APUMPEDI: Approximating Pan Matrix Profiles of Time Series Under Unnormalized Euclidean Distances by Interpolation

Zhang, Jing; Nikovski, Daniel

TR2022-088 August 06, 2022

Abstract

The Matrix Profile (MP) of a time series has been proposed as a versatile primitive for many data mining tasks. As a companion time series, the MP records distances between nearest neighbors of sub- sequences in the original time series. The Pan Matrix Profile (PMP) is a matrix with each row being an MP corresponding to a single subsequence length and computing explicitly an exact PMP is slow. We propose an approximation algorithm called APUMPEDI to compute the PMP under the unnormalized Euclidean distance based on MP algorithms combined with interpolation. We validate their efficiency and effectiveness through extensive numerical experiments on both real-world and synthesized data sets.

International conference on Time Series and Forecasting (ITISE) 2022

APUMPEDI: Approximating Pan Matrix Profiles of Time Series Under Unnormalized Euclidean Distances by Interpolation

Jing Zhang and Daniel Nikovski

Mitsubishi Electric Research Labs,
201 Broadway, 8th Floor, Cambridge, MA 02139, USA
{jingzhang,nikovski}@merl.com
https://www.merl.com/

Abstract. The Matrix Profile (MP) of a time series has been proposed as a versatile primitive for many data mining tasks. As a companion time series, the MP records distances between nearest neighbors of subsequences in the original time series. The Pan Matrix Profile (PMP) is a matrix with each row being an MP corresponding to a single subsequence length, and computing explicitly an exact PMP is slow. We propose an approximation algorithm called APUMPEDI to compute the PMP under the unnormalized Euclidean distance based on MP algorithms combined with interpolation. We validate their efficiency and effectiveness through extensive numerical experiments on both real-world and synthesized data sets.

Keywords: Pan Matrix Profile, unnormalized Euclidean distance, approximation algorithm, interpolation

1 Introduction

The Matrix Profile (MP) of time series has been proposed as a versatile primitive for various data mining tasks [1]. By definition, the MP keeps track of distances between nearest neighbors of subsequences in a given time series. Algorithms based on the fast Fourier transform (FFT) or dynamic programming (DP) have been proposed to compute the MP under the normalized Euclidean (ℓ_2, in particular) distance metric (leading to shape-based similarity search) [1,2]. Algorithms based on DP [3] or some special data structure (such as a double-ended queue) [4] have also been devised to compute the MP under general unnormalized Euclidean (could be ℓ_p, $\forall 1 \leq p \leq \infty$) distance metric (leading to value-based similarity search). Assuming the length of a given time series is n, then the best time complexity of the aforementioned algorithms for computing its MP is $O(n^2)$, which is independent of the subsequence length.

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 for similarity search. To eliminate the need to guess this one 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 the 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. As noted in [5], the normalized ℓ_2 distance is used 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}." Consequently, the SKIMP algorithm can not deal with the MPs for the remaining subsequence lengths when it terminates computing the MP's for a selected set of subsequence lengths.

Another recent work, [6], also tries to remove the only parameter for the MP computation, 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 the purpose of 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 (see, e.g., [4,7]), in this paper we further investigate alternative approximation algorithms to speed up the PMP computation. Similar to the LINKUMP algorithm [7], our APUMPEDI (stands for "Approximating Pan Matrix Profile under Unnormalized Euclidean Distance by Interpolation") algorithm developed in this paper is based on the monotonicity of MPs with respect to subsequence lengths and various interpolation methods. This approximate algorithm can be viewed as a generalization of the LINKUMP algorithm, noting that the latter only uses linear interpolation. As a building block, the computation for the MP under unnormalized ℓ_p ($1 \leq p \leq \infty$) distance corresponding to a single subsequence length is done using the algorithms developed in [4] (for $p = \infty$; based on a double-ended queue) or [3] (for $1 \leq p < \infty$; based on DP).

We organize the remainder of the paper as follows. In Section 2, under unnormalized Euclidean distances, the definition of the MP/PMP of time series and the monotonicity of the PMP are reviewed. In Section 3, we elaborate the APUMPEDI algorithm. Results from numerical experiments are presented in Section 4. We give concluding remarks in Section 5.

