Variable size segmentation for efficient representation and querying of non-uniform time series datasets

Lamia Djebour, Reza Akbarinia, Florent Masseglia

Variable Size Segmentation for Efficient Representation and Querying of Non-Uniform Time Series Datasets

Lamia Djebour
INRIA & LIRMM, Univ Montpellier
France
lamia.djebour@inria.fr

Reza Akbarinia
INRIA & LIRMM, Univ Montpellier
France
reza.akbarinia@inria.fr

Florent Masseglia
INRIA & LIRMM, Univ Montpellier
France
florent.masseglia@inria.fr

ABSTRACT

Existing approaches for time series similarity computing are the core of many data analytics tasks. Given the considered data volumes, or simply the need for fast response times, they often rely on shorter representations, usually with information loss. This incurs approximate comparisons where precision is a major issue. We present and experimentally evaluate ASAX, a new approach for segmenting time series before their transformation into symbolic representations. ASAX reduces significantly the information loss incurred by possible splittings at different steps of the representation calculation, particularly for datasets with unbalanced (non-uniform) distributions. We provide theoretical guarantees on the lower bound of similarity measures, and our experiments illustrate that our method outperforms the state of the art, with significant gain in precision for datasets with unbalanced distributions.

CCS CONCEPTS

• **Information systems** → **Data mining**; **Spatial-temporal systems**; **Nearest-neighbor search**.

KEYWORDS

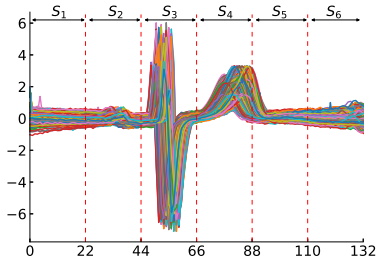
Time Series, Representations, Information Retrieval

ACM Reference Format:
Lamia Djebour, Reza Akbarinia, and Florent Masseglia. 2022. Variable Size Segmentation for Efficient Representation and Querying of Non-Uniform Time Series Datasets. In *Proceedings of ACM SAC Conference (SAC'22).* ACM, New York, NY, USA, Article 4, 8 pages.

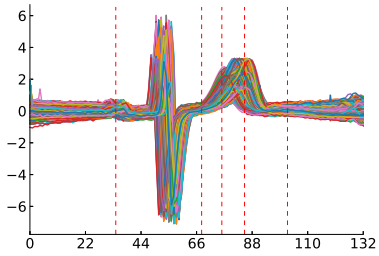
1 INTRODUCTION

Many applications in different domains generate time series data at an increasing rate. That continuous flow of emitted data may concern personal activities (*e.g.*, through smart-meters or smart-plugs for electricity or water consumption) or professional activities (*e.g.*, for monitoring heart activity or through the sensors installed on plants by farmers). This results in the production of large and complex data, usually in the form of time series [8, 9, 13, 15–19, 21, 22] that challenge knowledge discovery. Data mining techniques on such massive sets of time series have drawn a lot of interest since their application may lead to improvements in a large number of these activities, relying on fast and accurate similarity search in time series for performing tasks like , e.g., Classification, Clustering, and Motifs Discovery [14, 17, 23].

Because of the considered data volumes in such applications, these tasks can be slow on raw data. This is why approximation over time series is often regarded as a means to allow fast computation of similarity search. SAX [11] is one of the most popular representations of time series, allowing dimensionality reduction on the classic data mining tasks. SAX is efficient because it constructs symbolic representations by splitting the time domain into segments of equal size. This approximation model is effective for time series having a uniform and balanced distribution over the time domain. However, we observe that, in the case of time series having high variation over some time intervals, this "one size fits all" division into segments of fixed length is not advantageous.



(a) SAX segmentation on *D*, with 6 segments



(b) ASAX segmentation on *D*, with 6 segments

Figure 1: SAX segmentation Vs. ASAX segmentation

To illustrate the impact of a fixed length division of the series into segments, let us consider Figure 1. It shows a set D of time series, taken from *ECGFiveDays* dataset of UCR Archive [5], where the time series length is 132. In this dataset, the distribution of values over time domain is not uniform. We can notice that there is almost no variation from time point 1 to 44 and from 95 to 132. On the other hand, the remaining part, from time point 44 to 95, shows an important variation in the data values. Figure 1a shows the SAX division on D, with a fixed-size segmentation on the time series. In this example the segment size is 22, leading to 6 segments in

SAC'22, April 25 –April 29, 2022, Brno, Czech Republic
.

total. If we take any time series X from D and we convert it into its SAX representation, the first two segments are always represented by the same symbol, all the values of these two segments being close to each other. Actually, there is no need to consider these two distinct segments. And the same applies to the last two segments. Meanwhile, for segments 3 and 4, all the values of each segment are represented by a single symbol while the data values present great variations, causing a significant loss of information on these segments.

As one can observe, it is not necessary to split the parts that are constant or where the variation is low since they don't carry any relevant information and would therefore better form a single segment. It is more efficient to divide into several small segments the parts where variation is important in order to preserve potentially relevant information as shown in Figure 1b. The splitting of Figure 1b is the actual splitting obtained by our approach with a segment budget limited to 6. The first two and last two segments better correspond to the information carried by the series. It would be rather counter-intuitive to merge segments 1 and 2, while it is the opposite for Figure 1a. The time intervals where data values show important differences are split like, *e.g.*, between times point 66 and 88. By proposing such a customized splitting, we aim at improving the performance of information retrieval algorithms that will rely on our data representation.

