us the classification results.

4.1.1 Trend-Value Approximation Performance

We evaluate the performance of the numeric representations based on trends and mean values (TVA and MTVA) with respect to the representation based only on average values (PAA). To get the best performance, we use the best parameter tuning in each time series collection. In this sense, PAA and TVA require tuning the total number of segments in terms of the length of the time series $m = factor * n$, where the best value for *factor* is experimentally selected from the set $\{0.1, 0.15, 0.2, \ldots, 0.5\}$. MTVA requires tuning the level of resolution L from the set $\{2, 3, \ldots, L_{max}\}$. Table 3 shows the results obtained by the linear distance. Here, we calculate the average classification error, the win-tie-lose results of the numeric TVA and MTVA representations compared to PAA representation, and the p value (Wilcoxon signed ranks test [25], confidence level = 0.95). We mark a tie between two classification results when their error difference is lower than 0.001. We observe that our multi-resolution trend-value representation performs better than the PAA and TVA representations. Moreover, our technique required less number of segments than PAA, although we need twice the memory space for the numeric trend-value pairs.

Furthermore, we compare our multi-resolution representation with other state-of-the-art techniques (see Fig. 8). First, the numeric MTVA is compared with the Multi-resolution PAA (MPAA [18]) which is similar to the Discrete Haar Wavelet Transform (DWT [11]). Second, the symbolic MTVA is compared with other symbolic representations like SAX and 1d-SAX. For the latter, we use the same quantity of information for generating the same compression ratio in each dataset (Table 4). As the breakpoints depend on the slope distribution $N(0, \sigma^2)$, we empirically set the slope variance as $\sigma^2 = \frac{2}{\sqrt{m}}$ and $\sigma^2 = \frac{2}{L}$ for the 1d-SAX and the MTVA respectively, which obtained the best trade-off in most datasets. After setting the parameters, we compute the classification error using the Euclidean distance ($cost_2$ for the MTVA) over eight time series collections that have at least eight levels of resolution. We observe that MTVA reaches the smallest error sooner than MPAA. Moreover, the error obtained by both MTVA and 1d-SAX is stabilized from level $L \geq 4$. However, the error obtained by SAX shows a monotonically decreasing trend while the size of word gets bigger, and it obtains better performance than our method from level $L \geq 7$.

An important characteristic of our multi-resolution representation is that the segments always have the same size in each resolution level defined by L. In this way, for each time series length, we can generate all trend-value pairs of the maximum resolution level and store them in secondary memory. Afterwards, we just take out those that are requested by the user. This characteristic provides us with greater flexibility for testing our index model with different index levels without the need to recalculate the MTVA representation. In contrast, the segments generated by the piecewise approximations (like PAA, TVA, SAX and 1d-SAX) are of different sizes in each resolution level defined by m. Thus, they always need to recalculate the representation when one varies the value of m.

4.1.2 MTVA Distance Performance

Table 5 shows the results obtained by our multi-resolution distance $MDist$, using the linear distance and the dynamic time warping as base distance. For both cases, we calculate the average classification error, the win-tie-lose results of the numeric MTVA approximation

