Multi-resolution Time Series Discord Discovery

Heider Sanchez and Benjamin Bustos(✉)

Department of Computer Science, University of Chile, Santiago, Chile
{hesanche,bebustos}@dcc.uchile.cl

Abstract. Discord Discovery is a recent approach for anomaly detection in time series that has attracted much research because of the wide variety of real-world applications in monitoring systems. However, finding anomalies by different levels of resolution has received little attention in this research line. In this paper, we introduce a multi-resolution representation based on local trends and mean values of the time series. We require the level of resolution as parameter, but it can be automatically computed if we consider the maximum resolution of the time series. In order to provide a useful representation for discord discovery, we propose dissimilarity measures for achieving high effective results, and a symbolic representation based on SAX technique for efficient searches using a multi-resolution indexing scheme. We evaluate our method over a diversity of data domains achieving a better performance compared with some of the best-known classic techniques.

Keywords: Time series · Anomaly detection · Discord discovery · Indexing

1 Introduction

In light of recent advancements in streaming technologies, the anomaly detection in time series has become an important task in different applications of monitoring systems, such as: analysis of video surveillance, multiple sensors in car and aircraft crashes, tracking of objects in riot detection, traffic alert on roads, seismic signals, electrocardiograms (ECG), etc. To address the anomaly detection in time series, we first need to define the type of anomaly that fits the application purpose and the data domain. It is a complex issue, because the anomaly can be associated with outlier points (irregularities, change point), outlier subsequences (unusual patterns, novelty), or anomalous relation between variables. The most common solutions for anomaly detection are built by machine learning methods [2]. Usually, they are supervised learning techniques and need a time series sample of "normal" behavior as training model, and in other cases a set of unusual patterns is required. These sets are provided by a domain expert.

H. Sanchez—Work supported by a research grant from CONICYT-Chile.
B. Bustos—Supported by the Millennium Nucleus Center for Semantic Web Research under Grant NC120004.

I. Rojas et al. (Eds.): IWANN 2017, Part II, LNCS 10306, pp. 116–128, 2017.
DOI: 10.1007/978-3-319-59147-6_11

However, in many real contexts, obtaining this prior knowledge is a difficult task. This is where Unsupervised Learning Techniques are used to try to identify significant patterns, which adjust the knowledge model directly from the input stream.

We focus this research on outlier subsequences, taking as baseline the Discord Discovery Process proposed by Keogh et al. [6]. They previously conducted an important work to identify the most unusual subsequence of long time series designing an unsupervised window-based method. The main characteristic of the subsequences is their level of overlapping; consecutive subsequences are more similar to each other. Accordingly, there is a necessity for optimal structures that consider this property for efficient searching. A series of related works about discord discovery were proposed in the past decade [1,7,8,12]. The main objective of these related works was the detection efficiency, because the brute force algorithm has a quadratic order regarding the total number of subsequences. However, multi-resolution discord discovery has received little attention in this research line.

In this paper, we introduce (1) 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. We also propose (2) a symbolic representation derived from the numeric representation by applying SAX quantization [9] on the trend and value components. It also provides us a lower bounding function for indexing time series collection. The main contribution is (3) a multi-resolution discord discovery technique based on this time series representation. The efficacy and efficiency of our approach is experimentally evaluated over a diversity of data domains [5]. We empirically demonstrate that our method outperforms conventional methods.

2 Background and Related Work

We associate an anomaly into a time series as a subsequence (unusual pattern) which produces a qualitatively significant change in the behavior of data. Unusual patterns are outstanding subsequences that arbitrarily occur and are associated with residual variation of the time series. This is contrary to the frequent patterns that regularly occur and are associated with cyclical or periodic variation of time series, moreover, they are located into the normal observation of the time series. Keogh et al. [6] introduced this new approach to avoid creating a workable definition for "the most unusual subsequence", and furthermore it is an unsupervised method that does not require training data.

