Matrix Profile XII: MPdist: A Novel Time Series Distance Measure to Allow Data Mining in More Challenging Scenarios

Shaghayegh Gharghabi, Shima Imani, Anthony Bagnall[1], Amirali Darvishzadeh, Eamonn Keogh
Department of Computer Science and Engineering
University of California, Riverside, CA, US, [1]University of East Anglia, Norwich, UK
sghar003@ucr.edu, siman003@ucr.edu, anthony.bagnall@uea.ac.uk, darvisha@cs.ucr.edu, eamonn@cs.ucr.edu

***Abstract*— At their core, many time series data mining algorithms can be reduced to reasoning about the *shapes* of time series subsequences. This requires a distance measure, and most algorithms use Euclidean Distance or Dynamic Time Warping (DTW) as their core subroutine. We argue that these distance measures are not as robust as the community believes. The undue faith in these measures derives from an overreliance on benchmark datasets and self-selection bias. The community is reluctant to address more difficult domains, for which current distance measures are ill-suited. In this work, we introduce a novel distance measure *MPdist*. We show that our proposed distance measure is much more robust than current distance measures. Furthermore, it allows us to successfully mine datasets that would defeat any Euclidean or DTW distance-based algorithm. Additionally, we show that our distance measure can be computed so efficiently, it allows analytics on fast streams.**

Keywords*— *Time Series; Distance Measure; Matrix Profile;

I. INTRODUCTION

Most time series data mining algorithms, including algorithms for clustering, similarity search, many variants of classification, rule-discovery, and anomaly detection, are at their core algorithms that reason about the similarity of time series subsequences. Such reasoning requires a distance measure, and most algorithms use Euclidean Distance or DTW as their core subroutine [1][2][3]. We argue that these distance measures are not as robust as commonly believed. The unwarranted faith in these measures derives from:

- **Optimizing to benchmarks.** The UCR Time Series Archive is undoubtedly a useful resource for the community [5]. However, as [6] and others have noted, the data in the archive has been contrived in several ways that often make the datasets poor proxies for real-world problems. Failure to be competitive in *some* of these datasets is an excellent way to screen unpromising ideas. However, being competitive in most datasets in the archive does not necessarily mean the proposed distance measure will be useful in real-world deployments.
- **Self-selection bias.** The community remains reluctant to consider difficult domains, for which current distance measures are unsuited. Consider the snippets of data shown in Fig. 1, which shows eight examples of the same insect behavior. The reader can quickly generalize from these examples as to what constitutes the targeted behavior. We will show that both Euclidean Distance and DTW will fail here.

In this work, we introduce a novel distance measure, MPdist (Matrix Profile distance). We show that MPdist is more robust than current distance measures and allows us to tackle datasets that would defeat any Euclidean or DTW based algorithm.

Note that while we critique the overreliance in the UCR archive benchmarks as an indicator of the progress in time series data mining, this disparagement is not born out of "sour grapes." As we will show in [7], the MPdist produces highly competitive results on these benchmarks. Beyond this, we show that our measure has properties that allow it to tackle much more complex datasets. The useful properties of the MPdist include:

- Ability to compare time series of different lengths.
- Being robust to spikes, dropouts, wandering baseline and missing values, and other issues that are common outside of benchmark datasets.
- The invariances to *amplitude* and *offset* offered by DTW and Euclidean distance [3], as well as additional invariances, including *phase* invariance, *order* invariance, *liner trend* invariance and *stutter* invariance.
- Ability to be computed *very* efficiently, allowing great scalability.

