MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Algorithms for Fast Computation of Pan Matrix Profiles of Time Series Under Unnormalized Euclidean Distances

Zhang, Jing; Nikovski, Daniel N.

TR2022-041 May 19, 2022

Abstract

We propose an approximation algorithm called LINKUMP to compute the Pan Matrix Profile (PMP) under the unnormalized distance (useful for value-based similarity search) using double-ended queue and linear interpolation. The algorithm has comparable time/space complexities as the stateof-the-art algorithm for typical PMP computation under the normalized ‘2 distance (useful for shape-based similarity search). We validate its efficiency and effectiveness through extensive numerical experiments and a real-world anomaly detection application.

International Conference on Applied Statistics and Data Analytics 2022

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Algorithms for Fast Computation of Pan Matrix Profiles of Time Series Under Unnormalized Euclidean Distances

Jing Zhang and Daniel Nikovski

Mitsubishi Electric Research Labs
{jingzhang, nikovski}@merl.com

***Abstract*—We propose an approximation algorithm called LINKUMP to compute the Pan Matrix Profile (PMP) under the unnormalized ℓ_∞ distance (useful for value-based similarity search) using double-ended queue and linear interpolation. The algorithm has comparable time/space complexities as the state-of-the-art algorithm for typical PMP computation under the normalized ℓ_2 distance (useful for shape-based similarity search). We validate its efficiency and effectiveness through extensive numerical experiments and a real-world anomaly detection application.**

***Index Terms*—Pan Matrix Profile, unnormalized Euclidean distance, double-ended queue, discord discovery, anomaly detection**

I. Introduction

In recent years, the time series Matrix Profile (MP) has been proposed as a versatile data structure for many data mining tasks (time series discord discovery, segmentation, shapelets/motifs/chains discovery etc.) [1]. As a companion time series, the MP records distances between nearest neighbors of subsequences in the original time series. Under the normalized ℓ_2 distance metric (leading to shape-based similarity search), several fast algorithms have been devised to compute the MP based on the Fast Fourier Transform (FFT) [1] or Dynamic Programming (DP) [2]. Under the unnormalized ℓ_p $(1 \leq p \leq \infty)$ distance metric (leading to value-based similarity search), fast algorithms based on DP [3] or selected special data structures (double-ended queue and segment tree) [4] are also proposed. Note that, assuming the length of a given time series is n, then the time complexity of the state-of-the-art algorithm [2] for computing its MP is $O(n^2)$, which is independent of the subsequence length.

As pointed out in [4], the MP under a specific distance metric is appropriate for certain tasks, but not necessarily appropriate for others; its actual performance highly depends on data and applications. For example, though shown to be powerful in many tasks, the MP under normalized ℓ_2 distance cannot deal with "values of subsequences"-based discord discovery (anomaly detection), and the normalized ℓ_2 distance is even not defined for constant subsequences.

Although the MP is already a convenient enough tool for many time series data mining tasks, it still requires the practitioner to set one critical parameter – the subsequence length (i.e., the query size) for similarity search. To remove the only parameter, [5] proposes a new data structure – the Pan Matrix Profile (PMP), which is a matrix with each row being a Matrix Profile corresponding to a single subsequence length. Essentially, the PMP is a parameter-free version of the MP that has potential to deal with various time series data mining tasks. However, computing an exact PMP is slow; the time complexity is $O(|M|n^2)$, where M is the list of all considered subsequence lengths and $|M|$ is the total number of subsequence lengths that M contains. To speed up the computation for the PMP, [5] proposes an approximation algorithm called SKIMP, which tries to optimize the order of candidate subsequence lengths to compute their respective MP's. Note, however, that this approximation algorithm does not deal with the MP's for the remaining subsequence lengths when it terminates computing the MP's for a selected set of subsequence lengths. As noted by the authors of [5], they use the normalized ℓ_2 distance for the MP/PMP computation, and this leads to the fact that "given a matrix profile P_i, it is impossible to predict or even produce an upper or lower bound for matrix profile P_{i+1}."

Another recent work [6] also tries to remove the only parameter (i.e., the subsequence length for similarity search), but for a more specific purpose – discord discovery (anomaly detection). The authors propose a parameter-free algorithm named MERLIN, for discovery of arbitrary length discords (anomalies) in massive time series archives. Note that the MERLIN algorithm is the state-of-the-art in terms of speed, but it only tries to find discords as fast as possible; to that end, the algorithm avoids keeping track of the anomaly scores of the vast majority of all the subsequences. On the other hand, when applied for discord discovery, the MP/PMP algorithms output much more information, including anomaly scores of non-discords, and sometimes such information is intuitive and important for analysts.