In order to improve the quality of similarity search, and to achieve adaptive splitting as illustrated above, we propose a new approximation method for time series that considers the time series shape and does the splitting by means of segments of variable size on the time domain. By measuring the entropy of symbolic representations, our algorithm chooses between different possible splittings at each step of the representation computation. This approach allows reducing information loss, and thus increasing the accuracy of time series representations leading to better precision during retrieval phases, particularly from non-uniform datasets. In this paper, we make the following contributions:

- We propose a new representation technique, called ASAX (Adaptive SAX), that allows obtaining a variable-size segmentation of time series with better precision in retrieval tasks thanks to its lower information loss. Our representation is based on entropy measurement for detecting what time intervals should be split.
- We propose a lower bounding method that allows approximating the distance between the original time series based on their representations in ASAX.
- We implemented our approach and conducted empirical experiments using more than 10 real world datasets. The results suggest that ASAX can obtain significant performance gains in terms of precision for similarity search compared to SAX. They illustrate that the more the data distribution in the time domain is unbalanced, the greater is the precision gain of ASAX. For example, for the *EGCFiveDays* dataset that has a non-uniform distribution in the time domain, the precision of ASAX is 82% compared to 55% for SAX.

The rest of the paper is organized as follows, we review the related works in Section 2. In Section 3, we describe the details of ASAX representation. In Section 4, we present the experimental evaluation of our approach. We discuss the related work in Section 5, and finally conclude in Section 6.

2 PROBLEM DEFINITION AND BACKGROUND

In this section, we first present the background about SAX representation, and then define the problem we address.

Table 1: Some frequently used symbols

D	Time series database
X, Y , Q	Time series
$n = \|X\|$	The length of time series X
l	The segment size
w	The number of PAA segments
a	The cardinality (the alphabet size)
$\hat{X}$	The SAX representation of time series X
k	the k nearest neighbors

A time series X is a sequence of values $X = \{x_1, ..., x_n\}$. We assume that every time series has a value at every timestamp $t = 1, 2, ..., n$. The length of X is denoted by $|X|$.

SAX allows a time series T of length n to be reduced to a string of arbitrary length w. Table 1 lists the notations used in this paper.

2.1 SAX Representation

Given two time series $X = \{x_1, ..., x_n\}$ and $Y = \{y_1, ..., y_n\}$, the Euclidean distance between X and Y is defined as [6]: $ED(X, Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$

The Euclidean distance is one of the most straightforward similarity measurement methods used in time series analysis.

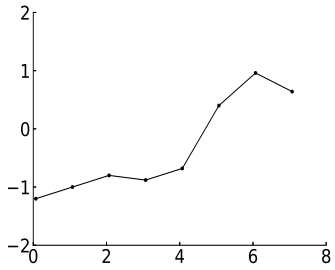
The SAX representation is based on the Piecewise Aggregate Approximation (PAA) representation [11] which allows for dimensionality reduction while providing the important lower bounding property as we will show later. The idea of PAA is to have a fixed segment size, and minimize dimensionality by using the mean values on each segment. Example 1 gives an illustration of PAA.

Example 1. *Figure 2b shows the PAA representation of X, the time series of Figure 2a. The representation is composed of $w = |X|/l$ values, where l is the segment size. With PAA, for each segment, the set of values is replaced with their mean. The length of the final representation w is the number of segments (and, usually, $w << |X|$).*

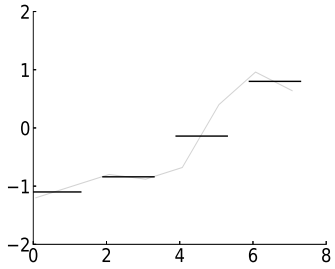
By transforming the original time series X and Y into PAA representations, $\overline{X} = \{\overline{x}_1, ..., \overline{x}_w\}$ and $\overline{Y} = \{\overline{y}_1, ..., \overline{y}_w\}$, the lower bounding approximation of the Euclidean distance for these two representations can be obtained by : $DR_f(\overline{X}, \overline{Y}) = \sqrt{\frac{n}{w}}\sqrt{\sum_{i=1}^{w}(\overline{x}_i - \overline{y}_i)^2}$

The SAX representation takes as input the reduced time series obtained using PAA. It discretizes this representation into a predefined set of symbols, with a given cardinality, where a symbol is a binary number. The size of the symbols set is called the cardinality. Example 2 gives an illustration of the SAX representation.

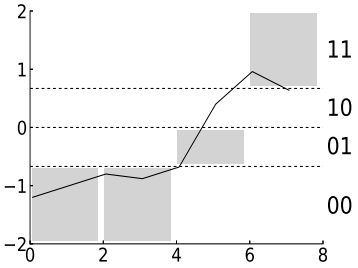
Example 2. *In Figure 2c, we have converted the time series X to SAX representation with 4 segments, and cardinality 4 using the PAA*



(a) A time series X of length 8



(b) A PAA representation of X, with 4 segments



(c) A SAX representation of X, with 4 segments and cardinality 4, $[00, 00, 01, 11]$.