Notation: Denote by $X = [x_0, x_1, \ldots, x_{n-1}]$ a real-valued time series (the length is n), with $x_t \in \mathbb{R}$ being the value sampled at time index $t,\ t = 0, 1, \ldots, n-1$. Let m be the length of a subsequence satisfying $1 \leq m \leq n-1$. Denote by $X_{j,\ldots,j+m-1} = [x_j, x_{j+1}, \ldots, x_{j+m-1}]$ the j-th subsequence of X, $0 \leq j \leq n-m$. Let $|A|$ denote the length of a given list (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 denote by range$(L, U+1, S)$ an ordered list (vector) consisting of all the elements that are within the interval $[L, U]$ and can be expressed as $L + i \times S$, where i is an integer.

2 Definition and Monotonicity

We first review the definitions of the Matrix Profile and the Pan Matrix Profile, under unnormalized Euclidean distances.

Definition 1 (Matrix Profile of Time Series [4])
The Matrix Profile (MP) of time series $X = [x_0, x_1, \ldots, x_{n-1}]$ is a new time series $Y = [y_0, y_1, \ldots, y_{n-m}]$, where

$$y_j = \min_{0 \leq j' \leq n-m, j' \neq j} d(X_{j,\ldots,j+m-1}, X_{j',\ldots,j'+m-1}), \tag{1}$$

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

In other words, at time index j, the value of the MP of X is the unnormalized Euclidean distance between the jth subsequence and its nearest-neighbor subsequence in X. To simplify the description, while still capturing the essence of the MP, in Def. 1 we exclude the other component of the MP (i.e., the respective indices of the nearest-neighbor subsequences) discussed in [1].

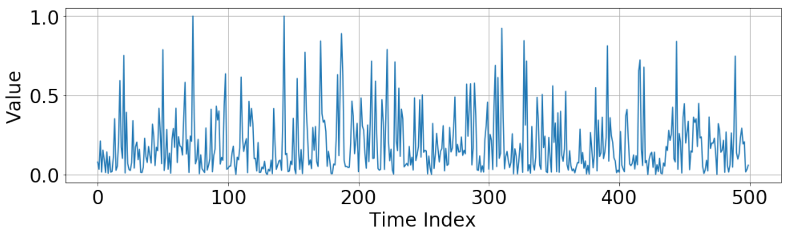
Definition 2 (Pan Matrix Profile of Time Series [7])
Given a list of subsequence lengths $M = [m_0, m_1, \ldots, m_{|M|-1}]$, the Pan Matrix Profile (PMP) of time series $X = [x_0, x_1, \ldots, x_{n-1}]$ is an $|M| \times n$ matrix P with each row filled by a Matrix Profile of X, which corresponds to a specific subsequence length; in particular,

$$P[i,j] = \min_{0 \leq j' \leq n-m_i, j' \neq j} d(X_{j,\ldots,j+m_i-1}, X_{j',\ldots,j'+m_i-1}), \tag{2}$$

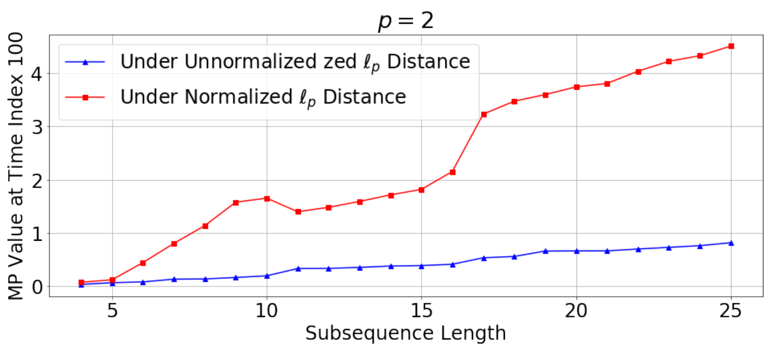
$\forall 0 \leq i \leq |M|-1, 0 \leq j \leq n-m_i$, where m_i is the subsequence length corresponding to the i-th row.

Note that, in Def. 2, we set the unfilled entries of P (i.e., $P[i,j], \forall 0 \leq i \leq |M|-1, n-m_i < j \leq n$) as a Not-a-Number (NaN) value.

Next, we review the monotonicity of the PMP with respect to the subsequence lengths.



(a)



(b)

Fig. 1: (a) A synthesized time series; (b) MP values at time index 100 vs. subsequence lengths of the synthesized time series.