Because algorithms for the MP/PMP under unnormalized distances would still produce good (or even better) results in various time series data mining tasks on many data sets, in this paper we investigate alternative approximation algorithms to speed up the PMP computation. Our algorithm utilizes the monotonicity of MP's with respect to subsequence lengths to "LINK" (through Linear Interpolation) unfinished MP's to already finished MP's, and thus is essentially different than the SKIMP algorithm. We call our algorithm LINKUMP, where "LI" also stands for Linear Interpolation, and "UMP" is short for Matrix Profile under Unnormalized distances. As a building block, the computation for the MP corresponding to a single subsequence length is done using the algorithms (based on double-ended queue or segment tree) developed in [4].

Recently there has been an explosion of papers on deep learning based anomaly detection [7], [8], [9], [10], [11], [12], [13], but "it is not obvious that deep learning outperforms simpler and more direct" MP based methods [6]. In addition, most of the proposed algorithms require setting multiple parameters, thus drastically challenging the practitioners and easily encountering overfitting issues.

As is shown in [4], the Matrix Profile (MP) under unnormalized distances could be applied to anomaly detection tasks; sometimes it would even outperform the MP under normalized distances, depending on the data sets. A key observation is that, no matter under what distance metrics, the MP values are essentially "anomaly scores" of the subsequences in the time series, and the maximum MP value means that the corresponding subsequence has the largest anomaly score; we call this subsequence the "discord." As pointed in [6], "since their introduction, time series discords have emerged as a competitive approach for discovering anomalies."

Admittedly, the MP/PMP under unnormalized distances also has potential to be successfully applied to other time series data mining tasks (e.g., motif discovery [5]), we focus on value-based discord discovery (anomaly detection) in this paper.

The rest of the paper is organized as follows. We define the MP/PMP (under unnormalized distances) of time series and prove the monotonicity of the PMP in Section II. In Section III, we elaborate the LINKUMP algorithm to compute the PMP. Numerical results are shown in Section IV. We conclude in Section V. The Appendix (Section VI) contains details of some subroutines required by the LINKUMP algorithm.

Notation: Let $X = [x_0, x_1, \ldots, x_{n-1}]$ be a real-valued time series with length n, where $x_t \in \mathbb{R}$ is the value sampled at time instance t, $t = 0, 1, \ldots, n-1$. Denote by m the length of a subsequence (window size), which satisfies $1 \le m \le n-1$. Let $X_{j,\ldots,j+m-1} = [x_j, x_{j+1}, \ldots, x_{j+m-1}]$ denote the jth subsequence of X, $0 \le j \le n-m$. Denote by $|A|$ the length of a given list (or vector) A. Assuming M is a list (vector) and $m \in M$, we let $m \| M$ denote the index of m in M. Given a lower bound L, an upper bound U ($-\infty < L < U < +\infty$), and a step size S ($0 < S < U - L$), we use range(L, U, S) to denote an ordered list containing all the elements that are on the interval $[L, U)$ and can be expressed as $L + i \times S$ where i is an integer.

II. Definition and Monotonicity

We first review the definition of the Matrix Profile under unnormalized distances [4], and then extend it to the Pan Matrix Profile.

Definition 1 (Matrix Profile of Time Series)
The Matrix Profile of X is a new time series $Y = [y_0, y_1, \ldots, y_{n-m}]$, where

$$y_j = \min_{0 \le j' \le n-m, j' \ne j} d(X_{j,\ldots,j+m-1}, X_{j',\ldots,j'+m-1}), \quad (1)$$

where $d(\cdot,\cdot)$ is the ℓ_p ($1 \le p \le \infty$) distance.

To put it another way, the Matrix Profile (MP) of X at time instance j is the ℓ_p distance between the jth subsequence and its nearest-neighbor subsequence in X. Similar to [4], to make descriptions concise while still capturing the essence of the MP, in Definition 1 we do not include the other component of the MP (i.e., the respective indices of the nearest-neighbor subsequences) from the original definition in [1].

Definition 2 (Pan Matrix Profile of Time Series)
Given a list of subsequence lengths $M = [m_0, m_1, \ldots, m_{|M|-1}]$, the Pan Matrix Profile of X is an $|M| \times n$ matrix P with each row filled by a Matrix Profile of X, which corresponds to a specific subsequence length; in particular,

$$P_{i,j} = \min_{0 \le j' \le n-m_i, j' \ne j} d(X_{j,\ldots,j+m_i-1}, X_{j',\ldots,j'+m_i-1}), \quad (2)$$

$\forall 0 \le i \le |M|-1, 0 \le j \le n-m_i$, where m_i is the subsequence length corresponding to the i-th row.