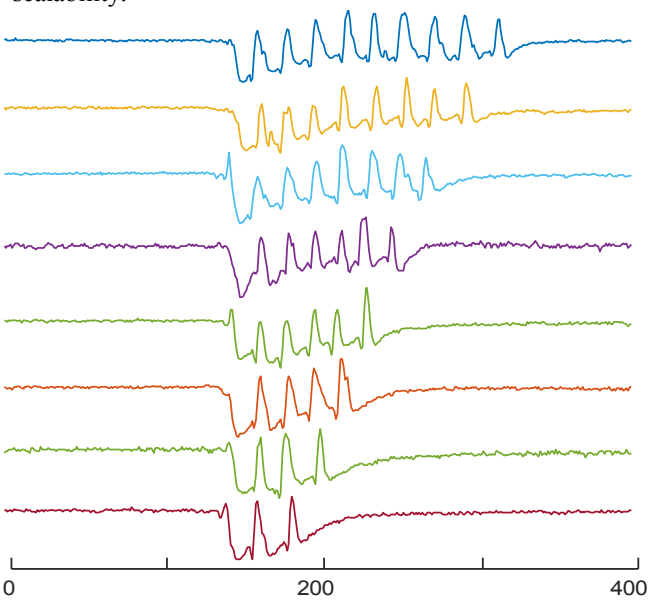


Fig. 1: Eight examples of the Phloem-Ingestion behavior of an Asian citrus psyllid, as measured by an electrical penetration graph (EPG) apparatus [16].

While this pattern is easy for a human to learn ("*from a baseline, a sudden drop, followed by two to nine peaks, as the value returns to the baseline*"), this type of behavior is very difficult to model with distance-based algorithms that use current distance measures. To illustrate this, in Fig. 2.*left* we clustered three examples of Phloem-Ingestion behavior with three smoothed random walks.

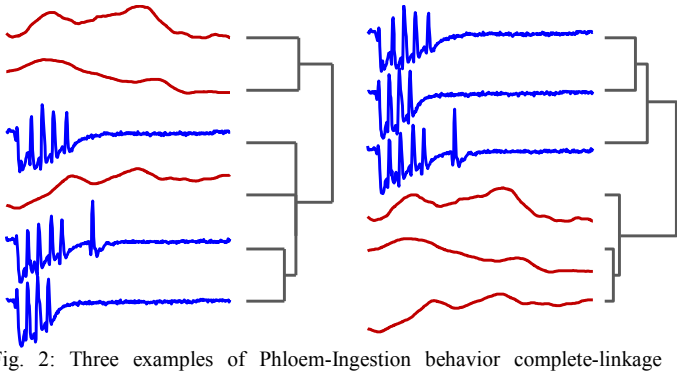


Fig. 2: Three examples of Phloem-Ingestion behavior complete-linkage clustered with three smoothed random walks (all of length 600), using (*left*) Euclidean distance and (*right*) our proposed distance measure, the MPdist.

The results for Euclidean distance are surprisingly poor; the clustering looks essentially random. Note the problem is not solved by using DTW or other measures. While DTW can "warp" out-of-alignment peaks, it cannot warp, for example, three peaks to five peaks. Thus, two peaks must be left "unexplained," incurring a high distance cost and swamping any similarity that exists. Similar remarks apply to *k*-Shape [10] and other phase invariant measures. We note in passing that near identical issues have been noted for sign language recognition. For example, "*the number of* [sub-shapes] *contained in a sign can vary among signers due to personal signing preference*," manufacturing processes, etc [10]. While our proposed measure is completely distinct from DTW, it does share some similarities with it. In particular, the MPdist:

- Subsumes Euclidean distance as a special case [11].
- On essentially all datasets, achieves accuracy greater than or equal to Euclidean distance. This is an unsurprising consequence of the previous point.
- Requires just a single, easy-to-learn parameter.
- It is a *measure* but not a *metric*; this non-metricity is an unavoidable consequence of the invariances it supports. As we explain in Section III.A, this non-metricity is highly desirable.
- While a *single* comparison is expense (relative to Euclidean distance), the *amortized* cost of subsequence search is relatively cheap, essentially the same as Euclidean distance.

Given this, we believe that the impact of MPdist on time series data mining may be similar to DTW's impact during the last decade [11].

