Efficient Algorithms for Knowledge Discovery from Time Series

ABSTRACT

Matrix profile is an efficient technique for knowledge extraction from time series, *e.g.*, motif and anomaly detection. Several algorithms have been yet proposed for computing it, *e.g.*, STAMP, STOMP and SCRIMP++. All these algorithms use the z-normalized Euclidean distance to measure the distance between subsequences. However, as we illustrate in this paper, for some datasets the non-normalized (classical) based matrix profile is more useful. Thus, efficient matrix profile techniques based on both z-normalized and non-normalized distances are necessary for knowledge extraction from different time series datasets.

In this paper, we propose such efficient techniques. We first propose an efficient algorithm called AAMP for computing matrix profile with the non-normalized Euclidean distance. Then, we extend our algorithm for the p-norm distance. We also propose two algorithms called ACAMP and ACAMP-Optimized that use the same principle as AAMP, but for calculating matrix profile by using z-normalized Euclidean distance. We implemented and evaluated the performance of our algorithms through experiments over real world datasets. The results illustrate that AAMP is very efficient for computing matrix profile for non-normalized Euclidean distances. They also illustrate that the ACAMP-Optimized algorithm is significantly faster than the state of the art matrix profile algorithms for the case of z-normalized Euclidean distance.

KEYWORDS

Matrix Profile, STAMP, STOMP, Similarity Search, Motifs & Discords Discovery.

1 INTRODUCTION

Matrix profile has been recently proposed as an efficient technique to the problem of all-pairs-similarity search in time series [1–8]. Given a time series T and a subsequence length m, the matrix profile returns for each subsequence, the distance to the most similar subsequence in the time series. It is itself a very useful time series for data analysis, *e.g.*, detecting the motifs (represented by low values), or anomalies (represented by high values), etc.

Recently, efficient algorithms have been proposed for matrix profile computation, *e.g.*, STAMP [1], STOMP [2] and SCRIMP++ [8]. All these algorithms use the z-normalized Euclidean distance to measure the distance between subsequences. They are based on a technique, named as *Mueen's Algorithm for Similarity Search (MASS)* [9] for efficient calculation of *z-normalized Euclidean distance*, by exploiting the *Fast Fourier Transform (FFT)*. The z-normalized Euclidean distance formula used in the MASS algorithm is derived from *Pearson correlation* which works only for computing z-normalized Euclidean distance, and makes it inappropriate for computing classical Euclidean distance.

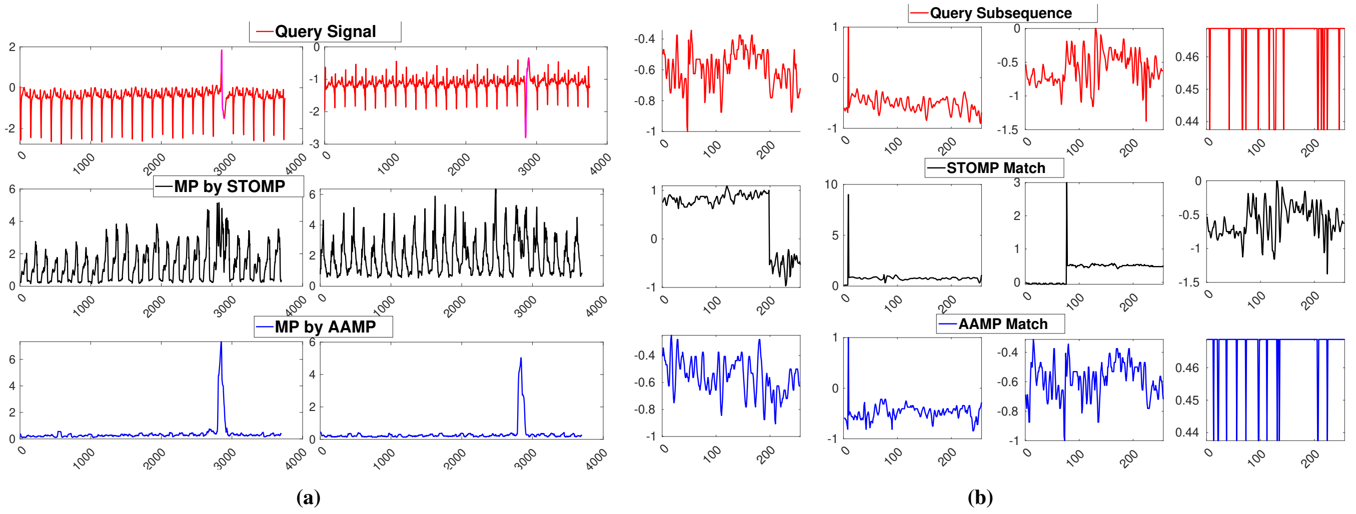
However, we observed that for some datasets, the non-normalized (classical) Euclidean distance is more useful for knowledge discovery. In fact, in some cases the *z-normalization* can remove rare and important information. As an example, consider Fig. 1a (top), which shows two time series from the real ECG dataset. In Fig. 1a (middle) and (bottom), we see the matrix profiles generated for the two time series by considering z-normalized (using *STOMP* algorithm) and non-normalized Euclidean (using our *AAMP*) distances respectively. In this example, the matrix profiles generated using the *z-normalized* distance loose the information about the anomalies (marked by magenta color in Fig. 1a top.). But, the matrix profile calculated by using non-normalized Euclidean distance can clearly highlight those anomalies.

In addition, the z-normalized Euclidean distance does not necessarily provide the nearest neighbors (matches) of the subsequences from the same range of values. Hence, the match of a subsequence can come from completely different range of values and in some applications these matches could be considered as irrelevant. An example is depicted in Fig. 1b, where we show the matches for four query subsequences, taken from the time series of a real sheep dataset, representing different activities like RUNNING and WALKING (see detail of the dataset in Section 5.1.1). It is clearly visible that our proposed *AAMP* algorithm that uses the non-normalized Euclidean distance is capable of returning matches that are in the same range of values as the query subsequences. In Fig. 1b, we only have shown few selective examples among several others, where by using non-normalized Euclidean distance, we found better matches.

In fact, the *z-normalized Euclidean distance based matrix profile* is able to find the shape-wise matches from any range of values and that's why the shape-wise similarity could be found irrespective of the numerical values. This is an advantage for some applications, but a disadvantage for others (*i.e.*, those that need the matches from the same range). This is why, a combination of both z-normalized and non-normalized based matrix profiles is necessary for knowledge extraction in a wide range of applications.

In this paper, we provide efficient techniques for the calculation of matrix profile for both *z-normalized* and *non-normalized distances*. Our contributions are as following:

- We propose an efficient algorithm called AAMP for computing matrix profile with the non-normalized Euclidean distance. AAMP is executed in a set of iterations, such that in each iteration the distance of subsequences is incrementally computed. We also extend AAMP to compute matrix profile for the *p-norm* distance that is more general than the Euclidean distance which is actually a *2-norm* distance.
- We propose an algorithm called ACAMP that uses the same principle as AAMP but for the *z-normalized* Euclidean distance. In ACAMP, we use an incremental formula for computing the *z-normalized* distance that is based on some variables, calculated incrementally in a sliding window that moves over the subsequences of the time series. We also propose an improved version of the ACAMP algorithm, called ACAMP-optimized, that is significantly faster than ACAMP.
- We implemented our algorithms and compared them with the state of the art algorithms on matrix profile, *i.e.*, STOMP,



(a)

(b)

Figure 1: a) Top: example of two different time series from ECG dataset; Middle: matrix profile generated by z-normalized Euclidean distance using STOMP algorithm; Bottom: matrix profile generated by non-normalized Euclidean distance using our AAMP algorithm. b) Top: four subsequences of length 50 from sheep dataset; Middle: the nearest neighbors obtained by STOMP; Bottom: the nearest neighbors obtained by AAMP are in the same range as the queries, while the results obtained by STOMP are in very different ranges.

SCRIMP and SCRIMP++, using several real world datasets. The results show excellent performance gains. They show that AAMP and ACAMP-optimized are significantly faster than the state-of-the-art algorithms for matrix profile computation. They also illustrate the utility of detecting discords/outliers in datasets by using AAMP based on the non-normalized Euclidean distance over STOMP, SCRIMP and SCRIMP++ that are based on the *z-normalized* Euclidean distance.

It is worth mentioning that our algorithms, *i.e.*, AAMP and ACAMP, are exact, anytime and incrementally maintainable. They take a deterministic execution time that only depends on the time series and subsequence length.

The rest of this paper is organized as follows. In Section 2, we give the problem definition. In Section 3, we describe our AAMP algorithm for computing matrix profile with non-normalized Euclidean and p-norm distances. In Section 4, we propose the ACAMP algorithm for z-normalized distance. Section 5 presents the experimental results. Section 6 discusses related work and Section 7 concludes the article.

2 PROBLEM DEFINITION

In this section, we give the formal definition of the matrix profile, and describe the problem which we address in this article.

Definition 2.1. A *time series* T is a sequence of real-valued numbers $T = \langle t_1, \ldots, t_n \rangle$ where n is the length of T.

A subsequence of a time series is defined as follows.

Definition 2.2. Let m be a given integer value such that $1 \leq m \leq n$. A *subsequence* $T_{i,m}$ of a time series T is a continuous sequence of values in T of length m, starting from position i. Formally, $T_{i,m} = \langle t_i, \ldots, t_{i+m-1} \rangle$ where $1 \leq i \leq n - m + 1$. We denote i as the start position of $T_{i,m}$ subsequence.

For each subsequence of a time series, we can compute its distance to all subsequences of the same length in the same time series. This is called a distance profile.

Definition 2.3. Given a query subsequence $T_{i,m}$, a *distance profile* D_i of $T_{i,m}$ in the time series T is a vector of the distances between $T_{i,m}$ and each subsequence of length m in time series T. Formally, $D_i = \langle d_{i,1}, \ldots, d_{i,n-m+1} \rangle$, where $d_{i,j}$ is the distance between $T_{i,m}$ and $T_{j,m}$.

Note that the term *distance* in Definition 2.3 does not refer to the mathematical definition of *distance*. It only gives a measure on the difference between two subsequences. For instance the z-normalized Euclidean distance does not satisfy the (mathematical) axioms of a distance. A *matrix profile* is a vector that represents the minimum distance between each subsequence and all other subsequences of a time series T.

Definition 2.4. Given a subsequence length m, the *matrix profile* of a time series T is a vector $P = \langle p_1, \ldots, p_{n-m+1} \rangle$ such that p_i is the minimum distance between the subsequence $T_{i,m}$ and all other subsequence of T, for $1 < i < n - m + 1$. In other words, $p_i = min(D_i)$, *i.e.*, p_i is the minimum value in the distance profile of $T_{i,m}$.

We are interested in the efficient computation of matrix profile using following three different distance measures: 1) Euclidean distance; 2) p-norm distance that is a generalization of Euclidean distance; 3) z-normalized Euclidean distance.

Definition 2.5. The *Euclidean distance* between two subsequences $T_{i,m}$ and $T_{j,m}$ is defined as:

$$D_{i,j} = \sqrt{\sum_{l=0}^{m-1} (t_{i+l} - t_{j+l})^2} \tag{1}$$

In this paper, sometimes we call the Euclidean distance as *non-normalized (classical) Euclidean distance*.