Definition 1 (Non-self match). *Given a time series P, containing a subsequence C_i of length w and a matching subsequence C_j of the same length, we say that C_j is a non-self match to C_i if $|i - j| \geq w$, where p and q are their respective starting positions in P.*

Definition 2 (Time Series Discord). *Given a time series P, the subsequence C_j of length w is said to be the discord of P if C_j has the largest distance to its nearest non-self match.*

This problem can be solved by a brute force search using a nested loop. The outer loop takes each subsequence as a possible candidate, and the inner loop is used to search the candidate's nearest non-self match. The candidate that has the greatest such value is the discord. The computational complexity is $O(N^2)$, where N is the number of subsequences. To improve this complexity, Keogh et al. [6] proposed a generic algorithm for efficient detection. This algorithm requires two heuristics that generate two ordered lists of subsequences; one for the outer loop and the other one for the inner loop. The heuristic *Outer* is useful for quickly finding the best candidate, and the heuristic *Inner* is useful for quickly finding the best nearest non-self match. We break out of the inner loop if the distance is less than the best-so-far discord distance. Two main related methods for discord discovery are HOT SAX [7] and HOT iSAX [1], which are based on SAX representation [9]. SAX splits the time series into segments and builds a new symbolic time series quantizing the mean values of each segment. Both techniques build efficient structures to find the time series discord using the discord discovery heuristics.

3 Multi-resolution Trend-Value Approximation

3.1 Why a Trend-Based Representation?

Esmael et al. claim 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" [4]. In this way, several piecewise approximations based on trend and value features have been recently proposed [3,4,11]. We focus on the technique 1d-SAX proposed by Malinowski et al. [11]. This is a compact binary representation to improve the retrieval performance using the same amount of information that SAX. Here, SAX is extended by adding new symbols that represent the slope of each segment. The algorithm uses linear regression to compute the slope. 1d-SAX works with alphabets of different sizes:

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

where $\hat{v}_i$ is the average value symbol from an alphabet of size α^v, and $\hat{s}_i$ is the slope symbol from an alphabet of size α^s.

We also compare both methods, SAX and 1d-SAX, using an agglomerative hierarchical clustering to group five time series in three different classes (Fig. 1). The time series is split into four segments. SAX only takes the mean value while that 1d-SAX also considers the slope obtaining a better match between time series 2 and 3 which belong to same class.

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 SAX and 1d-SAX produces a horizontal segmentation, we propose a hierarchical segmentation induced by the resolution level. This segmentation

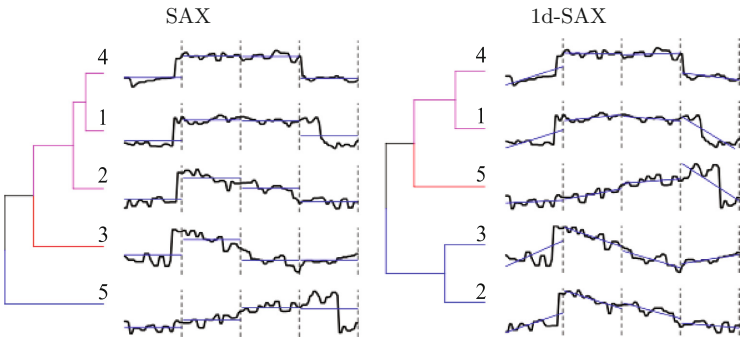

Fig. 1. A comparison of the ability of two time series representations to cluster five members of the CBF dataset using the Euclidean distance.

provides greater advantages in design and optimization that will be discussed. Our time series representation is called Multi-resolution Trend-Value Approximation (MTVA). The basic idea is to generate trend-value pairs on each level of resolution, and then compute the similarity between two MTVA representations using a distance measure. In addition, we design a symbolic representation to build a multi-resolution indexing structure for discord discovery.

3.2 Bottom-Up Construction Algorithm

Given the times series $P = \{p_1, \cdots, p_n\}$ 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 $Y = \{y_1, \cdots, y_w\}$, be a segment of P in the time segment $X = \{x_1, \cdots, x_w\}$, 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,
3. For the next resolution levels $M = 2^{\{L-2, L-3, \cdots, 0\}}$, 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 is the mean value and the average time associated to a segment in the upper level (see Fig. 2).