Table 2 UCR Time Series Archive: data collections used for our assessments

	# of classes	Dataset size		Length	Basic distances	
		Train	Test		ED	cDTW (r)
50Words	50	450	455	270	0.369	0.242 (6)
Adiac	37	390	391	176	0.389	0.391 (3)
Beef	5	30	30	470	0.467	0.467 (0)
CBF	3	30	900	128	0.148	0.004 (11)
ChlorineConcentration	3	467	3840	166	0.350	0.350 (0)
CinC_ECG_torso	4	40	1380	1639	0.103	0.070 (1)
Coffee	2	28	28	286	0.250	0.179 (3)
Cricket_X	12	390	390	300	0.426	0.236 (7)
Cricket_Y	12	390	390	300	0.356	0.197 (17)
Cricket_Z	12	390	390	300	0.380	0.180 (7)
DiatomSizeReduction	4	16	306	345	0.065	0.065 (0)
ECG	2	100	100	96	0.120	0.120 (0)
ECGFiveDays	2	23	861	136	0.203	0.203 (0)
Face (all)	14	560	1690	131	0.286	0.192 (3)
Face (four)	4	24	88	350	0.216	0.114 (2)
FacesUCR	14	200	2050	131	0.231	0.088 (12)
Fish (readme)	7	175	175	463	0.217	0.160 (4)
Gun-Point	2	50	150	150	0.087	0.087 (0)
Haptics	5	155	308	1092	0.630	0.588 (2)
InlineSkate	7	100	550	1882	0.658	0.613 (14)
ItalyPowerDemand	2	67	1029	24	0.045	0.045 (0)
Lightning-2	2	60	61	637	0.246	0.131 (6)
Lightning-7	7	70	73	319	0.425	0.288 (5)
MALLAT	8	55	2345	1024	0.086	0.086 (0)
MedicalImages	10	381	760	99	0.316	0.253 (20)
MoteStrain	2	20	1252	84	0.121	0.134 (1)
NonInvasiveFetalECG_Thorax1	42	1800	1965	750	0.171	0.185 (1)
NonInvasiveFetalECG_Thorax2	42	1800	1965	750	0.120	0.129 (1)
OliveOil	4	30	30	570	0.133	0.167 (1)
OSU Leaf	6	200	242	427	0.483	0.384 (7)
SonyAIBORobotSurface	2	20	601	70	0.305	0.305 (0)
SonyAIBORobotSurfaceII	2	27	953	65	0.141	0.141 (0)
Swedish Leaf	15	500	625	128	0.213	0.157 (2)
Symbols	6	25	995	398	0.100	0.062 (8)
SyntheticControl	6	300	300	60	0.120	0.017 (6)
Trace	4	100	100	275	0.240	0.010 (3)
TwoPatterns	4	1000	4000	128	0.090	0.002 (4)
TwoLeadECG	2	23	1139	82	0.253	0.132 (5)
uWaveGestureLibrary_X	8	896	3582	315	0.261	0.227 (4)
uWaveGestureLibrary_Y	8	896	3582	315	0.338	0.301 (4)
uWaveGestureLibrary_Z	8	896	3582	315	0.350	0.322 (6)

Table 2 continued

	# of classes	Dataset size		Length	Basic distances	
		Train	Test		ED	cDTW (r)
Wafer	2	1000	6174	152	0.005	0.005 (1)
WordsSynonyms	25	267	638	270	0.382	0.252 (8)
Yoga	2	300	3000	426	0.170	0.155 (2)

Table 3 Classification error using the best configuration of three numeric representations

Metrics	PAA	TVA	MTVA		
	ED-norm		$cost_1$	$cost_2$	$cost_{dot}$
Av. error	0.226	0.225	0.217	0.227	0.218
win/tie/lose	$\hookrightarrow$	12/9/23	26/4/14	19/6/19	25/5/14
p value	$\hookrightarrow$	0.291	0.053	0.814	0.059
# Segments	23%	21%	21%	19%	19%

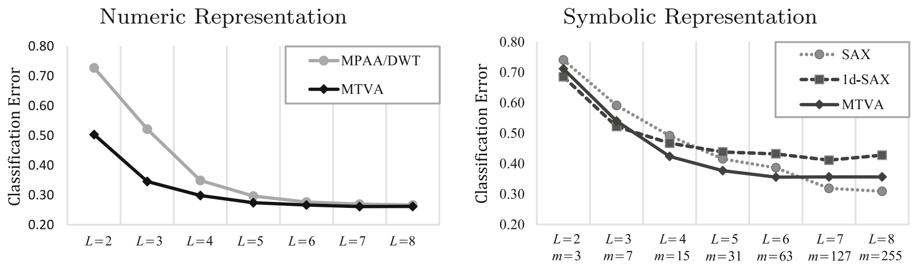


Fig. 8 Comparison of our multi-resolution representation with other state-of-the-art techniques

Table 4 Equalizing the quantitative information

Symbolic representation	Alphabet size		Word size
	α_v	α_s	m
SAX	8	–	$2^L - 1$
1d-SAX	4	4	
MTVA	4	4	

compared with the raw representation, and the p value. Moreover, we performed two tests: a first test using the maximum level of resolution for each dataset (Eq. 4), and a second test using the best level of resolution for each dataset.

We obtain a smaller classification error when we use MTVA, mainly in two of the proposed trend-value costs: $cost_1$ and $cost_{dot}$. Using linear distance, we note that the obtained decrease in classification error is statistically significant (p value < 0.05). Using the DTW distance, our multi-resolution representation has a non-significant difference regarding the raw representation (p value > 0.05). However, we observed that MTVA was particularly effective in half of the datasets, which means that our approach performs better than the classic distance depending on the data collection.