Definition 2.6. Let $p > 1$ be a positive integer, then the *p-norm distance* between two subsequences $T_{i,m}$ and $T_{j,m}$ is defined as:

$$DP_{i,j} = \sqrt[p]{\sum_{l=0}^{m-1} (t_{i+l} - t_{j+l})^p} \quad (2)$$

The z-normalized Euclidean distance between two subsequences is defined as follows.

Definition 2.7. Let μ_i and μ_j be the mean of the values in two subsequences $T_{i,m}$ and $T_{j,m}$ respectively. Also, let σ_i and σ_j be the standard deviation of the values in $T_{i,m}$ and $T_{j,m}$ respectively. Then, the *z-normalized Euclidean distance* between $T_{i,m}$ and $T_{j,m}$ is defined as:

$$DZ_{i,j} = \sqrt{\sum_{l=0}^{m-1} \left(\frac{t_{i+l} - \mu_i}{\sigma_i} - \frac{t_{j+l} - \mu_j}{\sigma_j} \right)^2} \quad (3)$$

A shapelet is a subsequences that can maximally represent the class of a time series. The matrix profile can be used for shapelet detection (see Section S.2). Let us define the *joint matrix profile* of two time series that is needed for explaining the shapelet discovery using matrix profile.

Definition 2.8. Let m be the subsequence length, and A and B be two time series of length n. The *joint matrix profile* of A with B is a vector $P_{AB} = \langle p_1, \ldots, p_{n-m+1} \rangle$ such that p_i is the minimum distance between the subsequence $A_{i,m}$ and all subsequence of time series B.

3 AAMP

In this section, we propose the AAMP algorithm for computing matrix profile by using the Euclidean distance. At first, we present the formula for incremental computation of the Euclidean distance and then propose the AAMP algorithm which uses this formula for computing matrix profile.

3.1 Incremental Computation of Euclidean Distance

Here, we present a formula that allows us to compute the Euclidean distance between two subsequences $T_{i,m}$ and $T_{j,m}$ based on the Euclidean distance of subsequences $T_{i-1,m}$ and $T_{j-1,m}$. The formula is presented by the following lemma.

Lemma 1. Let $D_{i,j}$ be the Euclidean distance between two subsequences $T_{i,m}$ and $T_{j,m}$. Let $D_{i-1,j-1}$ be the Euclidean distance between two subsequences $T_{i-1,m}$ and $T_{j-1,m}$. Then $D_{i,j}$ can be computed as:

$$D_{i,j} = \sqrt{D^2_{i-1,j-1} - (t_{i-1} - t_{j-1})^2 + (t_{i+m-1} - t_{j+m-1})^2} \quad (4)$$

Proof. Let $T_{i,m} = \langle t_i, t_{i+1}, \ldots, t_{i+m-1} \rangle$ and $T_{j,m} = \langle t_j, t_{j+1}, \ldots, t_{j+m-1} \rangle$. Then the square of the Euclidean distance between $T_{i,m}$ and $T_{j,m}$ is computed as:

$$D^2_{i,j} = \sum_{l=0}^{m-1} (t_{i+l} - t_{j+l})^2 \quad (5)$$

And the square of the Euclidean distance between $T_{i-1,m}$ and $T_{j-1,m}$ is:

$$D^2_{i-1,j-1} = \sum_{l=0}^{m-1} (t_{i-1+l} - t_{j-1+l})^2 \quad (6)$$

By comparing Equations (5) and (6), we have:

$$D^2_{i,j} = D^2_{i-1,j-1} - (t_{i-1} - t_{j-1})^2 + (t_{i+m-1} - t_{j+m-1})^2 \quad (7)$$

Thus, we have:

$$D_{i,j} = \sqrt{D^2_{i-1,j-1} - (t_{i-1} - t_{j-1})^2 + (t_{i+m-1} - t_{j+m-1})^2} \quad (8)$$

By using the above equation, we can compute the Euclidean distance $D_{i,j}$ by using the distance $D_{i-1,j-1}$ in $O(1)$.

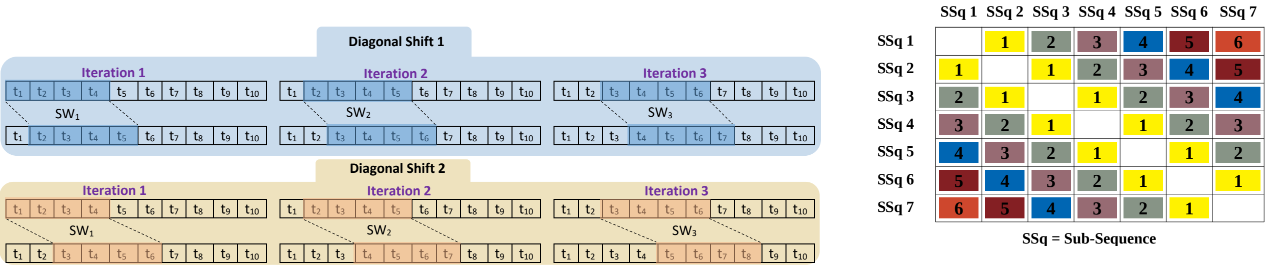
3.2 Algorithm

The main idea behind AAMP is that for computing the distance between subsequences, it uses *diagonal sliding windows*, such that in each sliding window, the Euclidean distance is computed only between the subsequences that have a precise difference in their *starting positions*. These sliding windows allow us to use Equation (4) for efficient distance computation.

Algorithm 1 shows the pseudo-code of AAMP (for now, ignore the violet colored lines). Initially, the algorithm sets all values of the *matrix profile array* to infinity (*i.e.*, maximum distance) and the *matrix profile index* array to 1. Then, it performs $n - m$ *iterations* using a variable k ($1 \le k \le n - m$). In each iteration of k, the algorithm calculates distance between i^{th} subsequence (i.e. $T_{i,m}$) and the subsequence which is k positions apart from it, *i.e.*, $T_{i+k,m+k}$. The value of i is primarily taken as 1 then it iterates from 2 to $n - m + 1 - k$ values in Line 13.

In each iteration k, AAMP firstly computes the Euclidean distance of the 1^{st} subsequence of the time series, *i.e.*, $T_{1:m}$, with the one that starts at k positions from it, *i.e.* subsequence $T_{k+1:m+k}$ The first distance computation is done using the classical formula of Euclidean distance, *i.e.* using Equation (1) (see Line 6). Then, in a sliding window, the algorithm incrementally computes the distance of other subsequences with the subsequences that are k position apart from them (i.e. 2^{nd} with 3^{rd} subsequence, 3^{rd} with 4^{th} subsequence etc.), and this is done in $O(1)$ time. If the computed distance is smaller than the existing distance value in the matrix profile array P, then the smaller distance is saved in the matrix profile along with it's index (see Lines $7 - 12$ and $15 - 20$). Note that, we use the property that the distance between i^{th} and j^{th} subsequences is equal to the distance between j^{th} and i^{th} subsequences; i.e. $dist_{i,j} = dist_{j,i}$ (see Lines $8 - 9$ & $11 - 12$; and Lines $16 - 17$ & $19 - 20$). In AAMP, we use square of the Euclidean distances for comparing the distances of different subsequences (see Lines 6 and 14), and at the end of the algorithm, square of these distances is replaced by taking the *sqrt* to obtain the real distances in the matrix profile (see Line 22). This reduces the number of *sqrt* operations done during the execution of the algorithm.

Example 1. Figure 2a shows an example of executing AAMP over a time series of length $n = 10$ and for subsequences of length $m = 4$. In *iteration* 1, the first Euclidean distance is calculated between $T_{1,m}$ and $T_{2,m}$ and the sliding window $SW1$. Then the sliding window moves to the next subsequences (i.e. sliding window $SW2$), and incrementally computes the distance between $T_{2,m}$ and $T_{3,m}$ by using



	SSq 1	SSq 2	SSq 3	SSq 4	SSq 5	SSq 6	SSq 7
SSq 1		1	2	3	4	5	6
SSq 2	1		1	2	3	4	5
SSq 3	2	1		1	2	3	4
SSq 4	3	2	1		1	2	3
SSq 5	4	3	2	1		1	2
SSq 6	5	4	3	2	1		1
SSq 7	6	5	4	3	2	1	

(a)

(b)

Figure 2: a) Example of AAMP execution on a time series of length $n = 10$, and with subsequence length $m = 4$. The total number of subsequences is $n - m + 1 = 10 - 4 + 1 = 7$. In iteration k, the distances between the subsequences that are k positions apart from each other are computed. The first distance in each iteration is computed using the normal Euclidean distance function in O(m), and the other distances are computed incrementally in a constant time. b) The subsequences are arranged in a matrix to better understand the functioning of AAMP algorithm. By looking at the cells of the matrix, we can see in which iteration, the distance of two subsequences is calculated. Different iterations are represented by different colors.

the Equation (4) in $O(1)$ time. Then, the sliding window moves to the next subsequences and computes their distances, *i.e.*, $T_{3,m}$ and $T_{4,m}$. This distances computation process continues for all the subsequence pairs, which are 1 element/index apart from each other's starting positions. For *iteration* 1, the distances computed between all the subsequence pairs are marked by yellow color in the matrix shown in Fig. 2b.

In *iteration* 2, the Euclidean distance is computed between each subsequence and the one which is 2 elements/indexes apart (follow the bottom image in Fig. 2a). Thus, we calculate the distances between subsequence 1 & 3 followed by the distance between subsequence 2 and 4 etc. (shown by black colored cells in the matrix of Fig. 2b). Note that, in each iteration the first distance is computed using the classical Euclidean distance formula and the other distances are computed by using the incremental formula.

By looking at the cells of the matrix in Fig. 2b, we can see in which iteration, the distance of two subsequences is calculated. Different iterations are represented by different colors.

3.3 Complexity Analysis

The AAMP algorithm contains two loops. In the 1^{st} loop (Line 6), the distance between $T_{1,m}$ and $T_{k,m}$ is computed by using the normal Euclidean distance function in $O(m)$ time, thus in total, Line 6 is executed in $O(m \times (n - m))$. In the nested loop (Lines $13 - 20$), all operations are done in $O(1)$, so in total these operations are done in $O((n - m)^2)$. Thus, the time complexity is $O((n - m)^2) + m \times (n - m))$ which is equivalent of $O(n \times (n - m))$. If $n >> m$, then the time complexity of AAMP can be written as $O(n^2)$. But, if m is very close to n, *i.e.*, $m = n - c$ for any small constant c, then the time complexity is $O(n)$. The space needed for our algorithm is only the array of matrix profile and some simple variables. Thus, the space complexity is $O(n)$.

3.4 Extension of AAMP to p-Norm Distance

In this section, we extend the AAMP algorithm to the p-norm distance that is a more general form of distance computation than Euclidean distance formula. The p-norm functions are used in *Lebesgue spaces* (L^P), which are useful in data analysis in physics, statistics, finance, engineering, etc.

Let $T_{i,m}$ and $T_{j,m}$ be two time series subsequences, then their p-norm distance (for $p > 1$) is defined as:

$$DP_{i,j} = \sqrt[p]{\sum_{l=0}^{m-1} (t_{i+l} - t_{j+l})^p} \tag{9}$$

Notice that the Euclidean distance is a special case of p-norm with $p = 2$. The following lemma gives an incremental formula for computing $PNORM_{i,j}$.