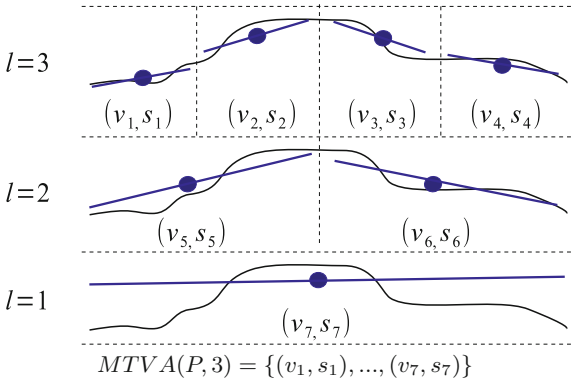

Fig. 2. Construction of the Multi-resolution trend-value approximation.

4. The result is an array of all the trend-value pairs:

$$MTVA(P, L) = \{(v_1, s_1), ..., (v_m, s_m)\}.$$

Figure 2 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 $m \leq n/2$. On the other hand, m can be defined in terms of the level of resolution $m = 2^L - 1$. Then solving both equations, we obtain that the maximum level of resolution for P is $L_{max} = \lfloor \log_2(n/2) \rfloor + 1$.

3.3 MTVA Distance

We first need a cost function to measure the distance between trend-value pairs. Given two pairs p_i and q_j, we define the cost function as follows:

$$cost(p_i, q_j) = |v_i^p - v_j^q|^2 + |s_i^p - s_j^q|^2,$$

where both value-domain and slope-domain should have similar range 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). We then propose a multi-resolution distance $MDist$ to measure the dissimilarity between two MTVA representations executing the cost function on all levels of resolution:

$$MDist(P, Q) = \sum_{l=1}^{L} \sum_{i=2^{(l-1)}}^{2^l - 1} cost(p_i, q_i).$$

The computational time of executing $MDist$ is the sum of the time in each level of resolution:

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

If we compute the distance in the worst case where L is exactly $\log_2(n/2) + 1$, the computational time is of order $O(n)$, where n is the length of the original time series. Therefore, $MDist$ in the worst case is theoretically as fast as the classic distances that work over the raw representation.

3.4 Symbolic Representation

Discretization techniques are used to 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 3. *"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)" [9]. For example, if* $\alpha = 4$ *then the breakpoints are* $\{\beta_1 = -0.67, \beta_2 = 0, \beta_3 = +0.67\}$.

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 separately both values using breakpoints that produce equal-size areas under the Gaussian curve $N(\mu, \sigma^2)$ (similar to 1d-SAX). Gaussian discretization is feasible for normalized time series, since statistically the mean value and the slope have a Gaussian distribution [10,11]. 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 (Fig. 3), 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 similar range. If the time series is normalized in $N(0,1)$, then the temporal component X must fit in this interval size. In this work, we normalize the length of each segment $X = [1, w] \rightarrow X = [-1, 1]$. In this manner, the variance σ_L^2 is defined in terms of the level of resolution independently of the size of the segment.

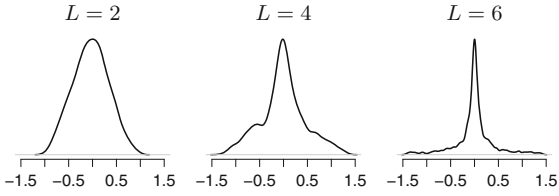

Fig. 3. Density of the slope varying the level of resolution in ECG time series.

Alphabet Size. The alphabet size is delimited by the number of breakpoints (Definition 3) and strongly influences over 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. For conformity, we use binary symbols where α is power of two [13]. Thus for example, to compress the numeric MTVA up the level 3 using $\alpha_v = 4$ and $\alpha_s = 4$, we need $(2+2)*(2^3-1)$ bits, it is less than 4 bytes by time series. This symbolic representation will serve us to different applications like indexing and anomaly detection.

3.5 Indexing

To efficiently manage MTVA time series datasets, we use the symbolic representation to build a hash-based index, where each bucket $\hat{P}$ envelops a set of similar MTVA time series. To filter out distances in the similarity search, we design a lower bounding function called MINDIST to measure the distance between the query object Q and a bucket $\hat{P}$, so that it is less than the distance between Q and any object $P \in \hat{P}$. Before defining MINDIST, we first need to define the lower bounding function of the trend-value cost, which is denoted as follows:

$$LB\_cost(\hat{p}_i, q_i) = (\Delta v)^2 + (\Delta s)^2 \leq cost(p_i, q_i), \text{where}$$

$$\Delta v = \begin{cases} |v_i^q - \beta_{Ui}| & v_i^q > \beta_{Ui} \\ |\beta_{Li} - v_i^q| & v_i^q < \beta_{Li} \\ 0 \text{ else,} & \end{cases} : \beta_{Li} \leq v_i^p < \beta_{Ui},$$

$$\Delta s = \begin{cases} |s_i^q - \beta_{Ui}| & s_i^q > \beta_{Ui} \\ |\beta_{Li} - s_i^q| & s_i^q < \beta_{Li} \\ 0 \text{ else,} & \end{cases} : \beta_{Li} \leq s_i^p < \beta_{Ui}.$$