Note that, in Definition 2, the unfilled entries of P (i.e., $P_{i,j}, \forall 0 \le i \le |M|-1, n-m_i < j \le n$) could always be set as a Not-a-Number (NaN) value. We are now in a position to prove the monotonicity of the PMP with respect to the subsequence lengths:

Theorem II.1 (Monotonicity of Pan Matrix Profile)
Assume for the list of subsequence lengths $M = [m_0, m_1, \ldots, m_{|M|-1}]$ we have $m_{i_1} < m_{i_2}, \forall 0 \le i_1 < i_2 \le |M|-1$. Then $P_{i,j}$'s defined by (2) satisfy $P_{i_1,j} \le P_{i_2,j}, \forall 0 \le i_1 < i_2 \le |M|-1, 0 \le j \le n-m_{i_2}$.

Proof: Arbitrarily fix $j \in [0, n-m_{i_2}]$. Noting $m_{i_1} < m_{i_2}$, from (2) we have

$$\begin{aligned} P_{i_1,j} &= \min_{0 \le j' \le n-m_{i_1}, j' \ne j} d(X_{j,\ldots,j+m_{i_1}-1}, X_{j',\ldots,j'+m_{i_1}-1}) \\ &\overset{(a)}{\le} \min_{0 \le j' \le n-m_{i_2}, j' \ne j} d(X_{j,\ldots,j+m_{i_1}-1}, X_{j',\ldots,j'+m_{i_1}-1}) \\ &\overset{(b)}{\le} \min_{0 \le j' \le n-m_{i_2}, j' \ne j} d(X_{j,\ldots,j+m_{i_2}-1}, X_{j',\ldots,j'+m_{i_2}-1}) \\ &= P_{i_2,j}. \end{aligned}$$

The inequality (a) above holds due to the fact that the minimization on the right hand side of (a) is taken over a subset of distances of the set (of distances) for the left hand side. To see why the inequality (b) also holds, we further arbitrarily fix $j' \in [0, n-m_{i_2}], j' \ne j$. Recalling that $d(\cdot,\cdot)$ is the ℓ_p distance, we obtain:

(i) if $1 \le p < \infty$, then

$$\begin{aligned} & d(X_{j,\ldots,j+m_{i_1}-1}, X_{j',\ldots,j'+m_{i_1}-1}) \\ = \; & \big(\textstyle\sum_{k=0}^{m_{i_1}-1} |x_{j+k} - x_{j'+k}|^p\big)^{1/p} \\ \le \; & \big(\textstyle\sum_{k=0}^{m_{i_2}-1} |x_{j+k} - x_{j'+k}|^p\big)^{1/p} \\ = \; & d(X_{j,\ldots,j+m_{i_2}-1}, X_{j',\ldots,j'+m_{i_2}-1}); \end{aligned}$$

(ii) if $p = \infty$, then

$$d(X_{j,\ldots,j+m_{i_1}-1}, X_{j',\ldots,j'+m_{i_1}-1})$$

$$
\begin{aligned}
&= \max_{0\le k\le m_{i_1}-1} |x_{j+k} - x_{j'+k}| \\
&\le \max_{0\le k\le m_{i_2}-1} |x_{j+k} - x_{j'+k}| \\
&= d(X_{j,\ldots,j+m_{i_2}-1}, X_{j',\ldots,j'+m_{i_2}-1}).
\end{aligned}
$$

□

It is worth pointing out that Theorem II.1 would not hold for the MP/PMP under normalized distances. To see this more clearly, let us look at an example using a synthesized time series shown in Fig. 1. In Fig. 2a we plot its exact MP values under the normalized ℓ_2 distance at the 50th time instance, versus the subsequence lengths. It is seen that the MP values are not consistently increasing with respect to the subsequence lengths (there is an obvious decrease around the subsequence length 50).

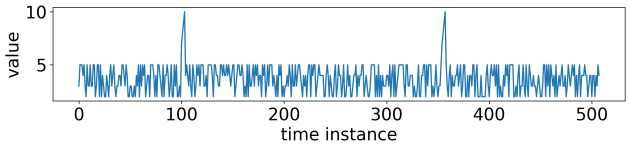
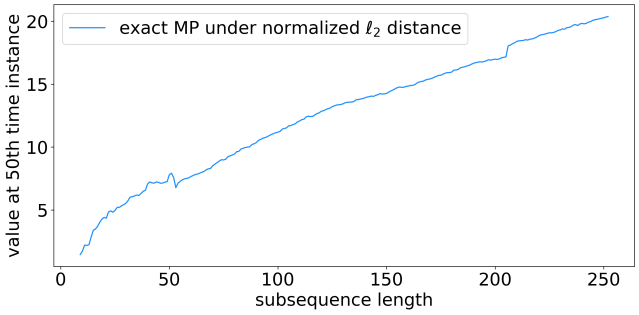
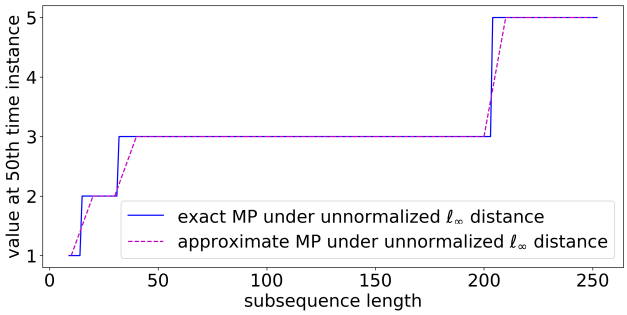