Lemma 2. Let $DP_{i,j}$ be the p-norm distance of subsequences $T_{i,m}$ and $T_{j,m}$. Then, $DP_{i,j}$ can be computed by using the p-norm distance of subsequences $T_{i-1,m}$ and $T_{j-1,m}$, denoted by $DP_{i-1,j-1}$, as:

$$DP_{i,j} = \sqrt[p]{(DP_{i-1,j-1})^p - (t_{i-1} - t_{j-1})^p + (t_{i+m-1} - t_{j+m-1})^p}$$

Proof. The proof can be easily done in a similar way as that of Lemma 1. Using Lemma 2, we can modify the AAMP algorithm to compute the matrix profile with the p-norm distance. This can be done just by modifying two lines in Algorithm 1: i) in Line 6 we replace the Euclidean distance with *p-norm* distance between the subsequences; i.e. $T_{1,m}$ and $T_{k,m}$; ii) in Line 14, we incrementally compute the *p-norm* distance using Lemma 2.

The time and space complexity of the AAMP algorithm for *p-norm* is the same as that of AAMP with the Euclidean distance.

4 ACAMP: MATRIX PROFILE FOR Z-NORMALIZED EUCLIDEAN DISTANCE

In this section, we propose an algorithm, called ACAMP, that computes matrix profile based on the z-normalized Euclidean distance and using the similar principle as AAMP, *i.e.*, incremental distance computation by using diagonal sliding windows.

Algorithm 1: AAMP algorithm: matrix profile with Euclidean distance

```
Input: T: time series; n: length of time series; m:
       subsequence length
Output: P: Matrix profile; I: Matrix profile Indexes;
begin
    for i=1 to n-m+1 do
        P[i] = ∞            ▹ initialize the matrix profile
        I[i] = 1        ▹ initialize the matrix profile indexes
    for k=1 to n-m do
        $dist = Euc\_Distance(T_{1:m}, T_{k+1:m+k})^2$        ▹ compute
          square of the distance between $1^{st}$ i.e. $T_{1:m}$ and $(k+1)^{th}$
          i.e. $T_{k+1:m+k}$ subsequences
        if $dist < P[1]$ then
            $P[1] = dist$
            $I[1] = k + 1;$
        if $dist < P[k+1]$ then
            $P[k+1] = dist$
            $I[k+1] = 1$
        // if $k + 1 == n - m + 1$ then
            // $\mathcal{B}[1] = dist$       ▹ if we are computing the distance
              between $1^{st}$ and last sub-sequence
        for $i$=2 to $(n - m + 1 - k)$ do
            $dist = (dist - (t_{i-1} - t_{i-1+k})^2 + (t_{i+m-1} - t_{i+m+k-1})^2$
            if $dist < P[i]$ then
                $P[i] = dist$
                $I[i] = k + i$
            if $dist < P[i+k]$ then
                $P[i+k] = dist$
                $I[i+k] = i$
            // if $i + k == n - m + 1$ then
                // $\mathcal{B}[1,1] = dist$        ▹ if we are computing the
                  distance with last sub-sequence
    for i=1 to n-m+1 do
        $P[i] = \sqrt{P[i]}$
```

4.1 Incremental Computation of Z-Normalized Euclidean Distance

Let us now explain how ACAMP computes the z-normalized Euclidean distance incrementally. Let $T_{i,m} = \langle t_i, \ldots, t_{i+m-1} \rangle$ and $T_{j,m} = \langle t_j, \ldots, t_{j+m-1} \rangle$ be two subsequences of a time series T. In ACAMP, we compute the z-normalized Euclidean distance between $T_{i,m}$ and $T_{j,m}$ by using the following five variables:

- $A_i = \sum_{l=0}^{m-1} t_{i+l}$: the sum of the values in $T_{i,m}$;
- $B_j = \sum_{l=0}^{m-1} t_{j+l}$: the sum of the values in $T_{j,m}$;
- $\mathbf{A_i} = \sum_{l=0}^{m-1} t_{i+l}^2$: the sum of the square of values in $T_{i,m}$;
- $\mathbf{B_j} = \sum_{l=0}^{m-1} t_{j+l}^2$: the sum of the square of values in $T_{j,m}$;
- $\mathbf{C_{i,j}} = \sum_{l=0}^{m-1} t_{i+l} \times t_{j+l}$: the product of values of $T_{i,m}$ and $T_{j,m}$.

Note that all above variables can be computed incrementally, when moving a sliding window from $T_{i,m}$ to $T_{i+1,m}$. Given these variables, the z-normalized Euclidean distance between two subsequences $T_{i,m}$ and $T_{j,m}$ can be computed using the formula given by the following lemma.

Lemma 3. Let $DZ_{i,j}$ be the z-normalized distance of subsequences $T_{i,m}$ and $T_{j,m}$. Then, $DZ_{i,j}$ can be computed as:

$$DZ_{i,j} = \sqrt{2m\left(1 - \frac{\mathbf{C_{i,j}} - \frac{1}{m}A_iB_j}{\sqrt{\left(\mathbf{A_i} - \frac{1}{m}A_i^2\right)\left(\mathbf{B_j} - \frac{1}{m}B_j^2\right)}}\right)} \tag{10}$$

Proof. Let μ_i and μ_j be the mean of the values in the sequences $T_{i,m}$ and $T_{j,m}$ respectively. Also, let σ_i and σ_j be the standard deviation of the values in the subsequences $T_{i,m}$ and $T_{j,m}$ respectively. Then, the z-normalized Euclidean distance between $T_{i,m}$ and $T_{j,m}$ is defined as:

$$DZ_{i,j} = \sqrt{\sum_{l=1}^{m-1}\left(\frac{t_{i+l} - \mu_i}{\sigma_i} - \frac{t_{j+l} - \mu_j}{\sigma_j}\right)^2} \tag{11a}$$

$$\mu_i = \frac{1}{m}\sum_{l=0}^{m-1} t_{i+l}; \ \mu_j = \frac{1}{m}\sum_{l=0}^{m-1} t_{j+l} \tag{11b}$$

$$\sigma_i = \sqrt{\frac{1}{m}\sum_{l=0}^{m-1} t_{i+l}^2 - (\mu_i)^2}; \ \sigma_j = \sqrt{\frac{1}{m}\sum_{k=0}^{m-1} t_{j+l}^2 - (\mu_j)^2}. \tag{11c}$$

We can write the square of DZ as following:

$$\begin{aligned} DZ_{i,j}^2 &= \sum_{l=0}^{m-1}\left(\frac{t_{i+l} - \mu_i}{\sigma_i} - \frac{t_{j+l} - \mu_j}{\sigma_j}\right)^2 \\ &= \sum_{l=0}^{m-1} (\left(\frac{t_{i+l} - \mu_i}{\sigma_i}\right)^2 - \\ & 2\left(\frac{t_{i+l} - \mu_i}{\sigma_i}\right)\left(\frac{t_{j+l} - \mu_j}{\sigma_j}\right) + \left(\frac{t_{j+l} - \mu_j}{\sigma_j}\right)^2) \\ &= \sum_{l=0}^{m-1} (\frac{t_{i+l}^2 - 2t_{i+l}\mu_i + (\mu_i)^2}{(\sigma_i)^2} - \\ & 2\left(\frac{t_{i+l}t_{j+l} - \mu_i t_{j+l} - t_{i+l}\mu_j + \mu_j\mu_i}{\sigma_i\sigma_j} \right) + \\ & \left.\frac{t_{j+l}^2 - 2t_{j+l}\mu_j + (\mu_j)^2}{(\sigma_j)^2}\right) \end{aligned} \tag{12}$$

Let:

$A_i = \sum_{l=0}^{m-1} t_{i+l}$; $B_j = \sum_{l=0}^{m-1} t_{j+l}$; $\mathbf{A}_i = \sum_{l=0}^{m-1} t_{i+l}^2$; $\mathbf{B}_j = \sum_{l=0}^{m-1} t_{j+l}^2$; $\mathbf{C_{i,j}} = \sum_{l=0}^{m-1} t_{i+l}t_{j+l}$.

Then, we have:

$\mu_i = \frac{1}{m}A_i$; $\mu_j = \frac{1}{m}B_j$; $(\sigma_i)^2 = \frac{1}{m}\mathbf{A_i} - \frac{1}{m^2}A_i^2$; $(\sigma_j)^2 = \frac{1}{m}\mathbf{B}_j - \frac{1}{m^2}B_j^2$.

Then, the square of the z-normalized Euclidean distance can be written as:

$$
\begin{aligned}
DZ_{i,j}^2 = \sum_{l=0}^{m-1}\Bigg(& \frac{t_{i+l}^2 - 2t_{i+l}\mu_i + (\mu_i)^2}{(\sigma_i)^2} \\
& - 2\left(\frac{t_{i+l}b_{j+l} - \mu_i t_{j+l} - t_{i+l}\mu_j + \mu_j\mu_i}{\sigma_i\sigma_j}\right) + \\
& \frac{t_{j+l}^2 - 2t_{j+l}\mu_j + (\mu_j)^2}{(\sigma_j)^2}\Bigg) \\
&= \frac{\mathbf{A}_i - 2A_i^2\frac{1}{m} + \frac{A_i^2}{m}}{\frac{1}{m}\mathbf{A}_i - \frac{1}{m^2}A_i^2} \\
& - 2 \times \frac{\mathbf{C_{i,j}} - \frac{2}{m}A_iB_j + \frac{A_iB_j}{m}}{\sqrt{(\frac{1}{m}\mathbf{A}_i - \frac{1}{m^2}A_i^2)(\frac{1}{m}\mathbf{B}_j - \frac{1}{m^2}B_j^2)}} + \\
& \frac{\mathbf{B}_j - 2B_j^2\frac{1}{m} + \frac{B_j^2}{m}}{\frac{1}{m}\mathbf{B}_j - \frac{1}{m^2}B_j^2} \\
&= 2m - 2 \times \frac{m^2\mathbf{C_{i,j}} - mA_iB_j}{\sqrt{(m\mathbf{A}_i - A_i^2)(m\mathbf{B}_j - B_j^2)}} \\
&= 2m\left(1 - \frac{\mathbf{C_{i,j}} - \frac{1}{m}A_iB_j}{\sqrt{(\mathbf{A}_i - \frac{1}{m}A_i^2)(\mathbf{B}_j - \frac{1}{m}B_j^2)}}\right)
\end{aligned}
\tag{13}
$$

4.2 Algorithm

The pseudo-code of ACAMP is shown in Algorithm 2. In Line 4 in a loop, k is iterated from 1 to $n - m$, and in each iteration the z-normalized Euclidean distance is calculated between the subsequences which are k points far from each other in the time series (Lines 5 to 14). In each iteration, the distances are computed by using the formula of Equation 10 that uses the five variables *i.e.*, A, B, $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. For each iteration of k, the distance between two initial subsequence is calculated (i.e. the distance between $T_{1,m}$ and $T_{1+k,m}$), by using the five variables in $O(m)$ time (see Lines 5 to 10). For the other subsequences, these variables and the distance are incrementally computed in $O(1)$ time. Note that in the algorithm, for performance reasons we compare the square of the z-normalized Euclidean distance of the subsequences (Line 10 and 21). At the end of the algorithm (Lines 26 to 27), in a loop we convert the square distances to the real distances.