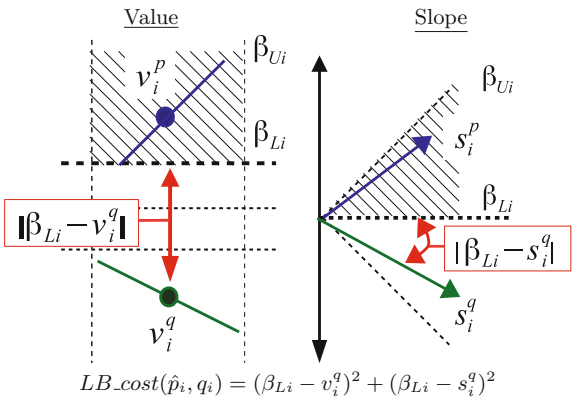

Fig. 4. Lower bounding trend-value cost. The blue line represents a trend-value pair stored in our database and the green line is the query (Color figure online).

The symbol $\hat{p}_i$ is derived from a trend-value pair p_i that is located between two breakpoints $\beta_{Ui} < p_i \leq \beta_{Li}$, independently for each pair value (Fig. 4). MINDIST is then calculated by following equation:

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

where l is the current resolution level and α is the accumulated distance of the previous levels.

4 Multi-resolution Discord Discovery

The main challenge of the discord discovery approach is to face its quadratic complexity. In this sense, our MTVA representation together with the discord discovery heuristics can be used to solve the anomaly detection in time series. We propose a multi-resolution method called HOT MTVA, which increases the level resolution of the index when a hash-bucket is overflowed (Fig. 5). This indexing structure allows the perfect fit to our MTVA representation. Moreover, it is flexible to control the level of resolution of the detected anomaly.

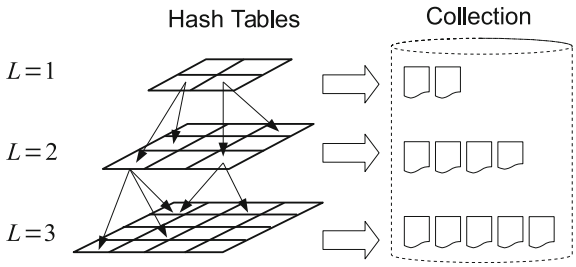

Fig. 5. Multi-resolution index model for the MTVA representation.

4.1 Building Algorithm

Given the times series P of length n, we use an overlapping sliding window of size $w \ll n$ for extracting all the possible subsequences $C_i,\ i \in \{1, \ldots, (n - w + 1)\}$ from P. The insertion procedure of a MTVA subsequence C_i into the indexing structure $\mathbb{R}$ is described as follows. Unlike HOT iSAX, we apply hierarchical quantization to access to 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, so we create new nodes with the next resolution level of the current node. We use a size threshold th_{max} to control the maximum number of objects in a terminal node (the so-called bucket). As we can see, the indexing level has dynamic behavior, its incremental value depends of the size of dataset and the maximum level of resolution (L_{max}).

4.2 Discord Discovery Heuristics

The discordant subsequence is found applying the optimal discord discovery procedure [7] 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. Afterwards, we visit the rest of buckets in random order. This heuristic ensures that the subsequences that are most isolated, in 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_j. We first visit all subsequences contained

Algorithm 1. NNM-Search for the Multi-resolution MTVA Index

Require: (Index $\mathbb{R}$, Query C_j, 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: $d = MINDIST(child\_node.str, C_j, node.level, min\_d)$
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** $C_i \in objects$ **do**
21: **if** $|i - j| \geq w$ **then** ▷ non-self match?
22: $d = MDist(C_i, C_j)$ ▷ Multi-resolution Distance
23: **if** $d < best\_dist$ **then**
24: $best\_dist = d$
25: $best\_post = j$
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** $(best\_dist, best\_pos)$

in the bucket from which C_j is retrieved. Afterwards, we apply the nearest non-self match search algorithm (NNM-Search, Algorithm 1) to visit the rest of the buckets. This heuristic allows to first visit all the subsequences most similar to C_j increasing the probability of early termination of the loop.