Theorem 1 (Monotonicity of Pan Matrix Profile [7]) *Assume for the list of subsequence lengths* $M = [m_0, m_1, \ldots, m_{|M|-1}]$ *we have* $m_{i_1} < m_{i_2}, \forall 0 \leq i_1 < i_2 \leq |M| - 1$. *Then the PMP of time series* $X = [x_0, x_1, \ldots, x_{n-1}]$ *defined by* (2) *satisfy* $P[i_1, j] \leq P[i_2, j], \forall 0 \leq i_1 < i_2 \leq |M| - 1, 0 \leq j \leq n - m_{i_2}$.

As pointed out in [7], Theorem 1 would not hold for the PMP under normalized Euclidean distances. To see this more clearly, let us look at an example using a synthesized time series shown in Fig. 1a. In Fig. 1b we plot its MP values under both the normalized and unnormalized ℓ_2 distances at the 100-th time instance, versus the subsequence lengths. It is seen that, under normalized ℓ_2 distance, the MP values are not consistently increasing with respect to the subsequence lengths (there is an obvious decrease when the subsequence length increases from 10 to 11).

3 Approximation Algorithm for PMP Computation

3.1 Description of the APUMPEDI Algorithm

In this section, we describe our APUMPEDI (stands for “Approximating Pan Matrix Profile under Unnormalized Euclidean Distance by Interpolation”) algorithm to compute the PMP. As a generalization of the LINKUMP algorithm [7],

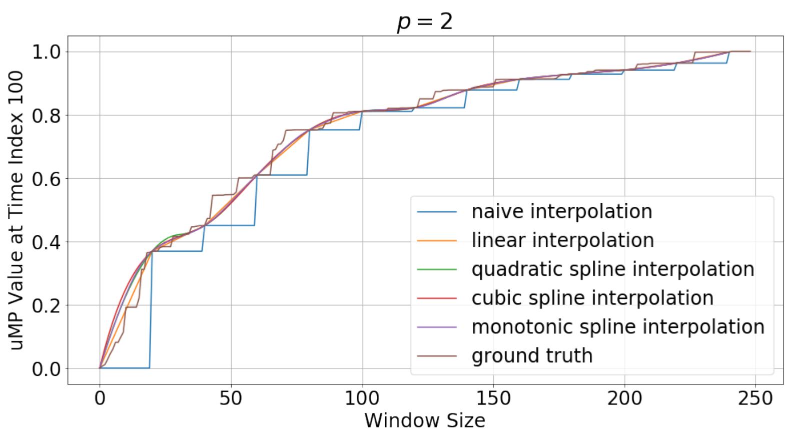


Fig. 2: A comparison of different interpolation curves for the PMP of the synthesized time series (depicted in Fig. 1a) under unnormalized ℓ_2 distance (with fixed time index 100).

the APUMPEDI algorithm utilizes various interpolation methods including, but not limited to, linear interpolation.

Roughly speaking, our algorithm works as follows. Once we have computed the MPs for a set of selected subsequence lengths, we do interpolation for each and every missing subsequence length; e.g., if we have done computing the MPs for the following 5 subsequence lengths, $[10, 20, 30, 40, 50]$, then we do interpolation for the corresponding intervals $(10, 20)$, $(20, 30)$, $(30, 40)$, and $(40, 50)$, respectively, to obtain approximate MPs for subsequence lengths within these intervals. In Fig. 2, we visualize the interpolation procedure for the synthesized time series example (shown in Fig. 1a), where both the exact and approximate MP functions (with respect to subsequence lengths) are depicted for time index 100, and various interpolation methods (described in Sec. 3.2) are tested. It is worth pointing out that our idea is essentially different from that of the SKIMP algorithm [5]; the SKIMP algorithm orders the selected subsequence lengths and computes MPs for only a portion of them from the beginning.

We are now in a position to formalize the algorithm. Let L (resp., U) denote the minimum (resp., maximum) subsequence length such that $2 \leq L < U \leq n-1$, $S \in (0, U-L)$ the step size, and $\theta \in (0, 1]$ the completion rate for the PMP computation. We wrap up the steps as Alg. 1, where Lines 10 through 18 describe the interpolation procedure. We apply the algorithm based on a double-ended queue (for the case where $p = \infty$; see [4]) or the dynamic programming based algorithm (for the case where $1 \leq p < \infty$; see [3]) for the `uMP` subroutine (Line 8 in Alg. 1), which computes the Matrix Profile of a given time series X, for a given subsequence length m, under the unnormalized ℓ_p distance. For the `Interpolate` subroutine (Line 13), we elaborate various options in Section 3.2. It is seen that the time complexity of this algorithm is $O(\theta |M| n^2)$, and the space complexity is $O(|M|n)$, where $|M| = \lceil (U - L)/S \rceil$ is the number of all predesignated candidate subsequence lengths.