Fig. 1: A synthesized time series.



(a)



(b)

Fig. 2: MP values vs. subsequence lengths of the synthesized time series shown in Fig. 1.

III. Algorithm

In this section, we describe the LINKUMP algorithm to compute the PMP. Without loss of generality, we only use the ℓ_∞ distance (i.e., $d(X_{j,\ldots,j+m-1}, X_{j',\ldots,j'+m-1}) = \max_{0\le k\le m-1} |x_{j+k} - x_{j'+k}|$) to illustrate our algorithms.

First, we give a brief review of our previously proposed algorithm for computing the MP corresponding to a single subsequence length; we refer the reader to [4] for more details. This algorithm (see Section VI-C) applies the double-ended queue data structure, and requires two subroutines: (i) finding sliding maxima (see Section VI-A) and (ii) computing element-wise distance to rotated time series (see Section VI-B). Its time complexity is $O(n^2)$ and space complexity is $O(n)$.

Now we describe the LINKUMP algorithm. It is an approximation algorithm for PMP computation by utilizing linear interpolation. The idea is essentially different than the SKIMP algorithm [5], which tried to order the selected subsequence lengths and compute only a portion of them from the beginning. On the other hand, our approach is as follows. Once we have computed the MPs for a set of selected subsequence lengths, we do linear interpolation for each and every missing subsequence length; e.g., if we have done computing the MPs for the following 4 subsequence lengths, $[10, 20, 30, 40]$, then we do linear interpolation for the corresponding intervals $(10, 20)$, $(20, 30)$, and $(30, 40)$, respectively, to obtain approximate MPs for subsequence lengths in these intervals. Fig. 2b visualizes this procedure for the synthesized time series example (shown in Fig. 1), where both the exact and approximate MP functions (with respect to subsequence lengths) are piecewise linear and nondecreasing.

To formalize the algorithm, we denote by L (resp., U) the minimum (resp., maximum) subsequence length, S the step size, and $\beta \in (0, 1]$ the completion rate for the PMP computation of the selected set of subsequence lengths. We summarize the steps as Algorithm 1, where Lines 10 through 17 describe how the linear interpolation works for the in-between subsequence lengths. It is seen that the time complexity of this algorithm is $O(\beta|M|n^2)$, and the space complexity is $O(|M|n)$, where $|M|$ is the number of candidate subsequence lengths.

Algorithm 1 Pan Matrix Profile under Unnormalized distance by Linear Interpolation (LINKUMP)

1: **procedure** LINKUMP(X, L, U, S, β)
2: $n \leftarrow$ length of X
3: $M \leftarrow \text{range}\big(L, U+1, S\big)$
4: $P \leftarrow |M| \times n$ matrix of NaN's
5: $M' \leftarrow \text{range}\big(L, U+1, \left\lfloor \frac{1}{\beta} \right\rfloor \cdot S\big)$
6: **for** m in M' **do**
7: $i \leftarrow m \| M$
8: $P_i \leftarrow$ MatrixProfileDeque(X, m)
9: **end for**
10: **for** l in range$(0, |M'|-1, 1)$ **do**
11: **for** m in range$\big(m'_l + S, m'_{l+1}, S\big)$ **do**
12: $i \leftarrow m \| M$
13: **for** j in range$(0, n-m+1, 1)$ **do**
14: $P_{i,j} \leftarrow (*)$[12]

[1] $(*) = P_{m'_l \| M, j} + \frac{m - m'_l}{m'_{l+1} - m'_l}(P_{m'_{l+1} \| M, j} - P_{m'_l \| M, j})$

[2] If $P_{m'_l \| M, j}$ is NaN, we simply take $(*)$ as NaN; otherwise, if $P_{m'_{l+1} \| M, j}$ is NaN, we simply take $(*)$ as the value of $P_{m'_l \| M, j}$.