The remainder of this paper is organized as follows. In Section II we introduce the necessary definitions and notations to understand our contributions. In Section III, we introduce the MPdist, explain its properties and its relationship to other distance measures, and show how we can accelerate MPdist subsequence searches. Section IV offers a detailed empirical evaluation of our ideas. Finally, in Section V we offer conclusions and directions for future work.

II. BACKGROUND AND RELATED WORK

In this section, we introduce all the necessary definitions and notations, and consider related work.

A. Definitions

Definition 1: A *Time Series* ($\mathbf{T} = t_1, t_2, \ldots, t_n$) is a sequence of n real values.

Our proposed distance measure will quantify the distance between two time series based on local subsections called *subsequences*.

Definition 2: A *subsequence* ($\mathbf{T}_{i,L}$) is a contiguous subset of values with length L, starting from position i in time series $\mathbf{T}$; the subsequence $\mathbf{T}_{i,L}$ is in form $\mathbf{T}_{i,L} = t_i, t_{i+1}, \ldots, t_{i+L-1}$, where $1 \leq i \leq (n - L + 1)$ and L is a user-defined subsequence length with value in range of $3 \leq L \leq |\mathbf{T}|$.

We chose 3 as the shortest permissible value for L, because it is not meaningful to normalize time series that are shorter, and non-normalized time series are rarely useful for measuring distances [11].

For our proposed algorithm, it is required to extract *all* subsequences. This is achieved using a *sliding window*.

Definition 3: *Sliding window:* All possible subsequences of a given time series $\mathbf{T}$ can be extracted by sliding a window of size L across $\mathbf{T}$. There are $(n - L + 1)$ such subsequences, which we denote as *SubseqNum*.

The *time series similarity join*, also known as all-pairs-similarity-search, is defined in [18]. Due to its importance as a subroutine in our proposed method, we briefly review it here. Intuitively, the task of the similarity join is "*Given a collection of data objects, retrieve the nearest neighbor for each object*"[19]. The similarity join set is defined on a set of all possible subsequences of a time series, referred to as the *All-Subsequences Set*.

Definition 4: An *All-Subsequences Set* ($\boldsymbol{A}$) is a set of all possible subsequences of a time series $\mathbf{T}$. The subsequences are obtained from sliding a window of length L across $\mathbf{T}$. Thus,

$$\boldsymbol{A} = \{\mathbf{T}_{1,L}, \mathbf{T}_{2,L}, \ldots, \mathbf{T}_{n-L+1,L}\}.$$

At a high level, our proposed distance measure will compute the distance between two time series, $\mathbf{T}_A$ and $\mathbf{T}_B$, by aggregating the distances between their All-Subsequences Sets. For this purpose, we need to find the nearest neighbor for each subsequence in $\boldsymbol{A}$ within $\boldsymbol{B}$ (and vice versa). To determine if a member of set $\boldsymbol{B}$ is the nearest neighbor of a member in set $\boldsymbol{A}$ we use *1NN-Join Function*.

Definition 5: *1NN-Join Function* is defined as the first nearest neighbor (1NN) between two subsequences $\boldsymbol{A}[i]$ and $\boldsymbol{B}[j]$. The 1NN-join function $\theta_{1NN}(\boldsymbol{A}[i], \boldsymbol{B}[i])$ returns "true", if $\boldsymbol{B}[j]$ is the nearest neighbor of $\boldsymbol{A}[i]$.

The 1NN-join function is a similarity join operator, which is applied on two All-Subsequences Sets; as a result, we can create *AB similarity join set*.