Table 5 Classification error using raw representation and MTVA representation

Metrics	Raw	MTVA		
		$cost_1$	$cost_2$	$cost_{dot}$
Linear distance				
	ED [a]	Maximum level of resolution		
Average error	0.251	0.226	0.236	0.229
Win/tie/lose	$\hookrightarrow$	32/1/11	20/11/13	32/2/10
p value	$\hookrightarrow$	0.001	0.160	0.002
	ED-norm [b]	The best level of resolution		
Average error	0.245	0.217	0.227	0.218
Win/tie/lose	$\hookrightarrow$	31/4/9	28/9/7	33/3/8
p value	$\hookrightarrow$	0.000	0.001	0.000
Dynamic time warping				
	DTW	Maximum level of resolution		
Average Error	0.212	0.194	0.205	0.195
Win/tie/lose	$\hookrightarrow$	25/2/17	22/3/19	25/1/18
p value	$\hookrightarrow$	0.050	0.299	0.059
	cDTW [c]	The best level of resolution		
Average error	0.192	0.190	0.197	0.192
Win/tie/lose	$\hookrightarrow$	21/6/17	21/2/21	21/5/18
p value	$\hookrightarrow$	0.695	0.357	0.804

[a]Euclidean Distance [4]
[b]Z-normalized Euclidean Distance [24,27]
[c]The best configuration for the constrained DTW [4]

As noted above, the efficiency is the most remarkable characteristic of using time series representation. To show the computational advantage of our approach, we select the largest dataset that contains time series of length 750, where the maximum level of resolution is $L_{max} = 8$. Figure 9 shows the improved efficiency of $MDist$ (using the linear distance) compared with the classic Euclidean Distance in terms of computed cost functions increasing the level of resolution. We first perform a linear scan using the $MDist$ according to Eq. 6. Afterwards, we apply the anticipatory pruning strategy (Algorithm 1). We note the wide lead of using the anticipatory pruning, which allows us to save about one order of magnitude in computed cost functions in the best case ($L = 8$). This efficiency advantage is due to multi-resolution properties described in Sect. 2.3. We finally evaluate our simple bucket index, where the search algorithm also employs the pruning property (Algorithm 2). We note that our indexing method outperforms the linear scan up to two orders of magnitude, thus being the most efficient option for retrieving TVA time series.

We conclude with an important fact about our multi-resolution representation: the best level of resolution was obtained in the interval $L = \{4, 5, 6\}$ for 38 of 44 time series collections. This will allow us to limit the range of possible values for L in new time series collections.

4.2 Anomaly Detection

In this experiment, we address the anomaly detection problem using the discord discovery approach together with our proposed methods. Effectiveness will be evaluated over a set of

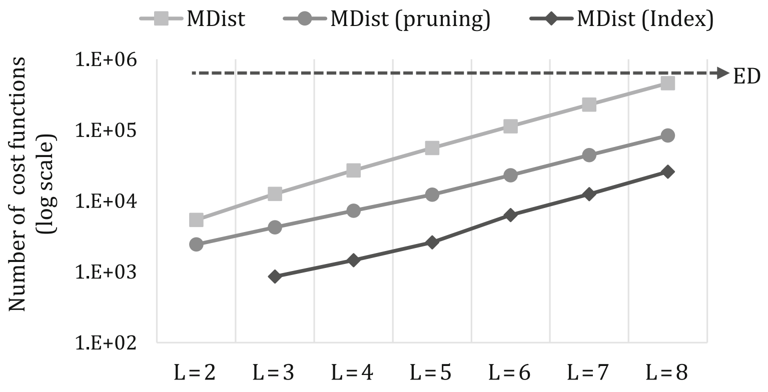


Fig. 9 Efficiency improvement by the MTVA Distance regarding the classic ED that uses raw representation