The time and space complexity of ACAMP algorithm is same as that of of AAMP algorithm, described in Section 3.3.

4.3 More Optimization of ACAMP

In the following section, we propose several optimizations for the ACAMP Algorithm.

One possible optimization is to move the first calculation of variables A, $\mathbf{A}$, B, and $\mathbf{B}$ (actually done in Lines 7 to 10) before the loop (*i.e.*, before Line 4). By doing this, firstly, we can avoid the redundant computation of A & $\mathbf{A}$ and B and $\mathbf{B}$. Then the calculation of distance between the 1^{st} and all other subsequences can be pre-computed. Hence, we would just need to incrementally update these variables in the loop (Lines 16 – 20).

Algorithm 2: ACAMP algorithm: matrix profile calculation with z-normalized Euclidean distance

```
   Input: T: time series; n: length of time series; m:
          subsequence length
   Output: P: Matrix profile; I: Matrix profile Indexes;
 1 begin
 2     for i=1 to n-m+1 do
 3         P[i] = ∞; I[i] = 1
 4     for k=1 to n-m do
 5         $A = \sum_{l=0}^{m-1} t_{1+l}$   ▷ sum of the values in $T_{1,m}$
 6         $B = \sum_{l=0}^{m-1} t_{1+k+l}$   ▷ sum of the values in $T_{1+k,m}$
 7         $\mathbf{A} = \sum_{l=0}^{m-1} t_{1+l}^2$ ▷ sum of the square of values in $T_{1,m}$
 8         $\mathbf{B} = \sum_{l=0}^{m-1} t_{1+k+l}^2$   ▷ sum of the square of values in
             $T_{1+k,m}$
 9         $\mathbf{C} = \sum_{l=0}^{m-1} t_{1+l}t_{k+l}$   ▷ product of values of $T_{1,m}$ and
             $T_{1+k,m}$
10         $dist = 2m\left(1 - \frac{\mathbf{C} - \frac{1}{m}AB}{\sqrt{(\mathbf{A} - \frac{1}{m}A^2)(\mathbf{B} - \frac{1}{m}B^2)}}\right)$   ▷ compute the
             square of z-normalized distance
11         if $dist < P[1]$ then
12             $P[1] = dist; I[1] = k + 1;$
13         if $dist < P[k + 1]$ then
14             $P[k + 1] = dist; I[k + 1] = 1$
15         for i=2 to $n - m + 1 - k$ do
16             $A = A - t_{i-1} + t_{i+m-1};$
17             $B = B - t_{i-1+k} + t_{i+m+k-1};$
18             $\mathbf{A} = \mathbf{A} - t_{i-1}^2 + t_{i+m-1}^2;$
19             $\mathbf{B} = \mathbf{B} - t_{i-1+k}^2 + t_{i+m+k-1}^2;$
20             $\mathbf{C} = \mathbf{C} - t_{i-1} \times t_{i-1+k} + t_{i+m-1} \times t_{i+m+k-1};$
21             $dist = 2m\left(1 - \frac{\mathbf{C} - \frac{1}{m}AB}{\sqrt{(\mathbf{A} - \frac{1}{m}A^2)(\mathbf{B} - \frac{1}{m}B^2)}}\right)$
22             if $dist < P[i]$ then
23                 $P[i] = dist; I[i] = k + i$
24             if $dist < P[i + k]$ then
25                 $P[k + i] = dist; I[k + i] = i$
26     for i=1 to n do
27         $P[i] = \sqrt{P[i]}$   ▷ compute the z-normalized distance
             from its square
```

We can further optimize ACAMP by not comparing the square of z-normalized distance in Lines 15, 17, 26 and 28 in Algorithm 2, but by comparing $F_{i,j}$ defined as follows:

$$
F_{i,j} = \frac{(A_iB_j - m\mathbf{C_{i,j}}) \times |A_iB_j - m\mathbf{C_{i,j}}|}{(\mathbf{A}_i - \frac{1}{m}A_i^2)(\mathbf{B}_j - \frac{1}{m}B_j)}, \tag{14}
$$

We can easily show that $DZ_{i,j} > DZ_{i,k}$ if and only if $F_{i,j} > F_{i,k}$. In the formula of $F_{i,j}$, there is no square root operation, and its computation takes less time than that of $DZ_{i,j}$. Thus, for comparing the *z-normalized* Euclidean distance of subsequences, we can simply compare their $F_{i,j}$. Then in Line 21 of the algorithm, the following equation can be used for computing the z-normalized Euclidean distance $DZ_{i,j}$ from $F_{i,j}$:

$$DZ_{i,j} = 2m + 2 \times \text{sign}(F_{i,j}) \times \sqrt{|F_{i,j}|} \quad (15)$$

5 PERFORMANCE EVALUATION

In this section, we compare the execution time of our algorithms AAMP and ACAMP with the state-of-the-art matrix profile algorithms STOMP, SCRIMP and SCRIMP++ [8]. We also evaluate the optimized version of ACAMP (using the optimizations proposed in Section 4.3) called as *ACAMP-Optimized*. We first describe the experimental setup, the datasets used for the performance evaluation and then present the results of the experiments.

5.1 Setup

We implemented our algorithms in MATLAB [1]. For STOMP[2] [3], SCRIMP[4] and Scrimp++[4], we used the Matlab code available from [10] using the step size of PreSCRIMP = 0.25. The evaluation and tests were carried out on a off-the-shelf computer with Intel ®Core(TM) ™i7-8850H CPU @ 2.60 GHz ×8 processors, on Ubuntu 18.04 LTS and 32 GB RAM with the R2019A version of Matlab.

5.1.1 Datasets. The first dataset corresponds to spectrums of 680 dimensions, representing a protein rate measured on 10 different products: rapeseed (CLZ), corn gluten (CNG), sun flower seed (SFG), grass silage (EHH), full fat soya (FFS), wheat (FRG), sun flower seed (SFG), animal feed (ANF), soyameal set(representsr and whey (MPW). The complete dataset represents 4075 time series of 680 values (680 elements per time series).

The second real world dataset corresponds to time series of 500 dimensions which have been measured by attaching accelerometer at the neck of some sheep. Acelerometers captured 3-axial acceleration at a constant rate of 100Hz. The complete dataset represents 8532 time series of 500 values.

We have also done experiments on several real world datasets from the UCR Time Series Classification Archive [11], such as CinCECGTorso, EOGVerticalSigna, EOGHorizontalSignal, ArrowHead, etc.

5.2 Execution time

The first experiment on execution time is performed by keeping the time series length (n) fixed, and varying the subsequence length (m; plotted along $X-$axis). For this experiment, we used the protein and sheep datasets. For the protein dataset, we have used the first 100 time series and concatenated them to generate a single time series of 68000 (100×680) elements. In the case of the sheep dataset, we took the first 100 time series and concatenated them to generate a single time series of 50000 (100×500) elements).

The execution times of the six algorithms are plotted in Fig. 3a and 3b using the protein and sheep datasets respectively. As seen, the execution time of all algorithms decreases with increasing subsequence length (m). On both databases, *AAMP* and *ACAMP-Optimized* outperform other algorithms. Until $m = 8000$, *ACAMP* is better than *STOMP*, but for higher values *STOMP* behaves better. For very high values of m (*e.g.*, when m is close to n), the execution time of all algorithms gets almost the same, because in these cases there are few subsequences in the time series. Notice that in practice the subsequence size is not very high (*e.g.*, less than 4000), and in these cases the performance of *AAMP* and *ACAMP-Optimized* is significantly better than the state-of-the-art algorithms.

The second experiment is performed by keeping a fixed subsequence length $m = 256$ (in accordance with the experiments in related work, *e.g.* [1] and [2]), and varying the length of time series, *i.e.*, n. The results for the two datasets are shown in Fig.3c and 3d. We observe that the execution time of all algorithms increases linearly with the increase of time series length. *AAMP* and *ACAMP-Optimized* algorithms outperform the state-of-the-art algorithms, and their performance difference increases significantly by increasing n. Thus, the bigger is the time series, the higher is the performance gain of our *AAMP* and *ACAMP-Optimized* algorithms.

5.3 Discord discovery

The *AAMP* and *ACAMP* algorithms are capable to detect the discords (anomalies) from the time series like other matrix profile based algorithms such as *STOMP*, *SCRIMP* and *SCRIMP++*. The matrix profile generated by *ACAMP* is exactly the same as the one generated by *STOMP*, *SCRIMP* and *SCRIMP++*, as all of these techniques use the *z-normalized* Euclidean distance. But, *AAMP* uses non-normalized Euclidean distance, thus the detected discords can be different. Hence, depending on the user requirements and the domains of applications, the techniques from both groups can be useful. An example is shown in Fig. 5a by using two real ECG datasets [2]. The visible discords (of subsequence length 50) are marked by red color in these time series. It can be seen that the anomaly or unusual pattern existing in the first time series can be detected by *AAMP*, whereas *SCRIMP++* (or any of the other z-normalized based algorithm) was unable to detect it. The reason is due to *z-normalization* by *SCRIMP++*. *AAMP* is able to take into account the range of values of the matches with respect to the range of values of the given subsequence. This is why *AAMP* does not find a close match for this unusual subsequence (it's range of values is mostly less than -2). In the second time series (top right image in Fig. 5a) another similar situation is presented where *AAMP* was able to correctly detect the discord but *SCRIMP++* failed to locate it.

Figure 4 shows examples of time series from different UCR datasets, and the matrix profiles generated by AAMP and STOMP algorithms for the time series. In each time series there is a visible anomaly (an unusual pattern), which is clearly detected by the AAMP algorithm, *i.e.*, as high value point in the matrix profile. But, in the matrix profile generated by the STOMP algorithm, the anomalies are not visible or hardly distinguishable from other subsequences.