Figure 2: A time series X is discretized by obtaining a PAA representation and then using predetermined break-points to map the PAA coefficients into SAX symbols. Here, the symbols are given in binary notation, where 00 is the first symbol, 01 is the second symbol, etc. The time series of Figure 2a in the representation of Figure 2c is [first, first, second, fourth] (which becomes $[00, 00, 01, 11]$ in binary).

representation shown in Figure 2b. In this example, we have 4 possible symbols: 11, 10, 01, 00. For each segment, SAX chooses the symbol that corresponds to the PAA value of the segment. Thus, the SAX representation of X is: [00, 00, 01, 11].

The lower bounding approximation of the Euclidean distance for SAX representation $\hat{X} = \{\hat{x}_1, ..., \hat{x}_w\}$ and $\hat{Y} = \{\hat{y}_1, ..., \hat{y}_w\}$ of two time series X and Y is defined as:
$MINDIST_f(\hat{X}, \hat{Y}) = \sqrt{\frac{n}{w}}\sqrt{\sum_{i=1}^{w}(dist(\hat{x}_i, \hat{y}_i))^2}$

where the function $dist(\hat{x}_i, \hat{y}_i)$ is the distance between two SAX symbols $\hat{x}_i$ and $\hat{x}_i$. The lower bounding condition is formulated as: $MINDIST_f(\hat{X}, \hat{Y}) \leq ED(X, Y)$

2.2 Similarity Queries

The problem of similarity queries is one of the main problems in time series analysis and mining. In information retrieval, finding the k nearest neighbors (k-NN) of a query is a fundamental problem. Let us define *exact* and *approximate* k nearest neighbors.

Definition 1. *(Exact k nearest neighbors) Given a query time series Q and a set of time series D, let $R = ExactkNN(Q, D)$ be the set of k nearest neighbors of Q from D. Let $ED(X, Y)$ be the Euclidean distance between two time series X and Y, then the set R is defined as follows:*
$(R \subseteq D) \wedge (|R| = k) \wedge (\forall a \in R, \forall b \in (D - R), ED(a, Q) \leq ED(b, Q))$

Definition 2. *(Approximate k nearest neighbors) Given a set of time series D, a query time series Q, and $\epsilon > 0$. We say that $R = AppkNN(Q, D)$ is the approximate k nearest neighbors of Q from D, if $ED(a, Q) \leq (1 + \epsilon)ED(b, Q)$, where a is the k^{th} nearest neighbor from R and b is the true k^{th} nearest neighbor.*

2.3 Time Series Approximation

The SAX representation proceeds to an approximation by minimizing the dimensionality: the original time series are divided into segments of equal size.

This representation does not depend on the time series values, but on their length. It allows SAX to perform the segmentation in $O(n)$ where n is the time series of length. However, for a given reduction in dimensionality, the modeling error may not be minimal since the model does not adapt to the information carried by the series. Our claim is that, by taking into account the information carried by time series for choosing the segments, we may obtain significant increase in the precision of kNN queries. This issue motivated us for proposing an adaptive representation aiming at minimizing the information loss.

2.4 Problem Statement

Our goal is to propose a variable-size segmentation of the time domain that minimizes the loss of information in the time series representation.

The problem we address is stated as follows. Given a database of time series D and a number w, divide the time domain into w segments of variable size such that the representation of the times series based on that segmentation lowers the error of kNN queries.

3 ADAPTIVE SAX (ASAX)

In this section, we propose ASAX, a variable-size segmentation technique for the time series representation. To create a segmentation with minimum information loss, ASAX divides the time domain based on the representation entropy.

In the rest of this section, we first describe the notion of entropy for the time series representation. Then, we describe our algorithm for creating the variable-size segments. Finally, we present our method for measuring the lower bound distance between time

series in the proposed representation. This lower bounding is useful for efficient evaluation of kNN queries.

3.1 Entropy

Entropy is a mathematical function which intuitively corresponds to the amount of information contained or delivered by a source of information. This source of information can be of various types. The more the source emits different information the higher is the entropy. If the source always sends the same information, the entropy is minimal. Formally, entropy is defined as follows.

Definition 3. *Given a set X of elements, and each element $x \in X$ having a probability P_x of occurrence, the entropy H of the set X is defined as:* $H(X) = -\sum_{x \in X} P_x \times \log P_x$

In our context, we calculate the entropy on a set containing the different symbolic representations obtained from the transformation of the original time series of a dataset according to a given segmentation. The entropy computed on this set allows to measure the quantity of information contained in the time series representations. Let us illustrate this using an example.

Example 3. *Consider the database D={x,y,z} in Figure 3 where x, y and z are time series with l=8. Let us create a representation having two segments (e.g., 0-4, and 4-8), and then compute the entropy of the representation of the set D. To generate the representation of the time series x, y and z, they are discretized by obtaining their PAA representation and then using predetermined break-points to map the PAA coefficients into the corresponding symbols like the SAX representation proceeds. We have converted the 3 time series into symbolic representations with size 2, and cardinality 4. Thus, the symbolic representations of x, y and z are* $\hat{x} = [00, 10]$, $\hat{y} = [00, 10]$ *and* $\hat{z} = [00, 10]$, *respectively. We notice that the 3 time series have the same symbolic representation, thus, the set X consists of only this unique symbolic representation with an occurrence equal to 3., i.e.,* $X = \{[00, 10]\}$. *The entropy H(X) of X is computed as follows:*

$H(X) = -(P(x = [00, 10]) \times \log_2 P(x = [00, 10]))$

where the probability for the word x is $P(x = [00, 10]) = \frac{3}{3} = 1$. *Therefore, we have* $H(X) = -(1 \log 1) = 0$ *meaning that in the representation X there is no information allowing to distinguish the three original time series from each other. This is explained by the fact that they have the same representation with a fixed-size segmentation.*

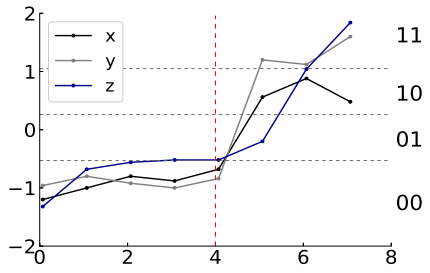


Figure 3: ASAX segmentation with 2 segments

In the next subsection, we describe our algorithm to create variable-size segments based on entropy.