Definition 6: *AB Similarity Join Set* ($\boldsymbol{J_{AB}}$) is a set containing each subsequence in $\boldsymbol{A}$ paired with its corresponding nearest neighbor in $\boldsymbol{B}$, in which $\boldsymbol{A}$ and $\boldsymbol{B}$ are two sets of All-Subsequences. $\boldsymbol{J_{AB}}$ is defined as:

$$\boldsymbol{J}_{\boldsymbol{AB}} = \{\langle \boldsymbol{A}[i], \boldsymbol{B}[i] \rangle | \theta_{1NN}(\boldsymbol{A}[i], \boldsymbol{B}[i])\}.$$

The similarity join set contains tuples, with each subsequence in set $\boldsymbol{A}$ from time series $\mathbf{T}_A$, and its nearest neighbor in set $\boldsymbol{B}$ from time series $\mathbf{T}_B$. Note that some subsequences in $\mathbf{T}_B$ may not be used as neighbors to any elements from $\mathbf{T}_A$, and some subsequences in $\mathbf{T}_B$ may be used more than once. This is because in general $\boldsymbol{J_{AB}} \neq \boldsymbol{J_{BA}}$.

For our proposed distance measure, we need to obtain the distance between *each* pair in the similarity join set. After obtaining the nearest neighbor of each subsequence in a set, an array which stores the Euclidean *distance* of each pair is called *Matrix Profile* [18][19].

Definition 7: *Matrix Profile* ($\boldsymbol{P_{AB}}$) is an array in which the Euclidean distance between each pair in $\boldsymbol{J_{AB}}$ is stored. The length of $\boldsymbol{P_{AB}}$ is $(n - L + 1)$ or *SubseqNum*.

Without loss of generality, we assume that the two time series $\mathbf{T}_A$ and $\mathbf{T}_B$ have the same length. Moreover, it rarely makes sense to measure the similarity of time series with significantly different lengths (not to be confused with *subsequence search*, which we show how to perform in Section III.C). Note the matrix profile is slightly shorter than the time series that was used to create it.

Fig. 3 shows the $\boldsymbol{P_{AB}}$ of two time series $\mathbf{T}_A$ and $\mathbf{T}_B$. As shown, since $\mathbf{T}_A$ and $\mathbf{T}_B$ have a mostly common structure, their $\boldsymbol{P_{AB}}$ has low values, except for the region where sine-waves change to triangular waves, in which case there is no "explanation" from $\mathbf{T}_B$ in $\mathbf{T}_A$. Hence, there is a bump in $\boldsymbol{P_{AB}}$ indicating a high value.

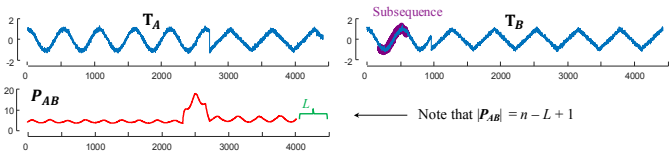


Fig. 3: *top*) Two time series $\mathbf{T}_A$ and $\mathbf{T}_B$. *bottom*) $\boldsymbol{P_{AB}}$ of two time series $\mathbf{T}_A$ and $\mathbf{T}_B$ with $L = 400$. Since there is no corresponding section in $\mathbf{T}_A$ at the point of signal change from $\mathbf{T}_B$, there is a bump in $\boldsymbol{P_{AB}}$.

The time complexity to calculate $\boldsymbol{P_{AB}}$ for two equal-length time series when L is much shorter than n is $O(n^2)$ [19]. If the length of L is a significant fraction of n, then the time complexity grows to $O((n - L + 1) \times n)$. In the limit, when $L = n$, this degenerates to the special case of the Euclidean distance between the two time series, which takes $O(n)$. The following notation summarizes this:

$$\textit{Time complexity } \boldsymbol{P_{AB}} = \begin{cases} O(n^2), & L \leq n \\ O\big((n - L + 1) \times n\big), & L < n \\ O(n), & L = n \end{cases}$$

As L approaches n, the time complexity approaches linear time. To make our distance measure between $\mathbf{T}_A$ and $\mathbf{T}_B$ symmetric, we will need to compute both $\boldsymbol{J_{AB}}$ and $\boldsymbol{J_{BA}}$; we denote this operation as the *ABBA Similarity Join*.

Definition 8: *ABBA Similarity Join* ($\boldsymbol{J_{ABBA}}$) is a set containing pairs of each subsequence in $\boldsymbol{A}$ with its nearest neighbor in $\boldsymbol{B}$ and vice versa.