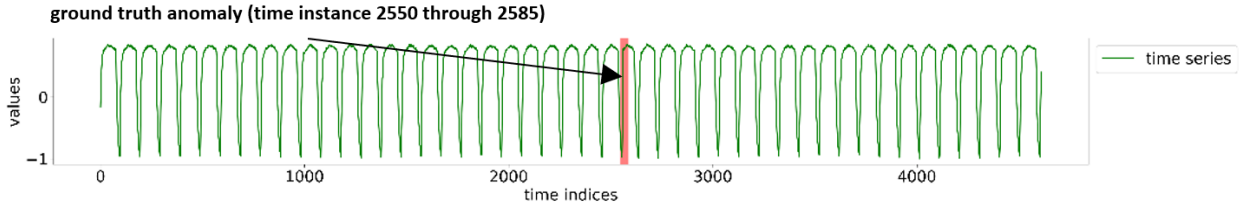


Fig. 3: A signal from the Mars Curiosity rover T1; annotated as having an anomaly on time instances 2550 through 2585 (an ultra-subtle anomaly example from [6]).

```
            end for
        end for
    end for
    return P
end procedure
```

IV. Numerical Results

In this section, we apply the LINKUMP algorithm for discord discovery (anomaly detection) purposes. We note that, in terms of the time complexity (speed), the current state-of-the-art parameter-free discord discovery algorithm is MERLIN [6]. However, the MERLIN algorithm only tries to find discords with the highest speed by avoiding keeping track of the anomaly scores of the vast majority of all subsequences. On the other hand, when applied for discord discovery, our LINKUMP algorithm would output much more intuitive information (such as anomaly scores of non-discords) that could be important for analysts (e.g., when one knows anomaly scores for both discords and non-discords, she would get an impression of how "serious" the discords are deviating from the normal subsequences).

It happened that for a dataset with an ultra subtle anomaly used in the MERLIN paper, the MP with unnormalized ℓ_∞ distance would very easily find the correct discord, while the version with normalized ℓ_2 distance would be harder, due to its strong sensitivity with respect to subsequence lengths.

A. Application in Anomaly Detection

As noted by [6], anomaly detection benchmarks in the literature typically contain anomalies that also yield to casual visual inspection, and "it is interesting to ask if we can detect very subtle anomalies, that would defy human inspection." To that end, [6] proposes experiments to obtain ground-truth subtle anomalies; here we borrow one of the resulting data sets: the time series of the Mars Curiosity Rover T1.

In Fig. 3 we show a signal from the Mars Curiosity rover T1 that was annotated as having an anomaly on time instances 2550 through 2585 (an ultra-subtle anomaly example from [6]). Fig. 4 shows the Matrix Profiles (MPs) of the time series in Fig. 3 when subsequence length $m = 3$.

It is seen that using the MP with unnormalized ℓ_∞ distance one could easily find a discord at the correct time instance (by setting a threshold, say 0.2, for the uMP value; here "u" means the MP is under the "unnormalized" distance), while if using the MP with normalized ℓ_2 distance one could not find a discord at the correct time instance, because the MP values do not have a peak at the correct time interval of the ground-truth anomaly. In our experiments we also observe such results for subsequence lengths $m = 4, 5, 10, 20, 30$, and only when m gets larger, can both uMPs and MPs enable us to correctly find the discord. This illustrates that at least for this example compared to the anomaly scores based on the MP with normalized ℓ_2 distance, the ones based on the MP with unnormalized ℓ_∞ distance are much less sensitive to the subsequence length m.

It turns out that for this anomaly detection application, the PMP computed by our LINKUMP algorithm would outperform the one computed by the SKIMP alternative, because these two algorithms use unnormalized and normalized distances, respectively. In Fig. 5 we show the approximate PMP's for the time series in Fig. 3 computed by the LINKUMP and SKIMP algorithms, respectively, both with 5% MP's exactly computed. Thanks to the linear interpolation procedure, we obtain much more details in the PMP heatmap in Fig. 5a than the one in Fig. 5b, and it is seen that the PMP computed by our LINKUMP algorithm can clearly identify the discord at the correct time interval, while the one computed by the SKIMP alternative would only be able to vaguely identify the same discord, after MP's with larger subsequence lengths are computed. To see how good the approximations can be obtained, for the same Mars Curiosity rover T1 signal, we also show in Fig. 6 the heatmaps of the PMP's produced by the LINKUMP algorithm with $\beta = 1, 0.01, 0.02$.