3.2 Variable-Size Segmentation Based on Entropy Measurement

Given a database of time series D, and a number w, our goal is to find the k variable size segments that minimize the loss of information in time series representations.

Intuitively, our algorithm works as follows. First it splits the time domain into two segments of equal size. Then, it performs $w - 2$ iterations, and in each iteration it finds the segment s whose split makes the minimum loss in entropy, and it splits that segment. By doing this, in each iteration a new segment is added to the set of segments. This continues until having w segments.

Let us now describe ASAX in more details. The pseudo-code is shown in Algorithm 1. It first splits the time domain into two equal parts and creates two segments that are included to the set *segments* (Line 1). Then, it sets the current number of segments, denoted as k, to 2 (Line 2).

Afterwards, in a loop, until the number of segments is less than w the algorithm proceeds as follows. For each segment i (from 1 to k), i is divided into two equal parts, if its size is greater than *minSize*, which is the minimum possible size of a segment, and it's default value is 1. Then, a temporary set of segments *tempSegments* is created including the two new segments and all previously created segments except i (i.e., expect the one that has been divided). Then, for each time series ts in the database D, the algorithm generates the symbolic representation of ts (denoted as *word*) using the segments included in *tempSegments* with the given cardinality a (Line 12), and inserts it to a hash table (Line 13). Note that for all time series, ASAX uses the same cardinality to map the PAA coefficients into the corresponding symbols. After having inserted all the representations of the time series contained in D to the hash table, the entropy of the representations is calculated (Line 14). If the entropy is higher than the maximum entropy obtained until now, the algorithm sets i as the segment to be split, and keeps the entropy of the representation. This procedure continues by splitting one of the segments at each time, and computing the entropy. The algorithm selects the one whose entropy is the highest, and updates the set of the segments by removing the selected segment, and inserting its splits to the set *segments* (Lines 18-20). Then, the variable k, which shows the number of current segments, is incremented by one. The algorithm ends if the number of segments is equal to the required number, *i.e.*, w.

Example 4. *Let us consider the dataset D in Figure 3 which represents the initialization of the algorithm, i.e., the time domain is divided into two segments of the same size. The next step is to create the 3rd segment by splitting one of the two existing segments. Two different scenarios are possible.*

Scenario 1 : The first scenario is shown in Figure 4a where the left segment is divided into two equal parts. We generate the symbolic representation of the time series x, y, and z by using the 3 segments. Let's assume the cardinality is 4. Then, $\hat{x} = [00, 00, 10]$, $\hat{y} = [00, 00, 10]$ *and* $\hat{z} = [00, 00, 10]$ *are the symbolic representation of x, y and z, respectively. Thus, the set* X_1 *consists of only one representation [00,00,10] with an occurrence of 3, i.e.,* $X_1 = [00, 00, 10]$. *The entropy is then calculated as:* $H(X_1) = -(P(x = [00, 00, 10]) \log P(x = [00, 00, 10]))$ *where* $P(x = [00, 00, 10]) = \frac{3}{3} =$

Algorithm 1: ASAX variable-size segmentation

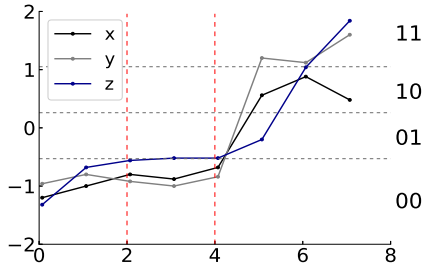
Input: D: time series database; n: the length of time series; $minSize$: the minimum possible size of a segment; a: cardinality of symbols; w: the required number of segments
Output: w variable-size segments

```
segments = {[0, n/2], [n/2, n]}; // split time domain into two equal size segments
k = 2
while k ≠ w do
    segmentToSplit = 1
    entropy = 0
    for i=1 to k do
        tempSegments = segments
        if length(tempSegments[i]) > minSize then
            split segment i into two equal parts, and replace the segment i by its corresponding parts in tempSegments
            hashtable = new HashTable
            foreach ts ∈ D do
                word = ASAX(ts, tempSegments, a)
                hashTable.put(word)
            e = entropy(hashTable)
            if e > entropy then
                segmentToSplit = i
                entropy = e
    split segmentToSplit into two equal size segments s1 and s2
    segments = segments - {segmentToSplit}
    segments = segments ⋃ {s1, s2}
    k = k+1
return segments
```

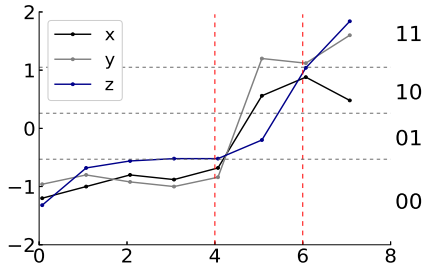
1 *and we have* $H(X_1) = -(1 \log 1) = 0$.