Table 6 Datasets for discord discovery evaluation

	File	Variables	Length	Window	Anomalous Region	
					Begin	End
1	stdb_308_0	2	5400	308	2364	2504
2	mitdbx_mitdbx_108	2	16,000	600	10,827	11,243
3	xmitdb_x108_0	2	5250	108	3932	4039
4	mitdb_100_180	2	5400	180	1826	2001
5	chfdb_chf01_275	2	3750	200	2345	2551
6	chfdb_chf13_45590	2	3750	200	2809	2950
7	ltstdb_20221_43	2	3700	200	672	746
8	lltstdb_20321_240	2	3600	200	668	732
9	qtdbsele0606 (1)	1	2500	40	1130	1175
10	qtdbsele0606 (2)	1	2500	150	1109	1226
11	chfdbchf15	2	15,000	256	2253	2392
12	qtdbsel102	2	10,000	200	4234	4377
13	ann_gun_CentroidA	2	5500	150	1585	2089
14	e_nogunCentroidA	2	6325	150	5389	5648
15	nprs43	1	18,000	350	14,835	14,976
16	nprs44	1	6500	350	4950	5195
17	TEK14	1	5000	128	1114	1168
18	TEK16	1	5000	128	4266	4358
19	TEK17	1	5000	128	2103	2199
20	power_data	1	35,000	750	11,380	12,130

Data domains: Electrocardiograms (rows: 1–12), video surveillance tracking (rows 13–14), space shuttle valve sensor (rows: 17–19), patient's respiration sensor (rows 15–16) and power demand measurement (row 20)

Table 7 Percentage of true detections for trend-value representations

ε	ED (%)	ED-TVA			ED-MTVA		
		$m = 3$ (%)	$m = 7$ (%)	$m = 15$ (%)	$L = 2$ (%)	$L = 3$ (%)	$L = 4$ (%)
0.025	60	63	73	63	60	67	70
0.050	77	77	90	83	70	83	87
0.075	73	73	90	83	70	80	83
0.100	80	80	90	87	73	83	87
0.125	83	80	90	80	77	83	87
0.150	77	70	80	80	73	77	80

real cases of anomalous time series collected by Keogh et al. [13] detailed in Table 6. An expert in the field (application domain) annotated the anomalous region. Also, the window size is different for each time series. To evaluate the efficiency of our approach on static time series, we use the datasets ECG, EEG, ERP, Koski, Random Walk and Packet from the UCR Time Series Archive [4]. We also use the "Time Series for Weather Data" from the National Oceanic and Atmospheric Administration in the USA (available at www.esrl.noaa.gov/psd/boulder/). For each dataset, we randomly extract time series of lengths 1k, 2k, 4k, 8k, 16k and 32k. We empirically set the maximum number of elements in a bucket as $th_{max} = 50$.

We first evaluate the accuracy of the two trend-value numeric representations, TVA and MTVA, over all anomaly cases. An important feature of the trend-value representations is that the slope of the noisy segments tends to zero and thereby the unusualness of noisy subsequences is reduced. The classic techniques use the Euclidean Distance as measure distance over the raw representation of the normalized subsequence. For TVA and MTVA representations, we use the linear distance with the cost function $cost_2$. We use the same number of inputs in both methods and three different values of dimensional reduction. Table 7 shows that trend-value representations achieve a higher percentage of true detections ($ACC \geq 0.5$) than using the raw representation. In this way, we assert that our method is more robust to local noise. Furthermore, we can improve this percentage up to 100% of true detections finding the best value for ε for each of the time series. Additionally, we also note that TVA outperforms MTVA, but nevertheless we highlight the flexibility of MTVA for dynamically working in different levels of resolution at runtime.

Secondly, we accelerate the search using our indexing structures and compare it with the main state-of-the-art techniques like HOT_SAX [13] and HOT_iSAX [15]. We set the same quantitative information for each technique (see Table 4). We empirically set the indexing level as $IL = 3$, so the word size is set as $m = 2^{IL} - 1$. To measure the efficiency of the algorithms, we consider the number of computed distances (see Fig. 10). We observe that our MTVA techniques are much more efficient than HOT_iSAX in terms of computed distances. Moreover, we achieved a better performance in CPU runtime due to the dimensional reduction and the anticipatory pruning distance. Finally, Fig. 11 shows experimentally the benefits of using a multi-resolution index for MTVA representation instead of a simple bucket index. The multi-resolution index significantly reduces the number of buckets and computed MINDIST for the same resolution level.

Clearly, discord discovery using our multi-resolution trend-value representation returns better results. Our approach also provides flexibility with finding anomalies at different levels of granularity.