B. CPU Time for Pan Matrix Profile Computation

To compute the PMP, similar to the SKIMP algorithm, our LINKUMP algorithm also has an "anytime" property; namely, at any time, the algorithm could be terminated, resulting an approximation of the exact PMP. Depending on how large the completion rate β is, the execution time of LINKUMP could be various, but in our experiments we could set β as small values (say 0.01, 0.05), while still achieving overall good performance. For example, for a time series X with length 5000, with $L = 3, U = 500, S = 1, \beta = 0.05$, the LINKUMP algorithm implemented in Python can finish running within 10 minutes on a desktop computer with an Intel(R) Core(TM) i7-4770 CPU and 16 GB of system memory.

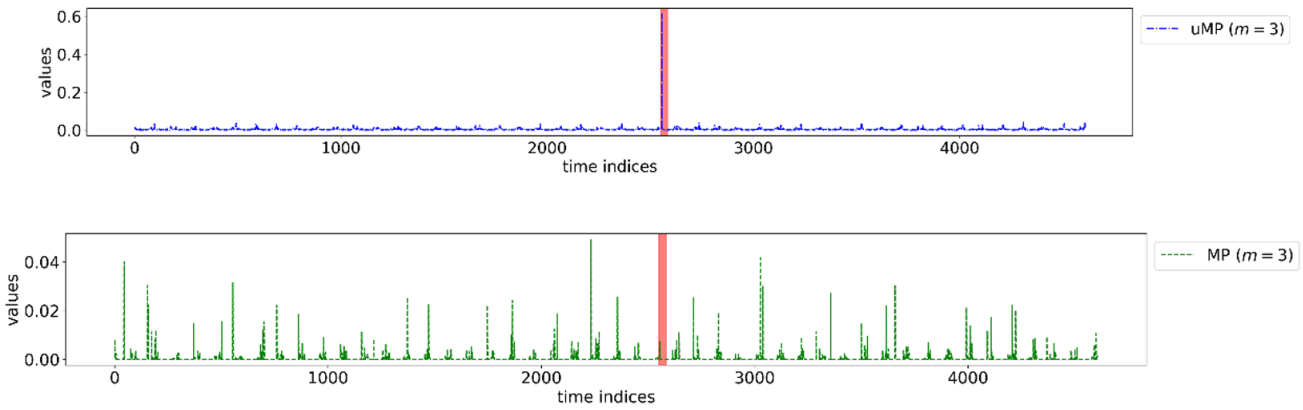


Fig. 4: Matrix Profiles of the time series in Fig. 3 when subsequence length $m = 3$; top: MP with unnormalized ℓ_∞ distance (could easily find a discord at the correct time instance), bottom: MP with normalized ℓ_2 distance (could not find a discord at the correct time instance).

V. Conclusion

In this paper, by utilizing the monotonicity of the Matrix Profiles (with respect to subsequence lengths) and linear interpolation, we propose an approximation algorithm called LINKUMP to compute the Pan Matrix Profile (PMP) under the unnormalized ℓ_∞ distance. We show that the algorithm has comparable time/space complexities as the state-of-the-art algorithm for typical PMP computation under the normalized ℓ_2 distance. The efficiency and effectiveness of the LINKUMP algorithm has been validated through extensive numerical experiments and a real-world anomaly detection application. Our future work includes extending the linear interpolation to general polynomial interpolation and considering additional applications.

VI. Appendix

A. Finding Sliding Maxima Using Double-ended Queue

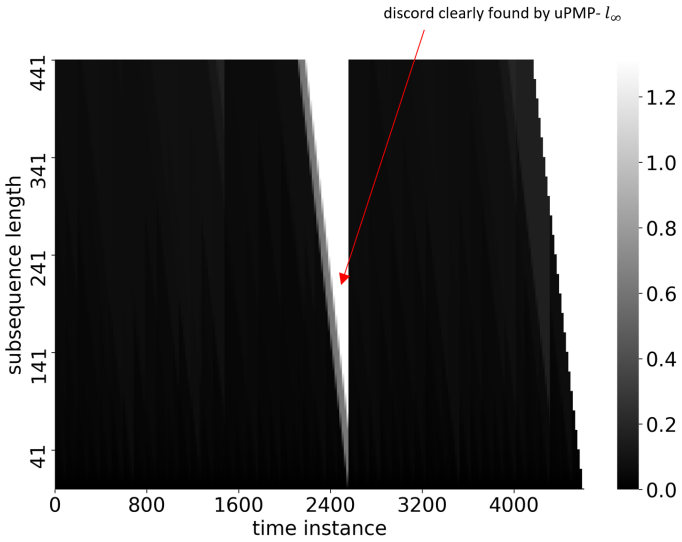
procedure SlidMaxDeque(Z, m)
  $n \leftarrow$ length of Z
  if m equals 1 **then**
    return Z
  end if
  initialize Q as a new empty double-ended queue
  initialize $slidMax$ as a new empty vector
  for $l = 0, 1, \ldots, n-1$ **do**
    if Q is nonempty and $Q[0] \leq l - m$ **then**
      remove the leftmost element from Q
    end if
    while $Q \neq \emptyset$ and $Z[l] > Z[Q[-1]]$ **do**
      remove the rightmost element from Q
    end while
    append l to the rightmost of Q
    if $l \geq m - 1$ **then**
      append $Z[Q[0]]$ to the rightmost of $slidMax$
    end if
  end for
  return $slidMax$
end procedure