Scenario 2 : This scenario is shown in Figure 4b in which the right segment is split. As for Scenario 1 we generate the symbolic representation of time series x, y and z using the 3 segments, and cardinality of 4. $\hat{x} = [00, 01, 10]$, $\hat{y} = [00, 01, 11]$ *and* $\hat{z} = [00, 01, 11]$ *are the symbolic representation of x, y and z, respectively. In this scenario the representation set* X_2 *consists of [00,01,10] with an occurrence of 1 and [00,01,11] with an occurrence of 2, i.e.,* $X = [00, 01, 10], [00, 01, 10]$. *The entropy is calculated as:*
$H(X_2) = -(P(x = [00, 01, 10]) \log P(x = [00, 01, 10]) + P(x = [00, 01, 11]) \log P(x = [00, 01, 11]))$ *where* $P(x = [00, 01, 10]) = \frac{1}{3}$ *and* $P(x = [00, 01, 11]) = \frac{2}{3}$. *Then,* $H(X_2) = -(\frac{1}{3} \log \frac{1}{3} + \frac{2}{3} \log \frac{2}{3}) = 0.918$.

After having calculated the entropy for the two scenarios, we see that $H(X_1) < H(X_2)$. *We aim to maximize the entropy, therefore we choose the segmentation generated in Scenario 2 for this iteration of our algorithm. We continue the next iterations, until the number of segment reaches w.*



(a) Scenario 1 of ASAX segmentation with 3 segments



(b) Scenario 2 of ASAX segmentation with 3 segments

Figure 4: The two different scenarios of ASAX segmentation with 3 segments. Scenario 4b is the one chosen because it optimizes the entropy.

3.3 Lower Bounding of the Similarity Measure

SAX [12] defines a distance measure on the representation of time series as described in Section 2.1. Given the representation of two time series, the $MINDIST_f$ function allows obtaining a lower bounding approximation of the Euclidean distance between the original time series. By the following theorem, we propose a lower bounding approximation formula for the case of variable size segmentation in ASAX.

Theorem 1. *Let X and Y be two time series. Suppose that by using ASAX we create a variable size segmentation with w segments, such that the size of the* i^{th} *segment is* l_i.
Let $\overline{X}$ *and* $\overline{Y}$ *be the PAA representation of variable size of X and Y in ASAX,* $DR_v(\overline{X}, \overline{Y})$ *gives a lower bounding approximation of the Euclidean distance between X and Y:* $DR_v(\overline{X}, \overline{Y}) = \sqrt{\sum_{i=1}^{w} ((\overline{x}_i - \overline{y}_i)^2 \times l_i)}$
Let $\hat{X}$ *and* $\hat{Y}$ *be the representations of X and Y in ASAX obtained by converting* $\overline{X}$ *and* $\overline{Y}$ *into symbolic representation. Then,* $MINDIST_v(\hat{X}, \hat{Y})$ *gives a lower bounding approximation of the Euclidean distance between X and Y:* $MINDIST_v(\hat{X}, \hat{Y}) = \sqrt{\sum_{i=1}^{w} (dist(\hat{x}_i, \hat{y}_i)^2 \times l_i)}$

Proof. The proof has been removed due to lack of space.

3.4 Uniform Distribution of Symbols

SAX breakpoints divide the value domain into regions of different size where small regions are concentrated on the middle of the value domain and regions at extreme values are larger. This is illustrated

by Figure 5, with three time series from our motivating example in Figure 1. The breakpoints of SAX with 10 symbols are represented by horizontal lines, and, logically, they appear close to the center of the distribution. If we keep such distribution of symbols, then we would have two issues. First, the extreme values of the series like those above 2 or below -4 would be assigned the same symbol (their PAA value on the segment would fall in the same symbol). Second, the adaptive segmentation would consider that the slight variations around zero are more important than the ones at extreme values, ending in irrelevant splits that favor minor information gain. For this reason, we propose to calculate the breakpoints differently. In ASAX, the discretization is done based on breakpoints that produce uniform distributions of symbols. These breakpoints divide the value domain into regions of equal size. In the case of Figure 5 the 10 symbol regions will be evenly distributed in the range of data values.

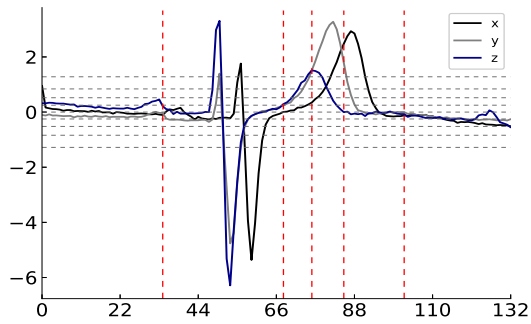


Figure 5: The Gaussian based distribution of symbols in SAX are not suitable for ASAX since they would favor minor information gain.

4 EXPERIMENTS

In this section, we report the results of experimental studies on the proposed ASAX segmentation approach that illustrate its performance in improving the accuracy of time series representations in order to get better precision during information search operations.