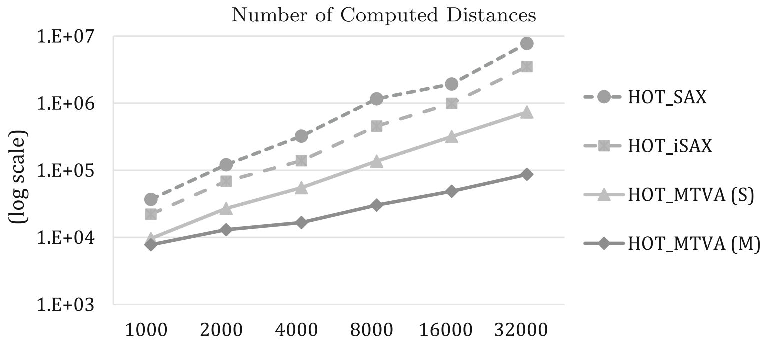


Fig. 10 Efficiency improvement of our multi-resolution method in Anomaly Detection

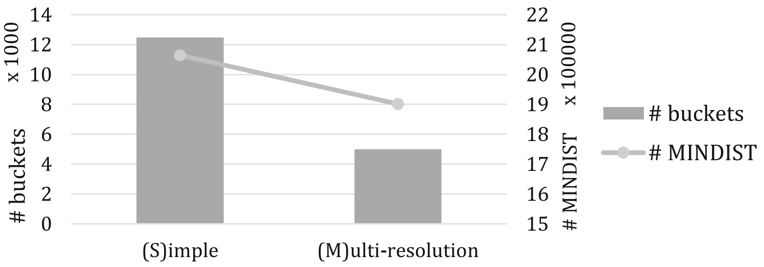


Fig. 11 Structural comparison of two indexing methods for MTVA representation

Table 8 Parameters to be tuned

Numeric		Symbolic	
PAA:	m	SAX:	m, α_v
TVA:	m	1d-SAX:	m, α_v, α_s
MTVA:	L	MTVA:	IL, α_v, α_s
MPAA:	L		

5 Conclusions

We proposed a multi-resolution time series representation (MTVA) which is composed of trend-value pairs obtained by applying regression linear in each resolution segment. Our parameter-free technique models the full resolution of time series, keeping the same spatial complexity of the raw representation. We also provided a (dis)similarity measure and its lower bounding function to perform efficient searches.

We demonstrated the utility of our proposed method in Classification Accuracy improving the performance of the classic techniques based on raw representation. Also, our numeric representation was more effective than the classic approximations: PAA, TVA and Multi-resolution PAA. We performed a comparison of three symbolic data models using the same quantitative information, where our symbolic MTVA slightly outperformed SAX and 1d-SAX. The efficiency of our method was tested in terms of computed cost functions, where our MTVA distance was two orders of magnitude faster than the classic Euclidean Distance.

MTVA also was evaluated in Anomaly Detection, where we highlighted the slope feature for mitigating the false unusualness of noisy subsequences. The efficiency of our multi-resolution discord discovery algorithm outperformed HOT_iSAX in terms of computed distances. One additional advantage of our multi-resolution representation is that the level of resolution was more intuitive and easier to fine-tune than the number of segments in each piecewise approximation technique.

One disadvantage of trend-value approximations is that they require twice the space per segment. Adding a parameter to represent the trend of the time series, it runs the risk of subtracting simplicity to our concise data model if it is compared with the SAX technique. Table 8 shows a summary of the parameters that need be tuned by each technique. Nevertheless, MTVA can be used as baseline for finding anomalies in different levels of granularity.