B. Computation of Element-wise Distance to Rotated Time Series

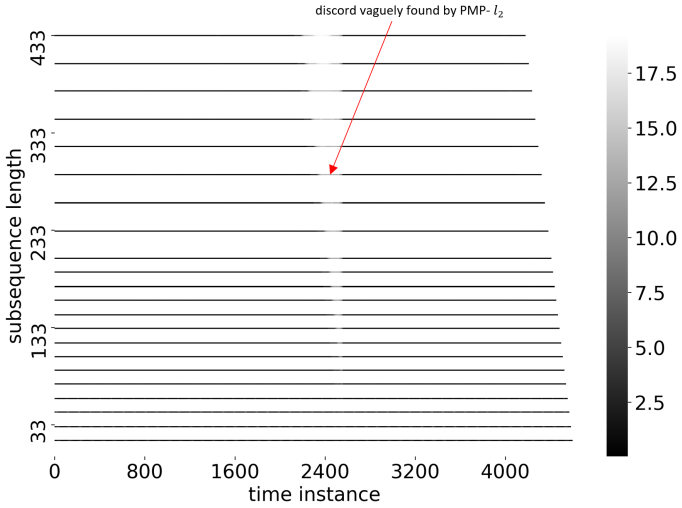
procedure EleWiseDistToRotTimeSeries(X, i)
  $n \leftarrow$ length of X
  $eleDist \leftarrow$ the i-th rotated X (i.e., $X^{(i)} = [x_{i+1}, x_{i+2}, \ldots, x_{n-1}, x_0, x_1, \ldots, x_i]$)
  for $l = 0, 1, \ldots, n-1$ **do**
    $eleDist[l] \leftarrow |eleDist[l] - X[l]|$
  end for
  return $eleDist$
end procedure

C. Computation of Matrix Profile Under Unnormalized ℓ_∞ Distance Using Double-ended Queue

procedure MatrixProfileDeque(X, m)
  $n \leftarrow$ length of X
  initialize MP as an array with length $n-m+1$ and each element being a large enough float number (e.g., 1.0×10^6)
  for $i = 0, 1, \ldots, n-2$ **do**
    $eleDist \leftarrow$ EleWiseDistToRotTimeSeries(X, i)
    $slidMax \leftarrow$ SlidMaxDeque($eleDist, m$)
    for $j = 0, 1, \ldots, n-m$ **do**
      if $i \leq n-m-j-1$ or $i \geq n-j-1$ **then**
        $MP[j] \leftarrow \min(MP[j], slidMax[j])$
      end if
    end for
  end for
  return MP
end procedure

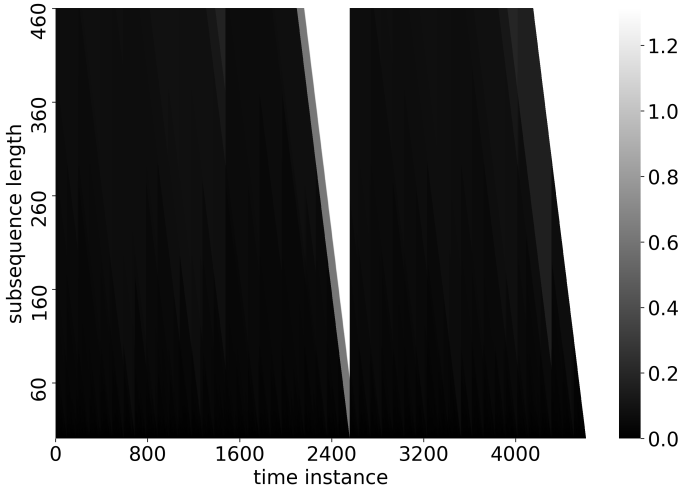


(a) Approximate PMP computed by LINKUMP.