4.1 Datasets and Experimental Settings

We compared the ASAX representation with the existing SAX representation on datasets selected for their particular (lack of) uniformity. Notice that SAX and its extensions in the literature use a fixed-size segmentation of the time domain. But, ASAX proposes a variable-size segmentation based on information theory techniques.

The approaches are implemented in Python programming language and Numba JIT compiler is used to optimize machine code at runtime [1]. The experimental evaluation was conducted on a machine using Fedora 31 operating system with 16 Gigabytes of main memory, an Intel Core i7 1,90 GHz-4,80 GHz processor with 4 cores.

[1]Our code is available for download here: https://github.com/lamiad/ASAX

We carried out our experiments on several real world datasets from the UCR Time Series Classification Archive [5]. Table 2 gives basic information about the datasets: name, type, length of the time series (number of values). Notice that almost all selected datasets have non-uniform distributions over time domain (see Figure 6), else SyntheticControl that has a quasi uniform distribution.

For each approach, we set the default cardinality value to 32 and the length w of the approximate representations is reduced to 10% of the original time series length.

Table 2: Datasets basic information

Name	Type	time series Length
AllGestureWiimoteZ	Sensor	500
ECG200	ECG	90
ECG5000	ECG	140
ECGFiveDays	ECG	130
Fungi	HRM	200
GesturePebbleZ1	Sensor	450
MedicalImages	Image	90
SonyAIBORobotSurface1	Sensor	70
SyntheticControl	Simulated	60

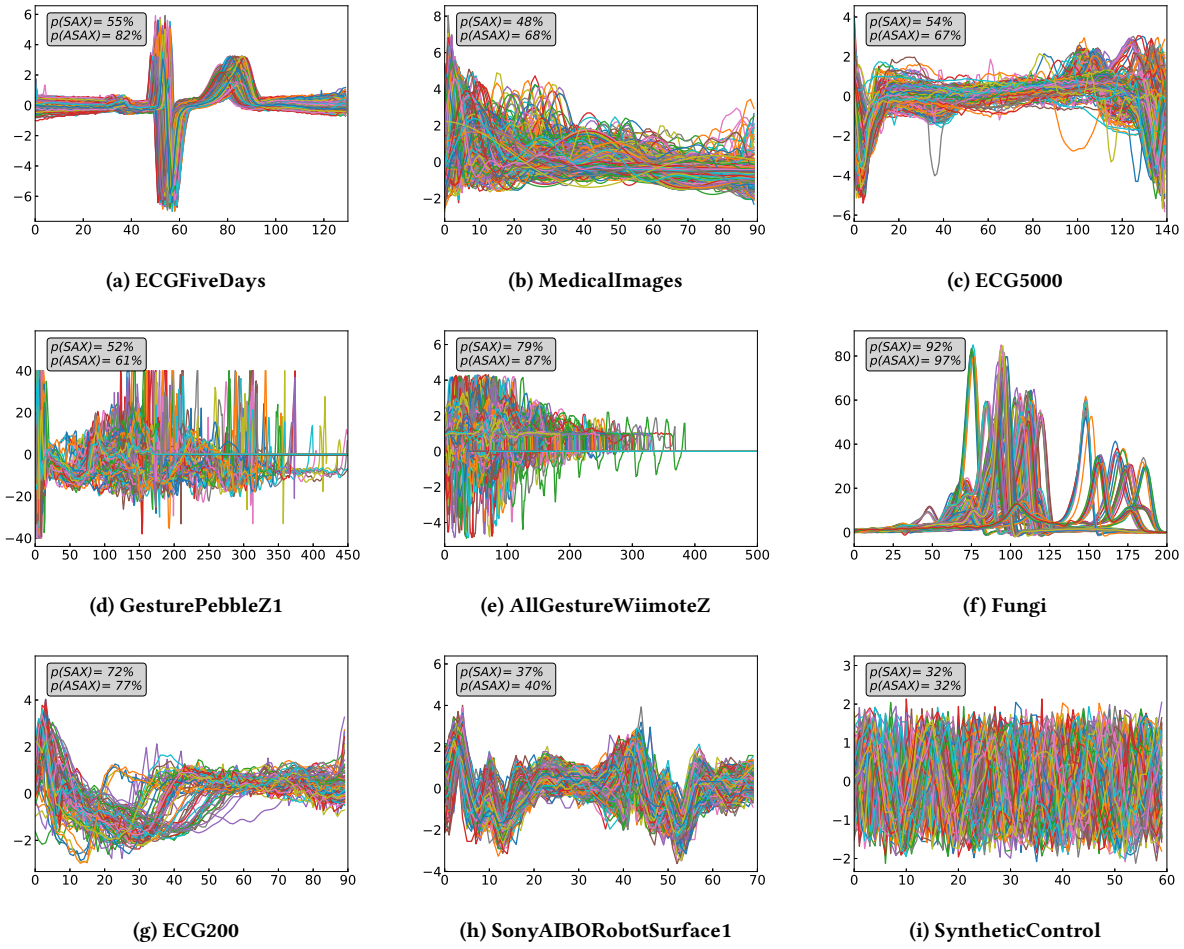
In the experiments, we measure the ASAX and SAX precision in similarity search by applying a k-Nearest Neighbor (k-NN) search, as detailed in Subsection 4.2. For ASAX, we measure the time cost of the variable-size segmentation in Subsection 4.3.