Note that if a subsequence in $\boldsymbol{A}$ (denoted as $\mathbf{T}_{A,i}$) is the nearest neighbor of a subsequence in $\boldsymbol{B}$ (denoted with $\mathbf{T}_{B,j}$), the reverse of that may not be true; that is, $\mathbf{T}_{B,j}$ may not be the nearest neighbor of $\mathbf{T}_{A,i}$. An array which stores all distances in ABBA similarity join set is *Join Matrix Profile*.

Definition 9: *Join Matrix Profile* ($\boldsymbol{P_{ABBA}}$) is an array containing the Euclidean distance for each pair in $\boldsymbol{J_{ABBA}}$. The length of the $\boldsymbol{P_{ABBA}}$ is $2 \times (n - L) + 2$.

The join matrix profile has distances for both similarity joins $\boldsymbol{J_{AB}}$ and $\boldsymbol{J_{BA}}$; thus, it is symmetric in terms of the order of time series. As a result, the distance calculated based on $\boldsymbol{J_{ABBA}}$ between $\mathbf{T}_A$ and $\mathbf{T}_B$ is also equal. Fig. 4 shows an illustration of the $\boldsymbol{P_{ABBA}}$ of two time series $\mathbf{T}_A$ and $\mathbf{T}_B$ with the same length.

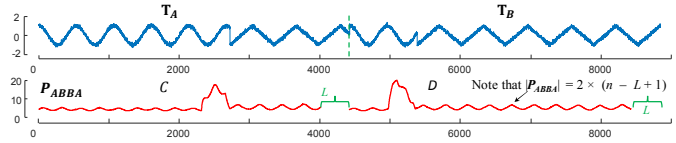


Fig. 4: *top*) The concatenation of two time series $\mathbf{T}_A$ and $\mathbf{T}_B$. *bottom*) $\boldsymbol{P_{ABBA}}$ of two time series $\mathbf{T}_A$ and $\mathbf{T}_B$ with $L = 400$. The distance between each subsequence from $\mathbf{T}_A$ and its nearest neighbor from $\mathbf{T}_B$ is calculated in C, and the reverse in D. There is a gap between C and D at the middle, because the length of remaining data in $\mathbf{T}_A$ is less than the subsequence length; thus, the distance cannot be calculated.

As we will show in the next section, this data structure $\boldsymbol{P_{ABBA}}$ contains all the information we need to compute the MPdist.

III. THE MPDIST

Intuitively, our proposed distance measure considers two time series to be similar if they share many similar subsequences, regardless of the *order* of matching subsequences. As the reader will readily appreciate, all such information is available in $\boldsymbol{P_{ABBA}}$ (Definition 9); we thus consider the question of how to best exploit it.

If we based the distance on the *largest* value in $\boldsymbol{P_{ABBA}}$, the measure would become brittle at a single noisy spike or dropout that appeared in either time series. At the other extreme, if we based the distance on the *smallest* value in $\boldsymbol{P_{ABBA}}$, there would be little discrimination between most time series. This would be like a distance measure for English sentences that only looked at a single word in common. Since most English sentences contain "*the*" or "*a*," almost all sentences would be equidistant.

Instead of these two extremes, we propose to consider the value of the k^{th} smallest number as the reported distance. Concretely, we set the value of k to be equal to 5 percent of $2 \times n$, which is the length of concatenation of $\mathbf{T}_A$ and $\mathbf{T}_B$. While we discuss this exact value on the [7], the choice of a small value helps us reduce the effect of noise and distorted values in our distance measure algorithm.