References

1. Assent I, Wichterich M, Krieger R, Kremer H, Seidl T (2009) Anticipatory DTW for efficient similarity search in time series databases. Proc VLDB Endow 2(1):826–837
2. Chakrabarti K, Keogh EJ, Mehrotra S, Pazzani MJ (2002) Locally adaptive dimensionality reduction for indexing large time series databases. ACM Trans Database Syst 27(2):188–228
3. Chen Q, Chen L, Lian X, Liu Y, Yu JX (2007) Indexable PLA for efficient similarity search. In: Proceedings of 33rd international conference on very large data bases, VLDB Endowment, pp 435–446
4. Chen Y, Keogh E, Hu B, Begum N, Bagnall A, Mueen A, Batista G (2015) The UCR time series classification archive. www.cs.ucr.edu/~eamonn/time_series_data/
5. Dan J, Shi W, Dong F, Hirota K (2013) Piecewise trend approximation: a ratio-based time series representation. Abst Appl Anal 2013:603629. https://doi.org/10.1155/2013/603629
6. Esmael B, Arnaout A, Fruhwirth RK, Thonhauser G (2012) Multivariate time series classification by combining trend-based and value-based approximations. In: Computational science and its applications—ICCSA 2012, LNCS 7336. Springer, New York, pp 392–403
7. Fuad MMM (2012) Differential evolution versus genetic algorithms: towards symbolic aggregate approximation of non-normalized time series. In: Proceedings of 16th international database engineering and applications symposium. ACM, pp 205–210
8. Gullo F, Ponti G, Tagarelli A, Greco S (2009) A time series representation model for accurate and fast similarity detection. Pattern Recognit 42(11):2998–3014
9. Keogh E, Kasetty S (2002) On the need for time series data mining benchmarks: A survey and empirical demonstration. In: Proceedings of 8th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 102–111
10. Keogh E, Ratanamahatana CA (2005) Exact indexing of dynamic time warping. Knowl Inf Syst 7(3):358–386
11. Keogh EJ, Chakrabarti K, Pazzani MJ, Mehrotra S (2001) Dimensionality reduction for fast similarity search in large time series databases. Knowl Inf Syst 3(3):263–286
12. Keogh EJ, Chu S, Hart D, Pazzani M (2004) Segmenting time series: a survey and novel approach. In: Last M, Kandel A, Bunke H (eds) Data mining in time series databases, series in machine perception and artificial intelligence, vol 57. World Scientific Publishing Company, Singapore, pp 1–22 chap 1
13. Keogh EJ, Lin J, Fu AW (2005) HOT SAX: efficiently finding the most unusual time series subsequence. In: Proceedings of 5th IEEE international conference on data mining, pp 226–233
14. Keogh EJ, Lin J, hee Lee S, Herle HV (2007) Finding the most unusual time series subsequence: algorithms and applications. Knowl Inf Syst 11(1):1–27
15. Khanh NDK, Anh DT (2012) Time series discord discovery using WAT algorithm and iSAX representation. In: Proceedings of 3rd symposium on information and communication technology. ACM, pp 207–213
16. Lin J, Keogh E, Lonardi S, Chiu B (2003) A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of 8th ACM SIGMOD workshop on research issues in data mining and knowledge discovery, pp 2–11
17. Lin J, Vlachos M, Keogh E, Gunopulos D (2004) Iterative incremental clustering of time series. In: Proceedings of international conference on extending database technology. Springer, New York, pp 106–122
18. Lin J, Vlachos M, Keogh E, Gunopulos D, Liu J, Yu S, Le J (2005) A MPAA-based iterative clustering algorithm augmented by nearest neighbors search for time-series data streams. In: Ho T, Cheung D, Liu H (eds) Advances in knowledge discovery and data mining, LNCS 3518. Springer, New York, pp 333–342

19. Lin J, Keogh EJ, Wei L, Lonardi S (2007) Experiencing SAX: a novel symbolic representation of time series. Data Min Knowl Discov 15(2):107–144
20. Malinowski S, Guyet T, Quiniou R, Tavenard R (2013) 1d-SAX: a novel symbolic representation for time series. In: Advances in intelligent data analysis XII, LNCS 8207. Springer, New York, pp 273–284
21. Ordonez P, Armstrong T, Oates T, Fackler J (2011) Classification of patients using novel multivariate time series representations of physiological data. In: Proceedings of 10th international conference on machine learning and applications and workshops, vol 2, pp 172–179
22. Sanchez H, Bustos B (2017) Multi-resolution time series discord discovery. In: Advances in computational intelligence: 14th international work-conference on artificial neural networks, proceedings, Part II. Springer, New York, pp 116–128
23. Shieh J, Keogh E (2008) iSAX: indexing and mining terabyte sized time series. In: Proceedings of 14th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 623–631
24. Wang X, Mueen A, Ding H, Trajcevski G, Scheuermann P, Keogh E (2013) Experimental comparison of representation methods and distance measures for time series data. Data Min Knowl Discov 26(2):275–309
25. Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 1(6):80–83
26. Wu YL, Agrawal D, El Abbadi A (2000) A comparison of DFT and DWT based similarity search in time-series databases. In: Proceedings of 9th international conference on information and knowledge management. ACM, pp 488–495
27. Yi BK, Faloutsos C (2000) Fast time sequence indexing for arbitrary Lp norms. In: Proceedings of 26th international conference on very large data bases. Morgan Kaufmann Publishers Inc., pp 385–394