Algorithm 1 Approximating Pan Matrix Profile under Unnormalized Euclidean Distance by Interpolation

```
 1: procedure APUMPEDI($X, L, U, S, \theta, p$)
 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:     $M' \leftarrow \text{range}\big(L, U+1, \lfloor \frac{1}{\theta} \rfloor \cdot S\big)$
 6:     for $m$ in $M'$ do
 7:         $i \leftarrow m@M$
 8:         $P[i,:] \leftarrow \texttt{uMP}(X, m, p)$
 9:     end for
10:     $x_{vec} \leftarrow M'$
11:     for $j$ in range$(0, n, 1)$ do
12:         $y_{vec} \leftarrow [P[m@M, j]$ for $m$ in $M']$
13:         $f(\cdot) \leftarrow \texttt{Interpolate}(x_{vec}, y_{vec})$
14:         for $m$ in $M \setminus M'$ do
15:             $i \leftarrow m@M$
16:             $P[i,j] \leftarrow f(m)$
17:         end for
18:     end for
19:     return $P$
20: end procedure
```

3.2 Interpolation Methods

Assume we are given a set $\mathcal{A}$ of $n+1$ data points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, where $x_i \neq x_j, \forall i \neq j$. In this work we apply the following interpolation methods.

Naive Interpolation As a baseline method, the naive interpolation simply returns the previous value of the point; that is,

$$f(x) = y_i, \forall x \in (x_i, x_{i+1}), i = 0, 1, \ldots, n-1. \tag{3}$$

Linear Interpolation Linear interpolation on the set of data points $\mathcal{A}$ is defined as the concatenation of linear interpolants between each pair of data points. Formally, we have

$$\begin{aligned} f(x) = \ & y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \cdot (x - x_i), \\ & \forall x \in (x_i, x_{i+1}), i = 0, 1, \ldots, n-1. \end{aligned} \tag{4}$$

We note that when linear interpolation is applied, the APUMPEDI algorithm works exactly the same as the LINKUMP algorithm.

Spline Interpolation Spline interpolation is a type of piecewise polynomial interpolation. In particular, rather than fitting a single, high-degree polynomial to all of the data points at once, spline interpolation fits low-degree polynomials (called splines) to small subsets of the data points. Spline interpolation typically outperforms polynomial interpolation due to the fact that the error of interpolation can be small even when applying low-degree polynomials for the spline [8]. It also protects against the Runge phenomenon [9], in which oscillation would occur between data points when using high-degree polynomials as interpolants. In this paper, we use both quadratic and cubic spline interpolation.

Remark 1 *We have tried using polynomial interpolation, and found that the resulting approximate PMP would significantly deviate from the true PMP. Thus, we will not include it when presenting numerical results.*

Monotonic Cubic Interpolation Because of the monotonicity of PMP (see Thm. 1), it is desirable to use monotonic splines to interpolate data points. To that end, we apply the Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) [10].

4 Numerical Results

In this section, we present results from extensive numerical experiments. The metric for assessment of the algorithm's approximation accuracy is RMSE (Root Mean Square Error) with respect to the exact PMP (i.e., $\theta = 1$). We conducted the experiments on both real-world and synthesized data sets.

4.1 Results for Mars Curiosity Rover Data Set

First, we present numerical results from experiments on a real-world data set. In particular, we use the time series of the Mars Curiosity Rover $A1 - A3, D1 - D3, E1 - E3, T1 - T3$, $P1 - P3$ [6]. After resampling, these 15 time series each has a length around 4000. As a further preprocessing step, we scale each time series to make all values be between 0 and 1.

We set $L = 3$, $U = 2000$, $S = 10$, and record results for $\theta \in \{0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1\}$. In Fig. 3, we show a comparison of the RMSE versus Rate of Completion (i.e., θ) curves of the APUMPEDI algorithm with different interpolation methods for PMP approximation under unnormalized ℓ_p distance, where $p = \infty, 1, 2, 3, 4, 5$, respectively. The RMSE scores are obtained by averaging over the results from the 15 real-world time series for each rate of approximation, and we also show the standard deviations for these scores as error bars.