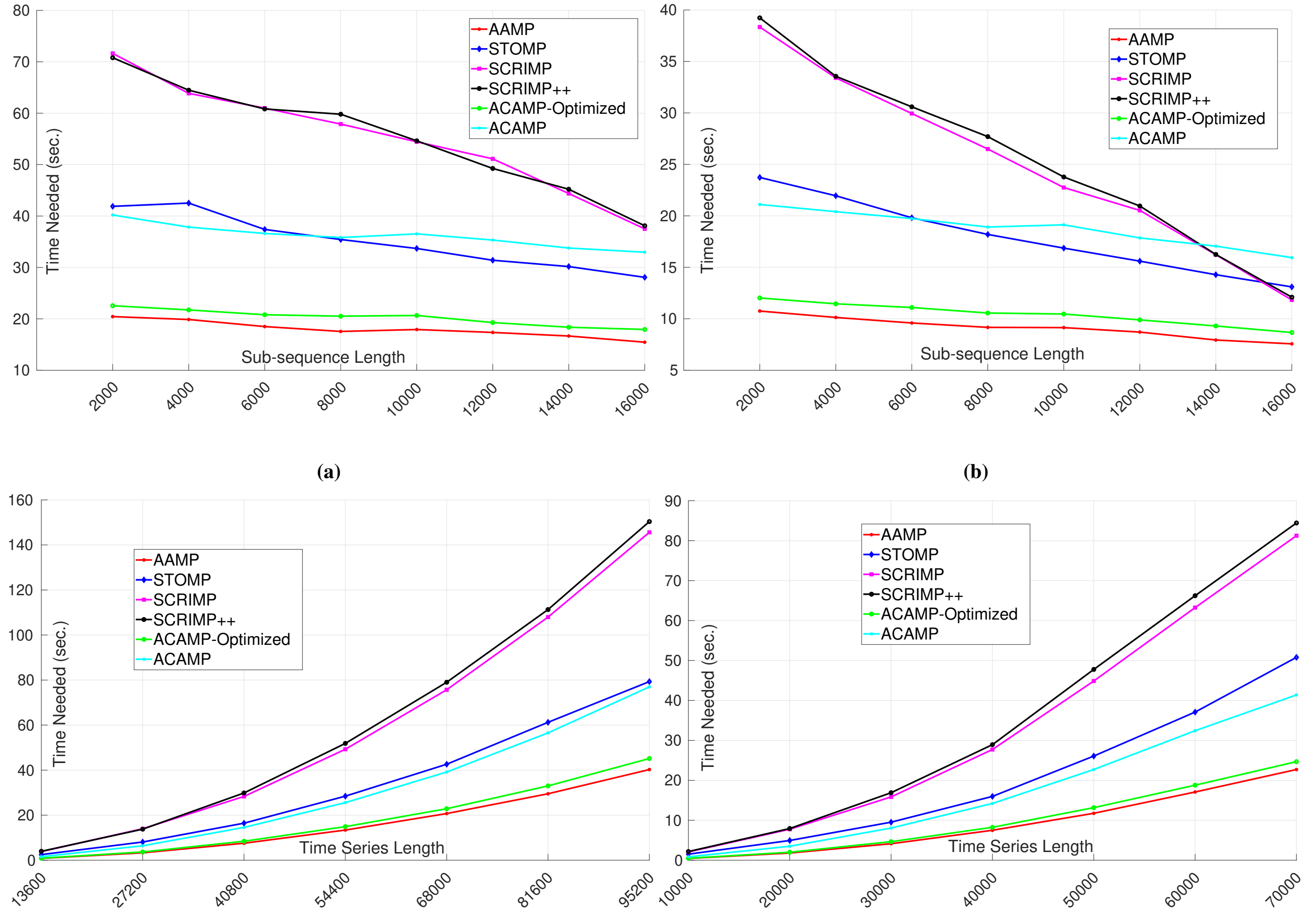
From the above mentioned experimental evaluations, we can conclude that our proposed *AAMP* algorithm shows better performance in detecting anomalies (and also motifs) in certain domain of applications, compared to the *z-normalization* based algorithms such as

[1] Our code and data are accessible at: https://sites.google.com/view/aamp-and-acamp/home

[2] https://sites.google.com/view/mstamp/

[3] https://www.cs.ucr.edu/~eamonn/MatrixProfile.html

[4] https://sites.google.com/site/scrimpplusplus/



(a)

(b)

(c)

(d)

Figure 3: The execution times of six algorithms with increasing the subsequence length (m): a) Execution time of the six algorithms on a time series of length 68000 (protein dataset). b) Execution time of the six algorithms on a time series of length 50000 (sheep dataset). The execution time of six algorithms are plotted with the increase of time series length (n): c) Execution time of the six algorithms on variable time series length (protein dataset) with $m = 256$. d) execution time of the six algorithms on variable time series length (sheep dataset) with $m = 256$.

STOMP and *SCRIMP++*. However, in certain domain of applications, *z-normalization* based algorithms are more useful. Then in such cases, it is better to use the *ACAMP-optimized* algorithm which has lower execution time than the state-of-the-art techniques, i.e. *STOMP*, *SCRIMP* and *SCRIMP++*, and is able to compute exactly the same matrix profile as the one computed by these algorithms.

5.4 Pros and Cons of Z-normalized over Non-normalized distance

There are pros & cons of both the *z-normalized* and non-normalized Euclidean distances. In this section, we discuss them.

5.4.1 Range of the matches. The techniques such as *STOMP*, *SCRIMP*, *SCRIMP++* and *ACAMP* are able to find the matches without taking into account the range of values of the matches. These techniques only consider the shape of the subsequences (because of z-normalization), whereas a non-normalized Euclidean distance based technique, e.g., *AAMP*, can find the matches from the same range of values as the given subsequence while taking into account its shape as well. Some examples of the matches obtained by *STOMP* and *AAMP* are shown in Fig. 1b. Hence, the z-normalization based techniques are capable of finding similar shape matches from any range of values, and can sometimes provide better matches than non-normalized techniques (see an example in Fig. 6). But when the range of values of the matches is important, then a technique such as *AAMP* is more useful.

5.4.2 Zero standard deviation. It is a quite bothersome problem that the z-normalized distance of two subsequences returns *infinity* when the standard deviation of one of the subsequences is zero

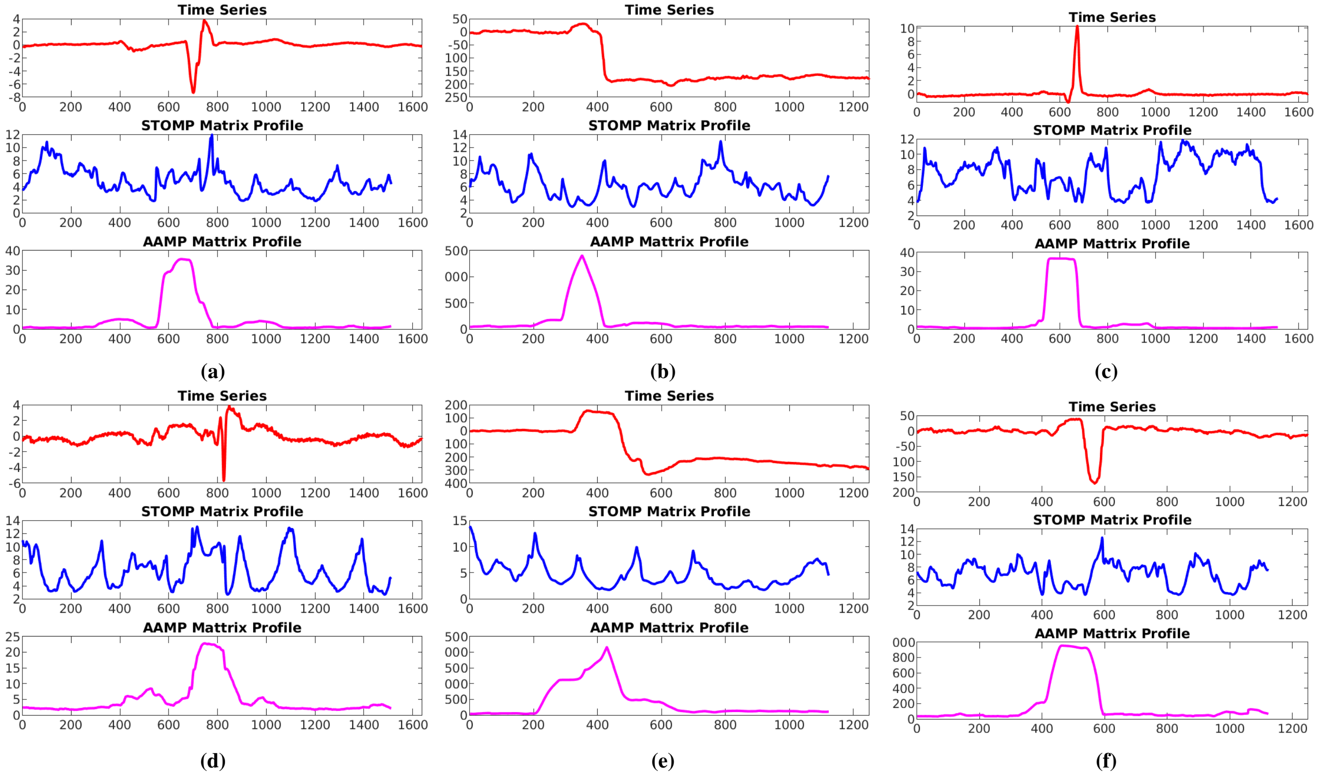


Figure 4: (a) Top: Time series from CinCECGTorso dataset. The discord is visible in it. Middle: the matrix profile, obtained by STOMP algorithm; Bottom: The matrix profile, obtained by AAMP algorithm. (b) Time series from EOGVerticalSignal dataset and corresponding matrix profile by STOMP and AAMP algorithms. (c) Time series from CinCECGTorso dataset (d) Time series from CinCECGTorso dataset (e) Time series from EOGHorizontalSignal dataset (f) Time series from EOGVerticalSignal dataset

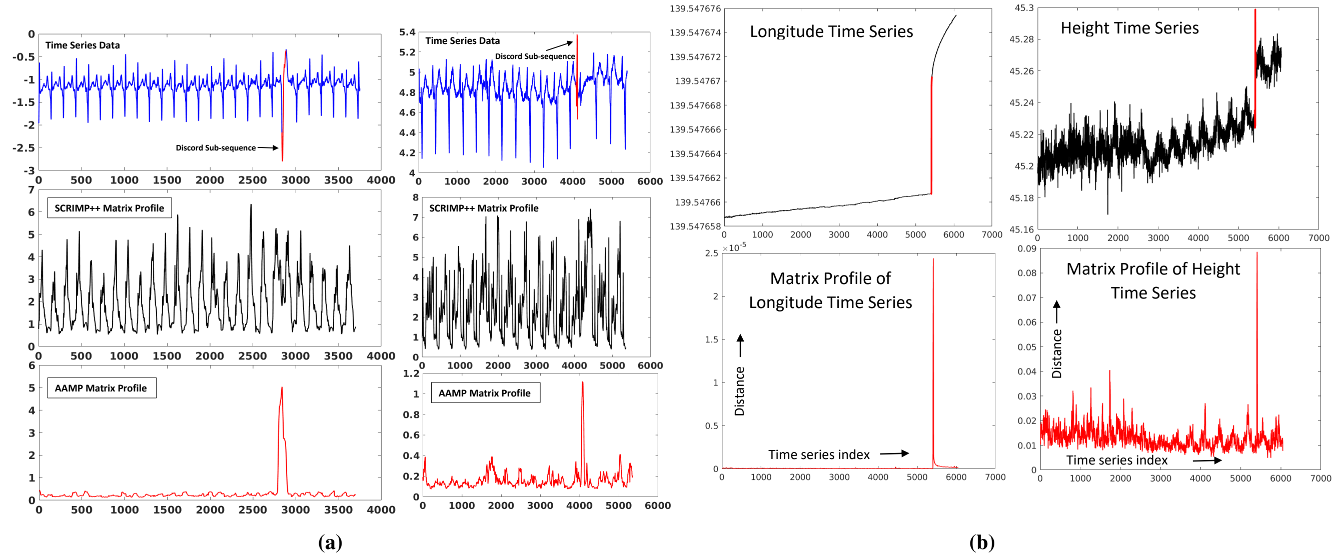


Figure 5: (a) Top: two time series from real ECG dataset. The visible discords in these time series are marked by red color. Middle: the matrix profile, obtained by SCRIMP++ algorithm; Bottom: The matrix profile, obtained by AAMP algorithm. (b) Top: the longitude and height time series of Seismic dataset (outliers are marked by red color); Bottom: the matrix profile obtained by AAMP algorithm.