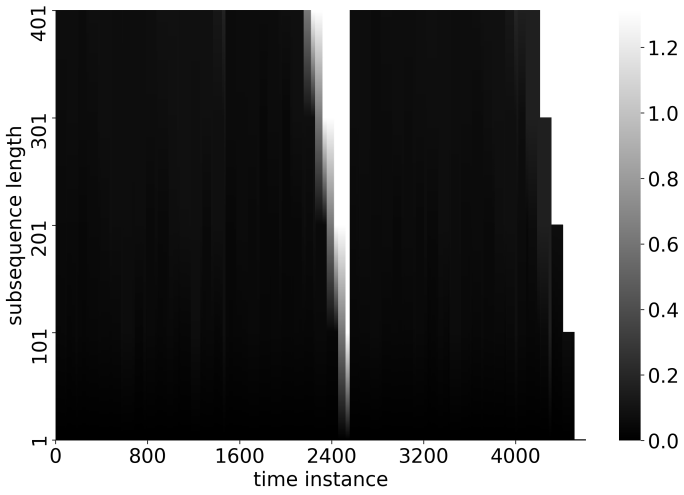


(b) Approximate PMP computed by SKIMP.

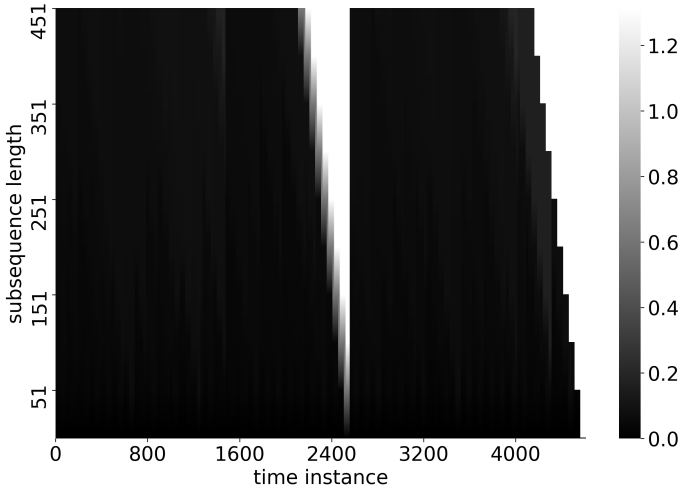
Fig. 5: Approximate PMP's for the time series in Fig. 3 computed by the LINKUMP and SKIMP algorithms, respectively; both with 5% MP's exactly computed.



(a) Exact PMP computed by LINKUMP with $\beta = 1$.



(b) Approximate PMP computed by LINKUMP with $\beta = 0.01$.



(c) Approximate PMP computed by LINKUMP with $\beta = 0.02$.

Fig. 6: PMP's for the time series in Fig. 3 computed by the LINKUMP algorithm.

REFERENCES

[1] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets," in *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 2016, pp. 1317–1322.

[2] Y. Zhu, Z. Zimmerman, N. S. Senobari, C.-C. M. Yeh, G. Funning, A. Mueen, P. Brisk, and E. Keogh, "Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins," in *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 2016, pp. 739–748.

[3] R. Akbarinia and B. Cloez, "Efficient matrix profile computation using different distance functions," *arXiv preprint arXiv:1901.05708*, 2019.

[4] J. Zhang and D. Nikovski, "Algorithms for fast computation of matrix profiles of time series under unnormalized euclidean distances," in *SIAM International Conference on Data Mining (SDM21)*. SIAM, 2021, submitted.

[5] F. Madrid, S. Imani, R. Mercer, Z. Zimmerman, N. Shakibay, and E. Keogh, "Matrix profile xx: Finding and visualizing time series motifs

of all lengths using the matrix profile," in *2019 IEEE International Conference on Big Knowledge (ICBK)*. IEEE, 2019, pp. 175–182.

[6] T. Nakamura, M. Imamura, R. Mercer, and E. Keogh, "Merlin: Parameter-free discovery of arbitrary length anomalies in massive time series archives," in *2020 IEEE 20th international conference on data mining (ICDM)*. IEEE, 2020.

[7] P. Filonov, A. Lavrentyev, and A. Vorontsov, "Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model," *arXiv preprint arXiv:1612.06676*, 2016.

[8] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1285–1298.

[9] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48 231–48 246, 2018.

[10] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "Deepant: A deep learning approach for unsupervised anomaly detection in time series," *IEEE Access*, vol. 7, pp. 1991–2005, 2018.

[11] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019.

[12] Y. Bao, Z. Tang, H. Li, and Y. Zhang, "Computer vision and deep learning–based data anomaly detection method for structural health monitoring," *Structural Health Monitoring*, vol. 18, no. 2, pp. 401–421, 2019.

[13] F. Ni, J. Zhang, and M. N. Noori, "Deep learning for data anomaly detection and data compression of a long-span suspension bridge," *Computer-Aided Civil and Infrastructure Engineering*, vol. 35, no. 7, pp. 685–700, 2020.