It is seen that, relatively speaking, in all scenarios, the naive interpolation method would lead to the worst RMSE scores, while the other interpolation methods would lead to close RMSE scores. Moreover, the smaller the rate of completion, the more the advantages of the monotonic spline interpolation method over other alternatives. However, for this specific real-world data set, for all scenarios we see that all interpolation methods other than the naive one would lead to very close RMSE scores. This might due to the fact that the 15 selected time series follow similar probability distributions.

4.2 Results for Synthesized Data Sets

To see how our APUMPEDI algorithm perform on data sets with essentially various stochasticities, we next conduct experiments using synthesized time series. In particular, we independently generate 15 time series denoted by $Z^{(k)}, k = 1, \ldots, 15$, each of which has a length 500. Their underlying probability distributions are specified as follows:

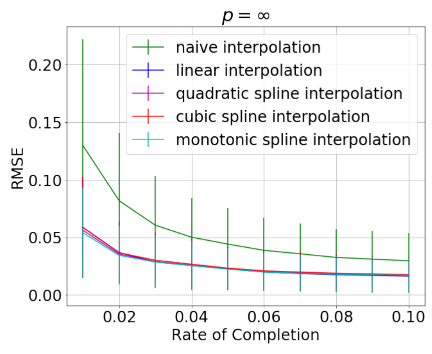
- $Z^{(1)}, Z^{(2)}, Z^{(3)}$: uniform distribution over $[0, 10)$;
- $Z^{(4)}, Z^{(5)}, Z^{(6)}$: normal distribution with parameters $(2, 7)$;
- $Z^{(7)}, Z^{(8)}, Z^{(9)}$: exponential distribution with parameter 3;
- $Z^{(10)}, Z^{(11)}, Z^{(12)}$: gamma distribution with parameters $(4, 9)$;
- $Z^{(13)}, Z^{(14)}, Z^{(15)}$: beta distribution with parameters $(1, 5)$.

Similar to what we did for the Mars Curiosity Rover data, as a preprocessing step, we scaled each generated time series to make all values be between 0 and 1. We set $L = 3$, $U = 250$, $S = 1$, and recorded results for $\theta \in \{0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1\}$. In Fig. 4, we show a comparison of the RMSE versus Rate of Completion (i.e., θ) curves of the APUMPEDI algorithm with different interpolation methods for PMP approximation under unnormalized ℓ_p distance, where $p = \infty, 1, 2, 3, 4, 5$, respectively. The RMSE scores are obtained by averaging over the results from the 15 synthesized time series for each rate of approximation, and we also show the standard deviations for these scores as error bars.

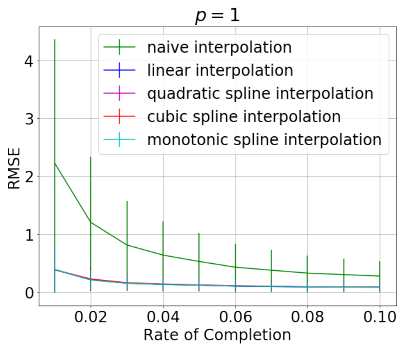
Similar to the results from the Mars Curiosity Rover data, it is seen that, relatively speaking, in all scenarios, the naive interpolation method would lead to the worst RMSE scores, while the other interpolation methods would lead to close RMSE scores, especially for the case where $p = 1$ (i.e., under the ℓ_1 distance). Moreover, the smaller the rate of completion, the more the advantages (resp., disadvantages) of the monotonic spline interpolation method (resp., the linear interpolation method) over other alternatives. Based on the above observations, we suggest using the PCHIP monotonic spline interpolation in practice, especially when needing to set an extremely small completion rate.