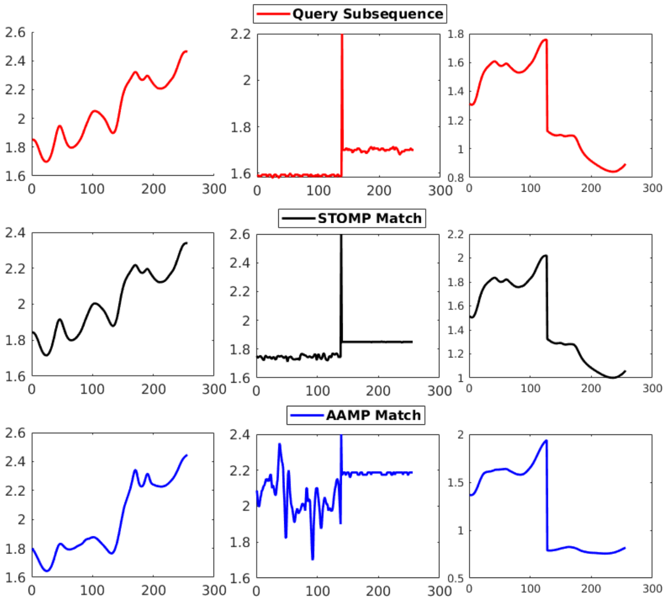


Figure 6: Top: First two query sub-sequences from protein and the third sub-sequence from sheep dataset. Middle: Better nearest neighbors, obtained by STOMP. Bottom: The nearest neighbors, obtained by AAMP algorithm.

(because of division by zero). This can happen when the signal of a subsequence remains stable (i.e., all the values are same in the subsequence). This kind of situation is quite frequent in real datasets, *e.g.*, during the periods when there is no noticeable activities. This problem does not exist for AAMP algorithm (based on the non-normalized Euclidean distance), because no division is done in its distance formula. An example is shown in Fig. 5b by using a real seismic dataset where the values of longitudes and heights are plotted. It can be visible that there are several places where the signals remain stable, hence the standard deviation of the subsequences (e.g. of size 50) would become zero. In these cases, we see that *AAMP* is able to detect the outliers by generating the matrix profile (see bottom images of Fig. 5b). But, the z-normalized based techniques can not find these anomalies.

6 RELATED WORK

Matrix profile has been recently proposed as an efficient technique for detecting motifs and discords in time series [7, 12]. In [1], Yeh et al. introduced the theoretical foundations of matrix profile and proposed a first algorithm, called STAMP for computing the matrix profile over a time series. It uses a similarity search algorithm, called MASS [1] that computes z-normalized Euclidean distance between two subsequences by using the Fast Fourier Transform (FFT). In [2], Zhu et al. proposed an algorithm, called STOMP, that is faster than STAMP. The STOMP algorithm is similar to STAMP but uses highly optimized nested loop algorithm by applying repeated calculation of distance profiles in the inner loop. However, while STAMP must evaluate the distance profiles in random order (to allow its anytime behavior), STOMP performs an ordered search. STOMP exploits the locality of these searches, and reduces the time complexity by a factor of $O(logn)$. In [8], the authors proposed an extension of STOMP, called SCRIMP++ (also an anytime algorithm), that usually converges faster than STOMP for large subsequence lengths. In [13], Zimmerman et al. proposed an extension of the GPU-based version of STOMP algorithm [2] by exploiting several novel insights for motif discovery envelope, using a scalable framework which can be deployed in commercial cloud based GPU clusters. To the best of our knowledge, almost all matrix profile algorithms have been developed for z-normalized Euclidean distance. In this paper, we proposed AAMP for the non-normalized Euclidean distance. We also proposed two algorithms for the z-normalized case, *i.e.*, ACAMP and ACAMP-Optimized, that are significantly faster than the state of the art algorithms working based on the z-normalized distance. The ACAMP and ACAMP-Optimized algorithms are designed based on an efficient incremental technique that does not need FFT calculations.

7 CONCLUSION

In this paper, we addressed the problem of matrix profile computation for a general class of Euclidean distances. We first proposed an efficient algorithm called AAMP for computing matrix profile for the non-normalized Euclidean distance. Then, we extended our algorithm for the p-norm distance, which is a general form of Euclidean. Then, we proposed ACAMP and its optimized version ACAMP-Optimized that use the same principle as AAMP, but for the case of z-normalized Euclidean distance. Our algorithms are exact, anytime, incrementally maintainable, and can be implemented easily using different languages. To evaluate the performance of our algorithms, we implemented them, and compared their performance with the baseline algorithms such as STOMP, SCRIMP, SCRIMP++. The results show the efficiency of AAMP and ACAMP-Optimized algorithms for computing matrix profile based on z-normalized and non-normalized Euclidean distances. They also illustrate the utility of the matrix profile generated by the AAMP algorithm for detecting anomalies in some daatsets, for which the state-of-the-art algorithms are not useful. Overall, we can conclude that both z-normalized and non-normalized based matrix profiles are required for knowledge extraction in a wide range of applications. In this paper, we proposed efficient techniques for both of them.

ACKNOWLEDGMENT

REFERENCES

[1] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. J. Keogh, "Matrix Profile {I:} All Pairs Similarity Joins for Time Series: {A} Unifying View That Includes Motifs, Discords and Shapelets," in *Proceedings of the International Conference on Data Mining (ICDM)*, pp. 1317–1322, 2016.

[2] Y. Zhu, Z. Zimmerman, N. S. Senobari, C.-C. M. Yeh, G. Funning, A. Mueen, P. Brisk, and E. J. Keogh, "Matrix Profile {II:} Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins," in *Proceedings of the International Conference on Data Mining (ICDM)*, pp. 739–748, 2016.

[3] Y. Zhu, A. Mueen, and E. J. Keogh, "Matrix profile IX: admissible time series motif discovery with missing data," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 6, pp. 2616–2626, 2021.

[4] M. Imamura, T. Nakamura, and E. J. Keogh, "Matrix profile XXI: A geometric approach to time series chains improves robustness," in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (R. Gupta, Y. Liu, J. Tang, and B. A. Prakash, eds.), pp. 1114–1122, ACM, 2020.

[5] C.-C. M. Yeh, H. V. Herle, and E. J. Keogh, "Matrix Profile {III:} The Matrix Profile Allows Visualization of Salient Subsequences in Massive Time Series," in

Proceedings of the International Conference on Data Mining (ICDM), pp. 579–588, 2016.

[6] H. A. Dau and E. J. Keogh, “Matrix Profile {V:} {A} Generic Technique to Incorporate Domain Knowledge into Motif Discovery,” in *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 125–134, 2017.

[7] Y. Zhu, C. M. Yeh, Z. Zimmerman, and E. J. Keogh, “Matrix profile XVII: indexing the matrix profile to allow arbitrary range queries,” in *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*, pp. 1846–1849, IEEE, 2020.

[8] Y. Zhu, C.-C. M. Yeh, Z. Zimmerman, K. Kamgar, and E. Keogh, “Matrix Profile XI: SCRIMP++: Time Series Motif Discovery at Interactive Speeds,” in *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 837–846, IEEE, nov 2018.

[9] A. Mueen, Y. Zhu, M. Yeh, K. Kamgar, K. Viswanathan, C. Gupta, and E. Keogh, “The fastest similarity search algorithm for time series subsequences under euclidean distance,” August 2017. http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html.

[10] “Website of SCRIMP++ .”

[11] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML, “The ucr time series classification archive,” October 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

[12] Z. Zimmerman, N. S. Senobari, G. J. Funning, E. E. Papalexakis, S. Oymak, P. Brisk, and E. J. Keogh, “Matrix profile XVIII: time series mining in the face of fast moving streams using a learned approximate matrix profile,” in *IEEE International Conference on Data Mining (ICDM)*, pp. 936–945, IEEE, 2019.

[13] Z. Zimmerman, K. Kamgar, Y. Zhu, N. S. Senobari, B. Crites, and G. Funning, “Scaling Time Series Motif Discovery with GPUs : Breaking the Quintillion Pairwise Comparisons a Day Barrier,”

SUPPLEMENTARY MATERIALS

S.1 Incremental Computation of Z-Normalized Euclidean Distance - Proof

Here, we present the proof of Lemma 3 and Equation 12 that gives an incremental formula for computing matrix profile by using z-normalized Euclidean distance.

Proof. Let μ_i and μ_j be the mean of the values in the sequences $T_{i,m}$ and $T_{j,m}$ respectively. Also, let σ_i and σ_j be the standard deviation of the values in the subsequences $T_{i,m}$ and $T_{j,m}$ respectively. Then, the z-normalized Euclidean distance between the subsequences $T_{i,m}$ and $T_{j,m}$ is defined as:

$$DZ_{i,j} = \sqrt{\sum_{l=1}^{m-1} \left(\frac{t_{i+l} - \mu_i}{\sigma_i} - \frac{t_{j+l} - \mu_j}{\sigma_j} \right)^2} \tag{16}$$

where

$$\mu_i = \frac{1}{m} \sum_{l=0}^{m-1} t_{i+l};\ \mu_j = \frac{1}{m} \sum_{l=0}^{m-1} t_{j+l} \tag{17}$$

and

$$\sigma_i = \sqrt{\frac{1}{m} \sum_{l=0}^{m-1} t_{i+l}^2 - (\mu_i)^2};\ \sigma_j = \sqrt{\frac{1}{m} \sum_{k=0}^{m-1} t_{j+l}^2 - (\mu_j)^2}. \tag{18}$$

We can write the square of DZ as following:

$$\begin{aligned}
DZ_{i,j}^2 &= \textstyle\sum_{l=0}^{m-1} \left(\frac{t_{i+l}-\mu_i}{\sigma_i} - \frac{t_{j+l}-\mu_j}{\sigma_j} \right)^2 \\
&= \textstyle\sum_{l=0}^{m-1} \left(\left(\frac{t_{i+l}-\mu_i}{\sigma_i} \right)^2 - 2 \left(\frac{t_{i+l}-\mu_i}{\sigma_i} \right) \left(\frac{t_{j+l}-\mu_j}{\sigma_j} \right) + \left(\frac{t_{j+l}-\mu_j}{\sigma_j} \right)^2 \right) \\
&= \textstyle\sum_{l=0}^{m-1} \left(\frac{t_{i+l}^2-2t_{i+l}\mu_i+(\mu_i)^2}{(\sigma_i)^2} - 2(\frac{t_{i+l}t_{j+l}-\mu_i t_{j+l}-t_{i+l}\mu_j+\mu_j\mu_i}{\sigma_i\sigma_j} \right) + \\
&\quad \frac{t_{j+l}^2-2t_{j+l}\mu_j+(\mu_j)^2}{(\sigma_j)^2})
\end{aligned} \tag{19}$$

Let

$$\begin{aligned}
&A_i = \textstyle\sum_{l=0}^{m-1} t_{i+l};\ B_j = \sum_{l=0}^{m-1} t_{j+l};\ \mathbf{A}_i = \sum_{l=0}^{m-1} t_{i+l}^2; \\
&\mathbf{B}_j = \textstyle\sum_{l=0}^{m-1} t_{j+l}^2;\ \mathbf{C}_{\mathbf{i,j}} = \sum_{l=0}^{m-1} t_{i+l}t_{j+l}.
\end{aligned} \tag{20}$$