4.2 Precision of k-Nearest Neighbor Search

In this part of experiments, we compare the quality of ASAX and SAX representation on the different datasets described in Table 2 by measuring the precision of the approximate k-NN search for both of the two approaches. The precision reported for each dataset represents the average precision for a set of arbitrary random queries taken from this dataset. The search precision for each query Q from a dataset D is calculated as follows :

$$p = \frac{|AppkNN(Q,D) \cap ExactkNN(Q,D)|}{k}$$

where *AppkNN(Q,D)* and *ExactkNN(Q,D)* are the sets of approximate k nearest neighbors and exact k nearest neighbors of Q from D, respectively. *AppkNN(Q,D)* is obtained using DR_f distance measure for SAX and DR_v for the ASAX representation and the set *ExactkNN(Q,D)* contains the k-NN of Q using the euclidean distance *ED*. *AppkNN(Q,D)* and *ExactkNN(Q,D)* use a linear search that consists in computing the distance from the query point Q to every other point in D, keeping track of the "best so far" result.
The precision results are reported in Figure 6 where each dataset is plotted with the precision obtained (as percentage) for both approaches and the datasets are sorted in descending order of precision gain. The plots show the shape of the different time series of each dataset and we can notice that the distribution of time series over the time domain varies from one dataset to another. Let us take for example the *ECGFiveDays* dataset presented in Figure 6a and *SyntheticControl* shown in Figure 6i. On the first one, we were able to achieve a precision of 82% for ASAX while it is 55% for SAX, which is a significant gain in precision. This higher precision



(a) ECGFiveDays **(b) MedicalImages** **(c) ECG5000**

(d) GesturePebbleZ1 **(e) AllGestureWiimoteZ** **(f) Fungi**

(g) ECG200 **(h) SonyAIBORobotSurface1** **(i) SyntheticControl**

Figure 6: The data distribution of the tested datasets, and the precision results for each dataset. p(SAX) and p(ASAX) show the precision of SAX and ASAX respectively. The datasets are sorted in descending order of precision gain.

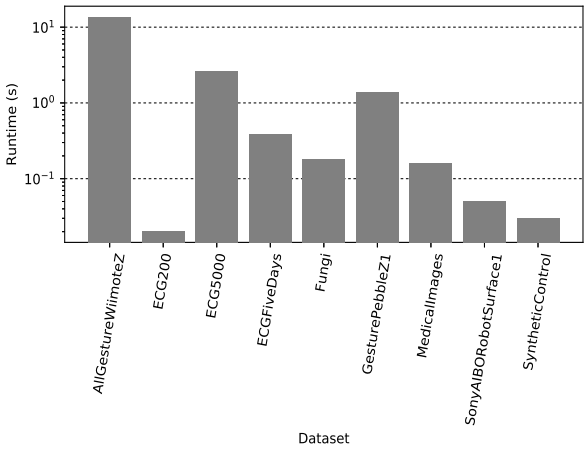


Figure 7: Runtime of ASAX segmentation algorithm for each dataset

for ASAX is due to the variable-size segmentation which created segments in the parts that undergo a significant variation (from time point 44 to 95) as discussed in our motivating example, except that, here, we have 13 segments (allowing ASAX to perform a better distribution of the segments according to information gain). For *SyntheticControl* we can see that the precision of the approximate k-NN search is the same for both ASAX and SAX approaches which is 32%. In this dataset, the shape of the time series is balanced over the time, and the segmentations obtained by ASAX and SAX are the same, resulting in equivalent precision.

These results suggest the advantage of our approach over the state-of-the-art when applied to time series with unbalanced distribution.

4.3 Time cost of ASAX segmentation algorithm

Figure 7 reports the time cost of our proposed approach. It gives the segmentation time of ASAX on the datasets of our experiments. It does not concern SAX since SAX divides the time domain into segments of fixed size which does not require any computation

beforehand. The longest segmentation time is approximately 13 seconds, while the shortest one is around 20 milliseconds. It depends on both the number of time series in the dataset and their length.

5 RELATED WORK

Several techniques have been yet proposed to reduce the dimensionality of time series. Examples of such techniques that can significantly decrease the time and space required for similarity search are: singular value decomposition (SVD) [7], the discrete Fourier transformation (DFT) [1], discrete wavelets transformation (DWT) [3], piecewise aggregate approximation (PAA) [10], random sketches [4], and symbolic aggregate approXimation (SAX) [12].

SAX [12] is one of the most popular techniques for time series representation. It uses a symbolic representation that segments all time series into equi-length segments and symbolizes the mean value of each segment. The symbolic representation allows to lower bound the corresponding distance measures defined on the original time series. Several extensions of SAX have been yet proposed, mainly for improving the similarity search performance via indexing. For example, iSAX [20] is an indexable version of SAX designed for indexing large collections of time series.

iSAX 2.0 [2] proposes a new mechanism and also algorithms for efficient bulk loading and node splitting policy, wich is not supported by iSAX index. In [2], two extensions of iSAX 2.0, namely iSAX 2.0 Clustered and iSAX2+, have been proposed. These extensions focus on the efficient handling of the raw time series data during the bulk loading process, by using a technique that uses main memory buffers to group and route similar time series together down the tree, performing the insertion in a lazy manner.