In the case where the length of a subsequence is close to the length of full time series, then the length of $\boldsymbol{P_{ABBA}}$ is less than 5 percent of the length of two time series. In such cases, we used the maximum value of sorted array $\boldsymbol{P_{ABBA}}$ as the distance. The following formula illustrates this:

$$MPdist = \begin{cases} k^{th} \text{ value of sorted } \boldsymbol{P_{ABBA}}, & |\boldsymbol{P_{ABBA}}| > k \\ \max(\boldsymbol{P_{ABBA}}), & |\boldsymbol{P_{ABBA}}| \leq k \end{cases}$$

Note that this implies that when the length of a subsequence is equal to the length of full time series, the MPdist degenerates to the classic Euclidean distance. This is because for that setting, the $\boldsymbol{P_{ABBA}}$ has exactly two equal values, each of which is the Euclidean distance between the entire lengths of $\boldsymbol{A}$ and $\boldsymbol{B}$. Thus, the max ($\boldsymbol{P_{ABBA}}$) is just the Euclidean distance.

Where appropriate, to denote the particular value of the L parameter used in the given experiment, we wrote MPdist_L. For example, in Fig. 2 we used MPdist_{20} (although any value under 60 works well).

In the following sections, we explain the properties of MPdist, and its relationship with the other distance measures. Then we show how we can significantly accelerate query-by-content under the MPdist.

A. *On the Lack of Metric Properties for MPdist*

Our MPdist is a *measure*, not a *metric*. In particular, it does not obey the triangular inequality. The lack of the triangular inequality property is potentially worrisome for two reasons:

1. Many speedup techniques for query-by-content, clustering, anomaly detection, etc., implicitly or explicitly exploit the triangular inequality to prune the search space, which otherwise becomes intractable for large datasets [3].
2. Without the triangular inequality property, one can produce distance evaluations that defy human intuitions, such as claiming that objects **A** and **B** are similar, and **A** and **C** are similar, but **B** and **C** are very *dissimilar*.

To some extent, we may be comforted by noting that DTW is *also* not a metric; yet, it can be sped up by many other techniques [11]. Furthermore, it has been empirically confirmed as a highly competitive measure for most time series problems in several large-scale comparisons [5].

It might be argued that DTW is *almost* a metric. This is especially true if we have a narrow warping window, which is strongly advocated due to other reasons [11]. However, we believe that there are situations/datasets that *require* a distance measure which can strongly violate the triangular inequality. To see this in practice, let us first consider an analogous problem in string matching. Consider the following common American girl names:

`Lisabeth, Beth, Lisa, Maryanne, Anne, Mary`

If asked to cluster these names into two groups, we would surely expect [{`Lisabeth`, `Lisa`, `Beth`}, {`Maryanne`, `Mary`, `Anne`}]. However, it is unlikely that a distance measure that insists on the triangular inequality would give us "`Beth`" and "`Lisa`" in the same group, since they do not share a single character with each other. Yet, both share one character with "`Anne`".

As Fig. 5 shows, we can create perfect analogues of such data in the time series domain by recording the Y-axis of *handwritten* versions of these names. They cannot be correctly clustered by the Euclidean distance, as expected. However, as Fig. 5 also shows, the MPdist *can* correctly cluster the data here. The property that causes the MPdist to violate the triangular inequality is an important one. The MPdist measure is able to *ignore* some of the data. In contrast, Euclidean distance and DTW must explain *all* the data in the sequences being compared.

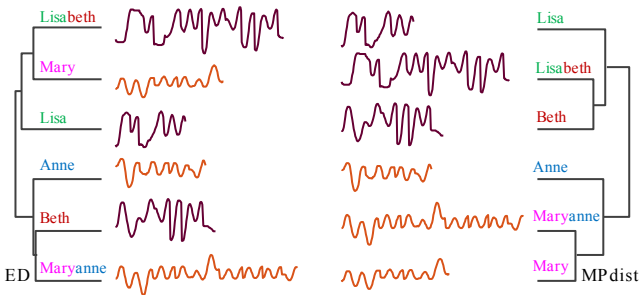


Fig. 5: A visual explanation as to why violating the triangular inequality can be useful. *top*) We created time series versions of the girl's names examples (see main body text), by capturing the Y-axis of cursive handwritten versions of the names. *bottom*) The (equalized length variant) of Euclidean distance fails to cluster such data correctly, but the Mpdist_{30} has no difficulties.