Then, we have:

$$\mu_i = \tfrac{1}{m}A_i, \qquad \mu_j = \tfrac{1}{m}B_j \tag{21}$$

$$(\sigma_i)^2 = \tfrac{1}{m}\mathbf{A}_i - \tfrac{1}{m^2}A_i^2, \qquad (\sigma_j)^2 = \tfrac{1}{m}\mathbf{B}_j - \tfrac{1}{m^2}B_j^2. \tag{22}$$

Then, the z-normalized Euclidean distance can be written as:

$$\begin{aligned}
DZ_{i,j}^2 &= \textstyle\sum_{l=0}^{m-1} (\frac{t_{i+l}^2-2t_{i+l}\mu_i+(\mu_i)^2}{(\sigma_i)^2} \\
&-2 \left(\frac{t_{i+l}b_{j+l}-\mu_i t_{j+l}-t_{i+l}\mu_j+\mu_j\mu_i}{\sigma_i\sigma_j} \right) + \frac{t_{j+l}^2-2t_{j+l}\mu_j+(\mu_j)^2}{(\sigma_j)^2}) \\
&= \frac{\mathbf{A}_i - 2A_i^2\frac{1}{m} + \frac{A_i^2}{m}}{\frac{1}{m}\mathbf{A}_i - \frac{1}{m^2}A_i^2} - 2 \times \frac{\mathbf{C}_{\mathbf{i,j}} - \frac{2}{m}A_iB_j + \frac{A_iB_j}{m}}{\sqrt{(\frac{1}{m}\mathbf{A}_i - \frac{1}{m^2}A_i^2)(\frac{1}{m}\mathbf{B}_j - \frac{1}{m^2}B_j^2)}} + \\
&\quad \frac{\mathbf{B}_j - 2B_j^2\frac{1}{m} + \frac{B_j^2}{m}}{\frac{1}{m}\mathbf{B}_j - \frac{1}{m^2}B_j^2} \\
&= 2m - 2 \times \frac{m^2\mathbf{C}_{\mathbf{i,j}} - mA_iB_j}{\sqrt{(m\mathbf{A}_i - A_i^2)(m\mathbf{B}_j - B_j^2)}} \\
&= 2m \left(1 - \frac{\mathbf{C}_{\mathbf{i,j}} - \frac{1}{m}A_iB_j}{\sqrt{(\mathbf{A}_i - \frac{1}{m}A_i^2)(\mathbf{B}_j - \frac{1}{m}B_j^2)}} \right).
\end{aligned} \tag{23}$$

As mentioned in Subsection 4.4.1, by taking

$$F_{i,j} = \frac{(A_iB_j - m\mathbf{C}_{\mathbf{i,j}}) \times |A_iB_j - m\mathbf{C}_{\mathbf{i,j}}|}{(\mathbf{A}_i - \frac{1}{m}A_i^2)(\mathbf{B}_j - \frac{1}{m}B_j)}, \tag{24}$$

we have $DZ_{i,j} = 2m + 2\text{sign}(F_{i,j}) \times \sqrt{|F_{i,j}|}$ and we can use the following equivalence in our algorithm:

$$DZ_{i,j} > DZ_{i,k} \Leftrightarrow F_{i,j} > F_{i,k}.$$

S.2 Shapelet discovery

Here we explain how shapelets can be discovered by matrix profile, and then show examples of shapelets discovered by z-normalized and non-normalized matrix profile algorithms from real datasets.

Consider two time series A and B, having class 1 and 0 as their corresponding class labels. We compute the matrix profiles of A an B, denoted by P_A and P_B, and also their joint matrix profiles P_{AB} and P_{BA} (see the definition of *joint matrix profile* in Section 2). The shapelets can be discovered by calculating the difference in heights of P_{AB} v/s P_A (or P_{BA} v/s P_B) which is then used as the indicator of good shapelet candidates. The idea here is that if a discriminating pattern is present in A and not in B, then it is highly probable that we will see a "bump" at the location of this pattern in P_{AB} (the same is true for P_{BA} also). Hence, when an element-wise difference (denoted by $\mathcal{U} = |P_A - P_{AB}|$) is calculated between P_A and P_{AB} vectors, we will find high values at those locations where such discriminating patterns (or subsequences) exist in A (same is true for B, if we look into P_B and P_{BA}).

Using time series from the ArrowHead dataset of UCR Archive, in Fig. 7 (b) and (d) we show the curve of P_A and P_{AB} along with the difference between P_A and P_{AB} plotted in Fig. 7 (c) and (e) for the STOMP and AAMP algorithms respectively. A significant difference (quantified by a threshold, shown in dashed line) is observed between the heights of P_A and P_{AB} curves, which intrinsically locates the occurrence of good candidate shapelets patterns (detected by STOMP and AAMP algorithms). These difference curves can serve to locate the patterns that only occur in one of the two time series (*i.e.*, good candidates for shapelets). This experiment is performed by randomly choosing 10 time series and concatenating them. The execution times required by AAMP to compute P_A and P_{AB} are 0.05 and 0.17 seconds respectively.

Using time series from the ArrowHead dataset of UCR Archive, in Fig. 7 (b) and (d) we show the curve of P_{AA} and P_{AB} along with the difference between P_{AA} and P_{AB} plotted in Fig. 7 (c) and (e) for the STOMP and AAMP algorithms respectively. A significant difference (quantified by a threshold, shown in dashed line) is observed between the heights of P_{AA} and P_{AB} curves, which intrinsically locates the occurrence of good candidate shapelets patterns (detected by STOMP and AAMP algorithms). These difference curves can serve to locate the patterns that only occur in one of the two time series (*i.e.*, good candidates for shapelets). This experiment is performed by randomly choosing 10 time series and concatenating them. The execution times required by AAMP to compute P_{AA} and P_{AB} are 0.05 and 0.17 seconds respectively.

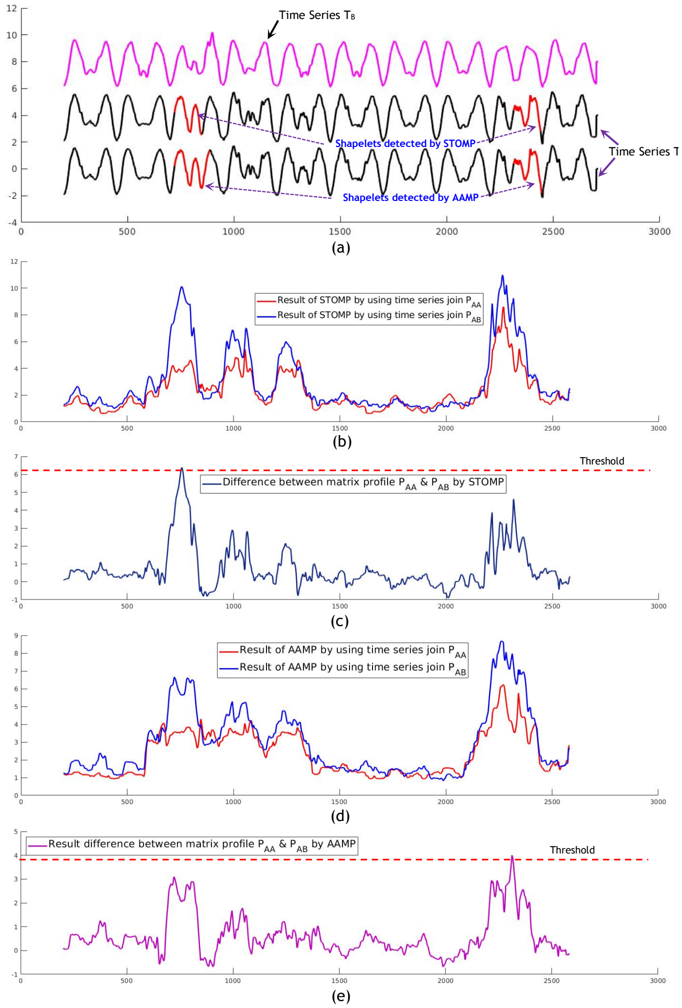


Figure 7: The time series shapelet discovery: (a) Two time series T_A and T_B formed by concatenating individual time series of class 1 and 0 respectively of the ArrowHead dataset. (b) (d) The matrix profile P_{AA} and P_{AB} by STOMP and AAMP algorithms respectively. (c) (e) The difference between P_{AB} and P_{AA}, by STOMP and AAMP algorithms respectively.

S.3 Better performance of Z-Normalized distance over non-normalized distance

In the following Fig.8, 9, 10, 11, we have shown some interesting examples where the *z-normalized* distance has performed better than *non normalized* distance based matrix profile. The images in Fig.8, shows that *z-normalized* distance is able to find more possible locations of outliers by creating sharper peaks of matrix profile curve, compared to AAMP based matrix profile.

Whereas, from examples shown in Fig.9, we can visualize that *z-normalized* based matrix profiles (by STOMP algorithm) are able to show better and relevant possible outliers by detecting multiple and sharper peaks (marked by red circles), compared to AAMP based matrix profile. The detection of multiple possible outliers location by *z-normalized* based matrix profile would help the data analyst and domain experts to manually validate it's legitimacy as they will have more options of possible outliers.

In Fig.10, 11 also, we show several matrix profile plots where *z-normalized* based matrix profile is able to find different and extra location of possible outliers (compared to *non-normalized* based matrix profile). Some time these detected outliers by *z-normalized* based matrix profile are relevant and some times they are irrelevant. But, it will always give a handful of extra and different possible outliers locations for the domain experts.

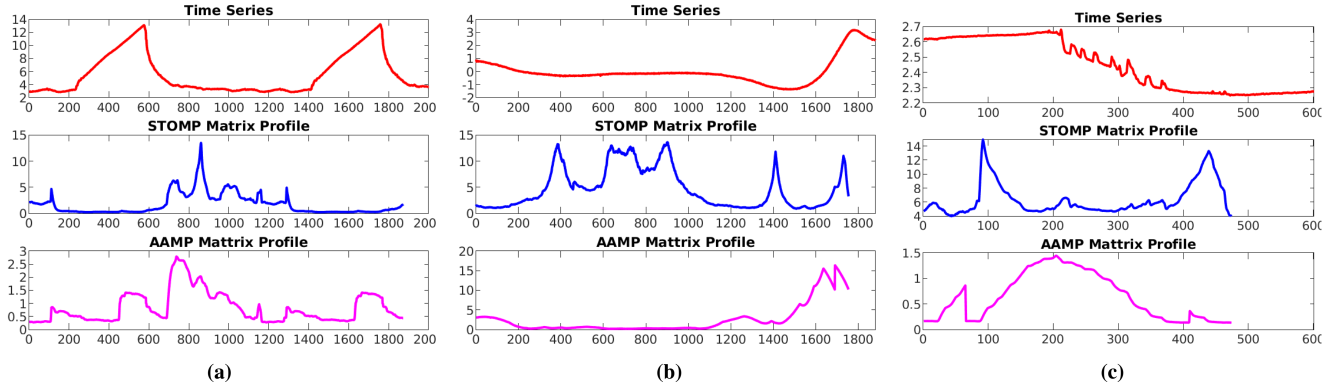


Figure 8: (a, b, c) Top: Time series from PigAirwayPressure, InlineSkate, InsectEPGSmallTrain dataset respectively. Middle: the matrix profile, obtained by STOMP algorithm; Bottom: The matrix profile, obtained by AAMP algorithm.