4.3 CPU Time for Pan Matrix Profile Computation

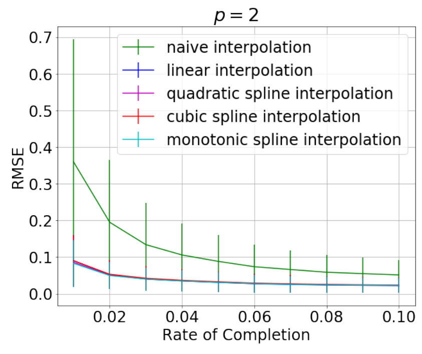
To approximate the PMP, similar to the SKIMP algorithm and the LINKUMP algorithm, our APUMPEDI algorithm also has an "anytime" property [5]; to be specific, at any time, the algorithm could be terminated, resulting in an approximation of the exact PMP. Depending on how large the completion rate θ is, the execution time of the APUMPEDI algorithm could be various. For example,



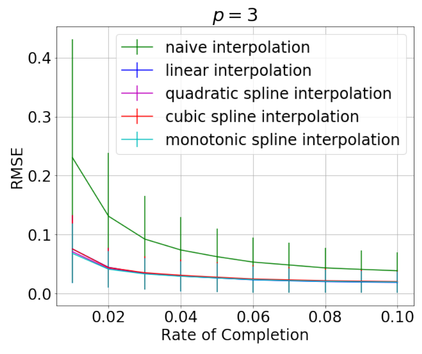
(a)



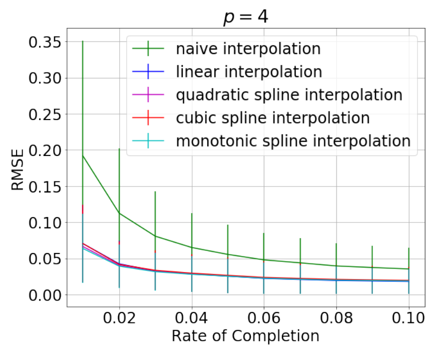
(b)



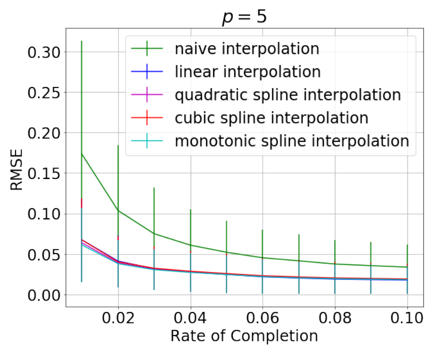
(c)



(d)

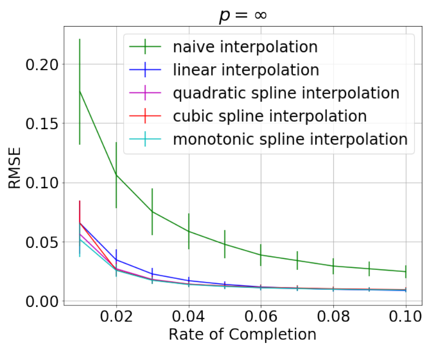


(e)

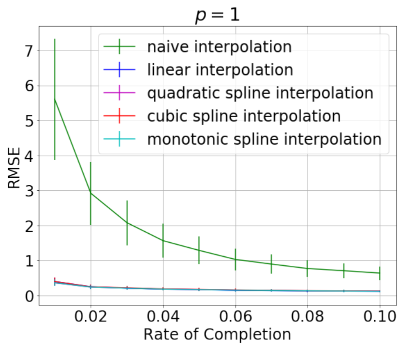


(f)

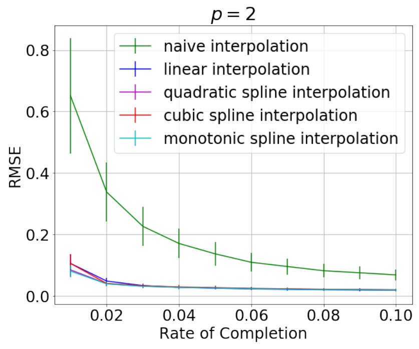
Fig. 3: A comparison of the RMSE versus Rate of Completion (i.e., θ) curves of the APUMPEDI algorithm with different interpolation methods for PMP approximation under unnormalized ℓ_p distance, where $p = \infty, 1, 2, 3, 4, 5$, respectively. Here we use the time series of the Mars Curiosity Rover $A1 - A3, D1 - D3, E1 - E3, T1 - T3$, $P1 - P3$ [6]. The RMSE scores are obtained by averaging over the results from the 15 real-world time series for each rate of approximation, and we also show the standard deviations for these scores as error bars.