The NNM-Search algorithm performs a hierarchical search across the internal nodes using a stack to maintain the nodes ordered in terms of MINDIST (lines 10 – 17). MINDIST measures the minimum distance between the query and the current node. The algorithm also applies two breaking statements to break the inner loop as early as possible: one is associated to MINDIST (line 6) and the other one is associated to the best-so-far discord distance (line 27).

5 Experimental Results

In this section, we evaluate the performance of our approach to address the anomaly detection problem. Effectiveness will be evaluated over a set of 20 real cases of anomalous time series (with different sliding windows) collected by Keogh et al. [5,6]. Efficiency will be evaluated using the same set of long time series described by Sanchez et al. [12].

5.1 Effectiveness of Our MTVA Representation

We first evaluate the accuracy of our trend-value numeric representation over all anomaly cases. The classic techniques use the Euclidean Distance as measure distance over the raw representation of the normalized subsequence. The normalization process is used to homogenize all subsequences applying implicitly two transformations: translation and scaling. This has a problem, though: the presence of local noise is amplified by the scaling transformation. This issue is solved by using a context-dependent parameter $\varepsilon > 0$ for smoothing noisy subsequences [9,12]. An important feature of our MTVA representation is that the slope of the noisy segments trends to zero and thereby the unusualness of noisy subsequences is reduced. Table 1 shows the results obtained by both techniques for six values of ε. We evaluate the MTVA using three different resolution levels. We note that the MTVA representation achieves a higher percentage of true detections when increasing the resolution level to $L = 4$, this is the common maximum level for all the sliding windows. In this way, we assert that our method is more robust to local noise than the classic ED. Furthermore, we can improve this percentage up to 100% of true detections finding the best value for ε in each of the time series. Additionally, we highlight the flexibility of MTVA for dynamically working in different levels of resolution at runtime.

5.2 Efficiency of Our Multi-resolution Index

We accelerate the search using our multi-resolution method HOT MTVA and compare it with two main state-of-the-art techniques: HOT SAX and HOT iSAX (Sect. 2). We set the same quantitative information for each technique: $\alpha_v = 4$

Table 1. Percentage of true detections using our MTVA representation.

ε	ED	MTVA Distance		
		$L = 2$	$L = 3$	$L = 4$
0.025	60%	60%	67%	70%
0.050	77%	70%	83%	87%
0.075	73%	70%	80%	83%
0.100	80%	73%	83%	87%
0.125	83%	77%	83%	87%
0.150	77%	73%	77%	80%
best	100%	–	100%	100%

and $\alpha_s = 4$ for the MTVA index, and $\alpha_v = 8$ for the SAX-based indexes. We set empirically the maximum number of elements in a bucket as $th_{max} = 50$ and the maximum resolution level is restricted to $L = 4$. Additionally, we add a search optimization strategy to the HOT SAX algorithm and call it HOT SAX*, which consists in applying a MINDIST function in the bucket before of visiting their associated subsequences [9]. Figure 6 shows the efficiency of the algorithms in terms of the number of computed distances. We observe that HOT MTVA is much more efficient than the other techniques in terms of computed distances. This efficiency advantage is due to multi-resolution properties of our method, which allows to the outer loop heuristic to find quickly the potential candidates through the resolution levels.

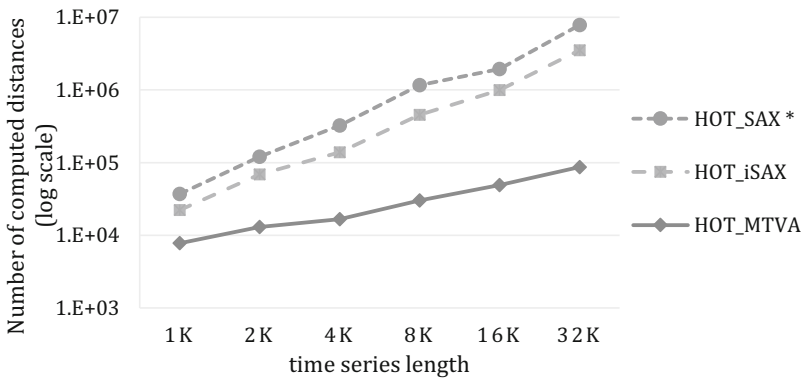

Fig. 6. Efficiency of our multi-resolution method in anomaly detection.