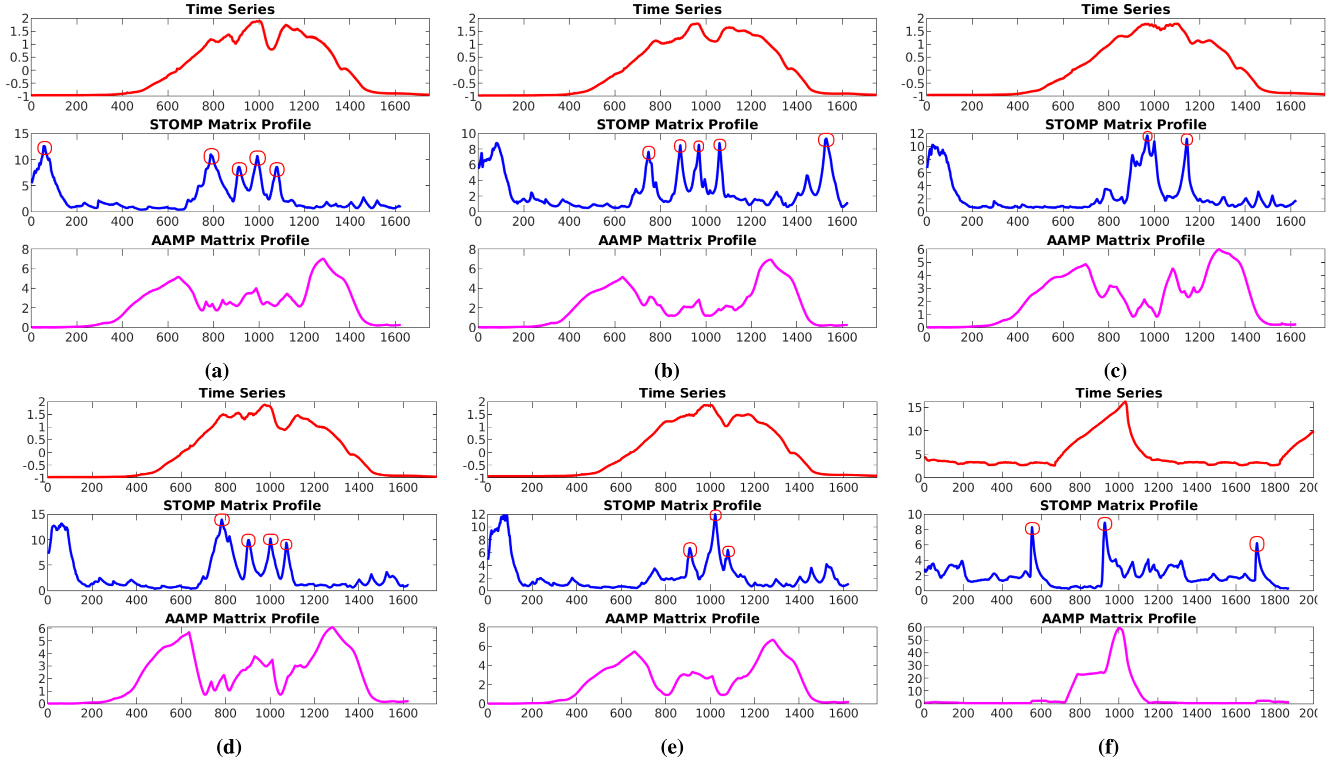


Figure 9: (a, b, c, d, e) Top: Various time series from EthanolLevel dataset. Middle: the matrix profile, obtained by STOMP algorithm; Bottom: The matrix profile, obtained by AAMP algorithm. (f) Time series from PigAirwayPressure dataset and corresponding matrix profile by STOMP and AAMP. These matrix profile plots shows that in several cases, the z-normalized based (STOMP) algorithm is able to find more clear (sharper peaks, marked by red circles) detection of outliers than AAMP.

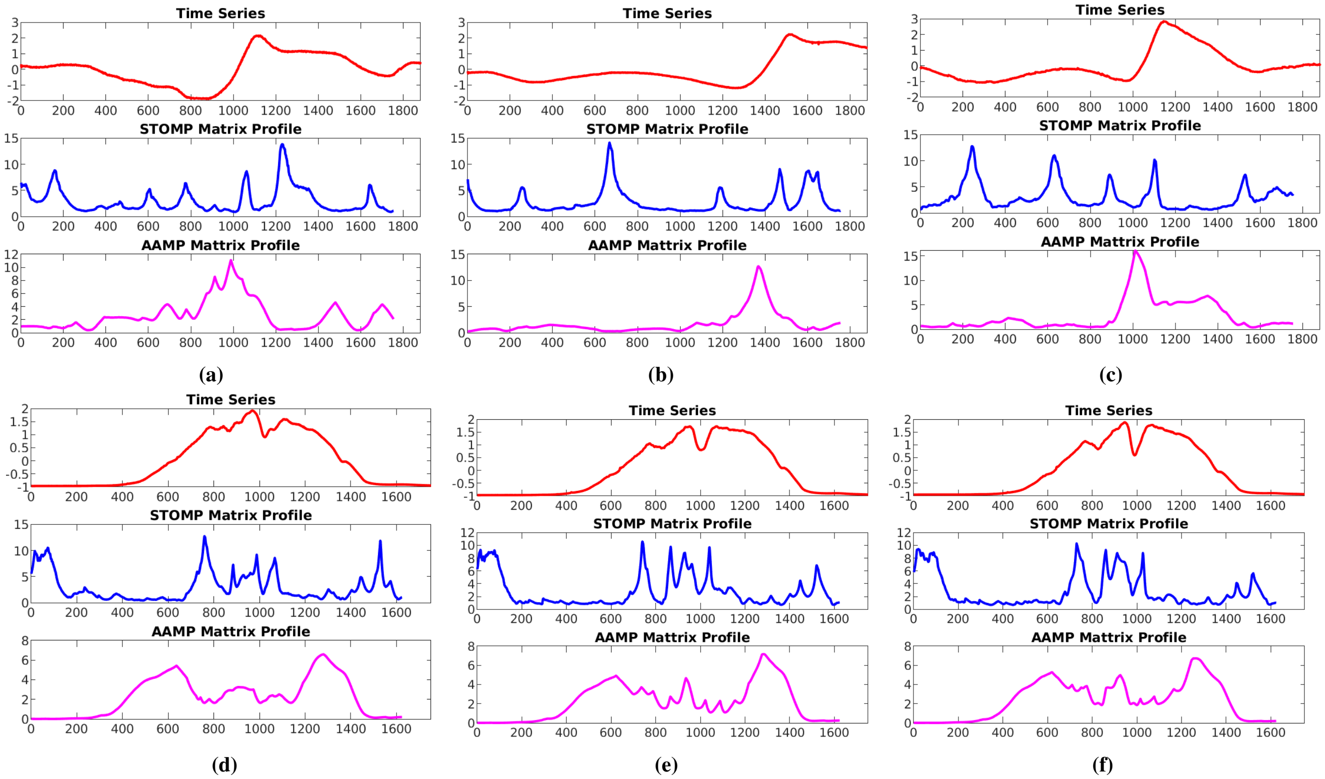
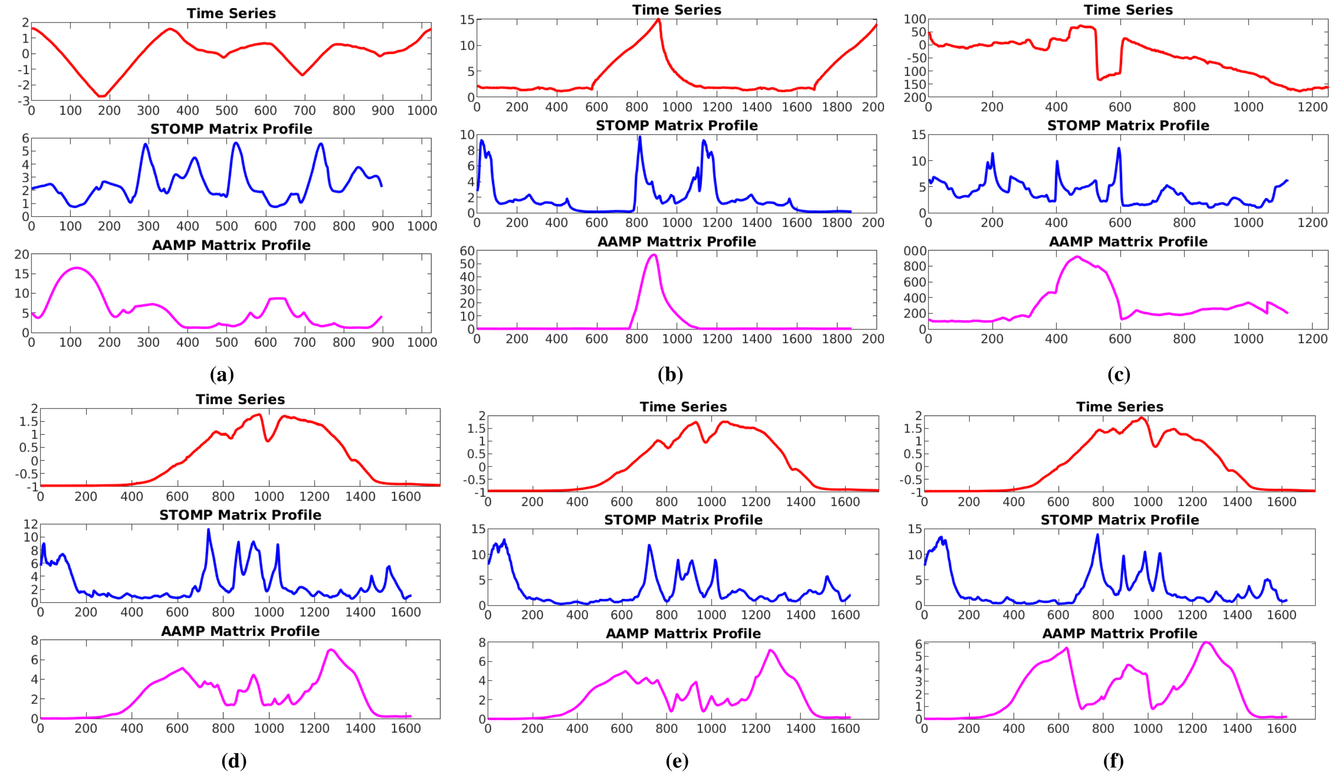


Figure 10: (a, b, c) Top: Various time series from InlineSkate dataset. Middle: the matrix profile, obtained by STOMP algorithm; Bottom: The matrix profile, obtained by AAMP algorithm. (f) Time series from EthanolLevel dataset and corresponding matrix profile by STOMP and AAMP.



(a) (b) (c)

(d) (e) (f)

Figure 11: (a, b, c) Top: Time series from MixedShapesRegularTrain, PigAirwayPressure, EOGVerticalSignal dataset. The discord is visible in it. Middle: the matrix profile, obtained by STOMP algorithm; Bottom: The matrix profile, obtained by AAMP algorithm. (d, e, f) Time series from EthanolLevel dataset and corresponding matrix profile by STOMP and AAMP algorithm respectively.