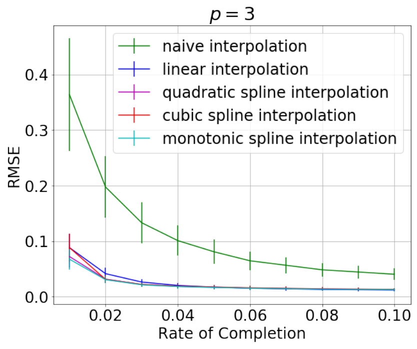
(a)



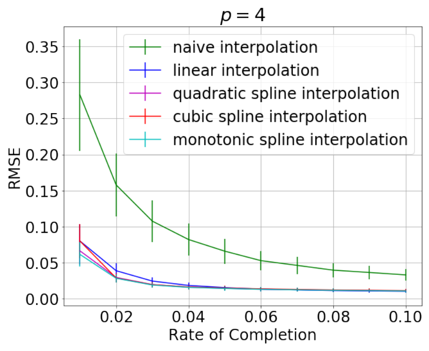
(b)



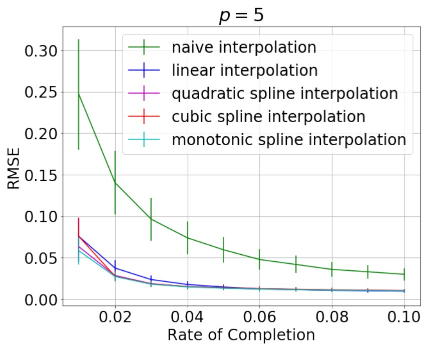
(c)



(d)



(e)



(f)

Fig. 4: A comparison of the RMSE versus Rate of Completion (i.e., θ) curves of the APUMPEDI algorithm with different interpolation methods for PMP approximation under unnormalized ℓ_p distance, where $p = \infty, 1, 2, 3, 4, 5$, respectively. The RMSE scores are obtained by averaging over the results from the 15 synthesized time series for each rate of approximation, and we also show the standard deviations for these scores as error bars.

for a time series with length 4000, if setting $L = 3, U = 500, S = 1, \theta = 0.05$, the APUMPEDI 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. Our experimental results suggest that, in practice, we could set θ to relatively small values (say 0.01, 0.05), while still achieving good enough performance (in terms of the RMSE scores).

5 Conclusion

We have devised an approximation algorithm called APUMPEDI to compute the Pan Matrix Profile (PMP) under the unnormalized Euclidean distance. It combines double-ended queue based or dynamic programming based Matrix Profile (MP) algorithms and various interpolation methods. We have validated its efficiency and effectiveness through extensive numerical experiments on both real-world and synthesized data sets.

References

1. Yeh, C.C.M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H.A., Silva, D.F., Mueen, A., Keogh, E.: 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). (2016) 1317–1322
2. Zhu, Y., Zimmerman, Z., Senobari, N.S., Yeh, C.C.M., Funning, G., Mueen, A., Brisk, P., Keogh, E.: 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) 739–748
3. Akbarinia, R., Cloez, B.: Efficient matrix profile computation using different distance functions. arXiv preprint arXiv:1901.05708 (2019)
4. Zhang, J., Nikovski, D.: Algorithms for fast computation of matrix profiles of time series under unnormalized euclidean distances. In: International Conference on Applied Statistics and Data Analytics, submitted. (2022)
5. Madrid, F., Imani, S., Mercer, R., Zimmerman, Z., Shakibay, N., Keogh, E.: 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) 175–182
6. Nakamura, T., Imamura, M., Mercer, R., Keogh, E.: Merlin: Parameter-free discovery of arbitrary length anomalies in massive time series archives. In: 2020 IEEE International Conference on Data Mining (ICDM). (2020) 1190–1195
7. Zhang, J., Nikovski, D.: Algorithms for fast computation of pan matrix profiles of time series under unnormalized euclidean distances. In: International Conference on Applied Statistics and Data Analytics, submitted. (2022)
8. Hall, C.A., Meyer, W.W.: Optimal error bounds for cubic spline interpolation. Journal of Approximation Theory **16**(2) (1976) 105–122
9. Epperson, J.F.: On the Runge example. The American Mathematical Monthly **94**(4) (1987) 329–341
10. Fritsch, F.N., Butland, J.: A method for constructing local monotone piecewise cubic interpolants. SIAM journal on scientific and statistical computing **5**(2) (1984) 300–304