6 Conclusions and Future Work

We proposed a multi-resolution time series representation (MTVA) which is composed of trend-value pairs obtained by applying linear regression linear in each resolution segment. We also provided a distance measure and its lower bounding function to perform efficient searches. We demonstrated the utility of our MTVA representation in Anomaly Detection, where we have highlighted the slope feature for mitigating the false unusualness of noisy subsequences. Furthermore, the efficiency of our multi-resolution discord discovery algorithm outperformed the best existing methods in terms of computed distances. One additional advantage of the MTVA representation is that the level of resolution was more intuitive and easier to fine-tune than the number of segments in piecewise approximations. Nevertheless, one disadvantage of the trend-value approximation is that it requires 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. Our approach may be used as baseline for finding anomalies in different levels of granularity. We plan to focus our method on online anomaly detection for massive streaming data.

References

1. Buu, H.T.Q., Anh, D.T.: Time series discord discovery based on iSAX symbolic representation. In: Proceedings of Third International Conference on Knowledge and Systems Engineering (KSE), pp. 11–18 (2011)
2. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. **41**, 1–58 (2009)
3. Dan, J., Shi, W., Dong, F., Hirota, K.: Piecewise trend approximation: a ratio-based time series representation. Abstr. Appl. Anal. **2013**(4) (2013)
4. Esmael, B., Arnaout, A., Fruhwirth, R.K., Thonhauser, G.: Multivariate time series classification by combining trend-based and value-based approximations. In: Murgante, B., Gervasi, O., Misra, S., Nedjah, N., Rocha, A.M.A.C., Taniar, D., Apduhan, B.O. (eds.) ICCSA 2012. LNCS, vol. 7336, pp. 392–403. Springer, Heidelberg (2012). doi:10.1007/978-3-642-31128-4_29
5. Keogh, E., Lin, J., Fu, A.: Univariate Time series discords datasets (2005). http://www.cs.ucr.edu/~eamonn/discords/
6. Keogh, E.J., Lin, J., Fu, A.W.: HOT SAX: efficiently finding the most unusual time series subsequence. In: Fifth IEEE International Conference on Data Mining, pp. 226–233, November 2005
7. Keogh, E.J., Lin, J., Lee, S.H., Herle, H.V.: Finding the most unusual time series subsequence: algorithms and applications. Knowl. Inf. Syst. **11**, 1–27 (2007)
8. Kha, N.H., Anh, D.T.: From cluster-based outlier detection to time series discord discovery. In: Li, X.-L., Cao, T., Lim, E.-P., Zhou, Z.-H., Ho, T.-B., Cheung, D., Motoda, H. (eds.) PAKDD 2015. LNCS, vol. 9441, pp. 16–28. Springer, Cham (2015). doi:10.1007/978-3-319-25660-3_2
9. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: 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 (2003)

10. Lin, J., Keogh, E.J., Wei, L., Lonardi, S.: Experiencing SAX: a novel symbolic representation of time series. Data Min. Knowl. Discov. **15**, 107–144 (2007)
11. Malinowski, S., Guyet, T., Quiniou, R., Tavenard, R.: 1d-SAX: a novel symbolic representation for time series. In: Tucker, A., Höppner, F., Siebes, A., Swift, S. (eds.) IDA 2013. LNCS, vol. 8207, pp. 273–284. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41398-8_24
12. Sanchez, H., Bustos, B.: Anomaly detection in streaming time series based on bounding boxes. In: Traina, A.J.M., Traina, C., Cordeiro, R.L.F. (eds.) SISAP 2014. LNCS, vol. 8821, pp. 201–213. Springer, Cham (2014). doi:10.1007/978-3-319-11988-5_19
13. Shieh, J., Keogh, E.: iSAX: indexing and mining terabyte sized time series. In: Proceedings of 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 623–631. ACM (2008)