However, the segmentation in SAX and its extensions is not adaptive to the data since their division principle of the time domain is based on fixed-size segments which is not always effective for unbalanced data distributions. To increase the quality of the approximation for this type of time series we propose our approach ASAX based on SAX representation and a variable-length segmentation algorithm. Our approach is complementary to the SAX extensions, such as iSAX and iSAX 2.0 which have been proposed for improving the kNN queries response time.

6 CONCLUSION

In this paper, we proposed a new approximation technique, called ASAX, that considers the time series distribution on the time domain and performs variable-size segmentation, by using the entropy of symbolic representations. Our technique allows reducing information loss and thus increasing the accuracy of time series representations. We implemented our technique and evaluated its performance using several real world datasets. The experimental results suggest that ASAX can obtain significant performance gains in terms of precision for similarity search compared to SAX. The results show that the more the data distribution in the time domain is unbalanced (non-uniform), the greater is the precision gain of ASAX, *e.g.*, for the *EGCFiveDays* dataset that has a non-uniform distribution in the time domain, the precision of ASAX is 82% compared to 55% for SAX.

REFERENCES

[1] Rakesh Agrawal, Christos Faloutsos, and Arun N. Swami. 1993. Efficient Similarity Search In Sequence Databases. In *Proc. of the 4th Int. Conf. on FODO.*

[2] A. Camerra, J. Shieh, T. Palpanas, T. Rakthanmanon, and E. J. Keogh. 2014. Beyond one billion time series: indexing and mining very large time series collections with i SAX2+. *Knowl. Inf. Syst.* (2014).

[3] Kin-pong Chan and Ada Wai-Chee Fu. 1999. Efficient Time Series Matching by Wavelets. In *Proc. of the ICDE.*

[4] Richard Cole, Dennis Shasha, and Xiaojian Zhao. 2005. Fast Window Correlations over Uncooperative Time Series. In *KDD Conf.* 743–749.

[5] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista, and Hexagon-ML. 2018. The UCR Time Series Classification Archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[6] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. 1994. Fast Subsequence Matching in Time-series Databases. *SigRec* 23, 2 (1994), 419–429. https://doi.org/10.1145/191843.191925

[7] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. 1994. Fast Subsequence Matching in Time-series Databases. In *Proc. of the SIGMOD.*

[8] Pablo Huijse, Pablo A. Estévez, Pavlos Protopapas, Jose C. Principe, and Pablo Zegers. 2014. Computational Intelligence Challenges and Applications on Large-Scale Astronomical Time Series Databases. *IEEE Comp. Int. Mag.* 9, 3 (2014), 27–39.

[9] Kunio Kashino, Gavin Smith, and Hiroshi Murase. 1999. Time-series active search for quick retrieval of audio and video. In *ICASSP.*

[10] Eamonn J. Keogh, Kaushik Chakrabarti, Michael J. Pazzani, and Sharad Mehrotra. 2001. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowl. Inf. Syst.* 3, 3 (2001), 263–286.

[11] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. 2003. A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In *SIGMOD.*

[12] J. Lin, E. Keogh, L. Wei, and S. Lonardi. 2007. Experiencing SAX: A Novel Symbolic Representation of Time Series. *Data Min. Knowl. Discov.* (2007).

[13] Michele Linardi and Themis Palpanas. 2018. ULISSE: ULtra Compact Index for Variable-Length Similarity Search in Data Series. In *ICDE.*

[14] Michele Linardi, Yan Zhu, Themis Palpanas, and Eamonn J. Keogh. 2018. Matrix Profile X: VALMOD - Scalable Discovery of Variable-Length Motifs in Data Series. In *SIGMOD.*

[15] Michele Linardi, Yan Zhu, Themis Palpanas, and Eamonn J. Keogh. 2018. VALMOD: A Suite for Easy and Exact Detection of Variable Length Motifs in Data Series. In *SIGMOD.*

[16] Themis Palpanas. 2015. Data Series Management: The Road to Big Sequence Analytics. *SIGMOD Record* 44, 2 (2015), 47–52.

[17] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. 2012. Searching and Mining Trillions of Time Series Subsequences Under Dynamic Time Warping. In *KDD.*

[18] Usman Raza, Alessandro Camerra, Amy L. Murphy, Themis Palpanas, and Gian Pietro Picco. accepted for publication, 2015. Practical Data Prediction for Real-World Wireless Sensor Networks. *IEEE Trans. Knowl. Data Eng.* (accepted for publication, 2015). https://doi.org/10.1109/TKDE.2015.2411594

[19] Dennis Shasha. 1999. Tuning Time Series Queries in Finance: Case Studies and Recommendations. *IEEE Data Eng. Bull.* 22, 2 (1999), 40–46.

[20] J. Shieh and E. Keogh. 2008. iSAX: Indexing and Mining Terabyte Sized Time Series. In *KDD Conf.* 623–631.

[21] S. Soldi, Volker Beckmann, Wayne H. Baumgartner, Gabriele Ponti, Chris R. Shrader, P. Lubinski, H. A. Krimm, F. Mattana, and Jack Tueller. 2014. Long-term variability of AGN at hard X-rays. *Astronomy and Astrophysics - A&A* 563, A57 (March 2014), 16. https://doi.org/10.1051/0004-6361/201322653

[22] Lexiang Ye and Eamonn J. Keogh. 2009. Time series shapelets: a new primitive for data mining. In *KDD.*

[23] Kostas Zoumpatianos and Themis Palpanas. 2018. Data Series Management: Fulfilling the Need for Big Sequence Analytics. In *ICDE.*