As noted above, the other cited reason is for the desirability for scalability. As we will show in Section III.C, this is not an issue for us. In fact, we can compute the MPdist at least three orders of magnitude faster than real time in realistic settings.

B. ***The Relationship to Other Distance Measures***

Having seen the MPdist, it may be useful to place it in the context of the other major distance measures, which are:

- **Euclidean Distance**: Two time series are considered similar if one is a noisy version of the other [2][3][17][19].
- **Dynamic Time Warping**: Two time series are considered similar if, *after* adjusting the non-linear time axis, they can be made similar under Euclidean Distance [11].
- **LCSS Distance**: Two time series are considered similar if, *after* deleting some small sections from one of them, they can be made similar under the DTW Distance[1][2].
- ***k*-Shape**: Two time series are considered similar if, *after* some circular shift of the time axis, they can be made similar under Euclidean Distance[2] [10].
- **MPdist**: Two time series are similar if they share many similar subsequences under Euclidean Distance.

[1] There are several variants of LCSS proposed (under this, and other names). This is the more general explanation of such methods.

[2] This idea is simply the *cross correlation*; *k*-Shape is an algorithm that *uses* cross correlation [10]. However, we abuse terminology a little here to be consistent with the emerging literature.

This list is by no means exhaustive. Dozens of alterative measures have been introduced in the last decade [2]. However, in several rigorous comparisons, the initial enthusiasm for them has cooled [2]. Many of them are perhaps best seen as simply variants of DTW.

C. *Speeding up MPdist Search*

As noted above, the time complexity of MPdist is $O(n^2)$ in the worst case. This lethargy would be a serious problem if we wish to perform MPdist similarity searches (i.e. query-by-content) in large datasets. Similarly, DTW was introduced to the data mining community in Berndt and Clifford's famous 1994 paper [4]. However, it had almost no impact on practical applications until lower bounding searches brought its amortized time complexity down from $O(n^2)$ to just $O(n)$ [11]. In this section we will show that the MPdist is amenable to a similar acceleration for *query-by-content*.

Problem Statement: Given a query **Q** of length n and a much longer time series **T** of length m, we wish to create a distance vector ($MPdist_{vect}$) that contains the MPdist between **Q** and $\mathbf{T}_{i:i+n}$, for all i in the range 1 to m-n+1.

The $MPdist_{vect}$ is shown in red in Fig. 6. The $MPdist_{vect}$ is minimized at the location of the nearest neighbor of **Q**. More generally, this distance vector is all we need to find the k-nearest neighbors, to answer arbitrary range queries, etc.

The brute algorithm to compute $MPdist_{vect}$ is $O(mn^2)$, which is clearly untenable. However, as we shall now show, we can compute it in just $O(SubseqNum \times m)$ time.

We begin by obtaining the All-Subsequences Set of **Q** (Definition 4), and then calculating the distance between each individual subsequence in the set to every subsequence in **T**. The MASS algorithm allows us to do this very efficiently [8]. As Fig. 6 shows, this gives us *SubseqNum* Euclidean *distance profiles,* the j^{th} of which we denote as d_j.

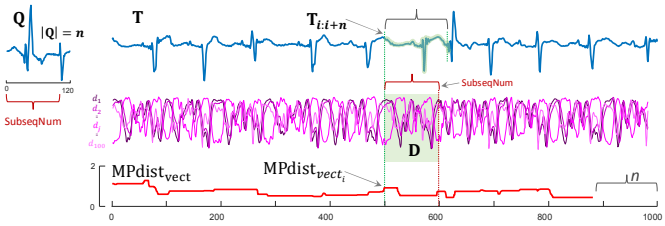


Fig. 6: A query, **Q**, with a long time series, **T**, to search. We begin by creating *SubseqNum* Euclidean distance profiles.

Fig. 6 highlights an arbitrary region of **T** beginning at $\mathbf{T}_i$ with a length of n, and the corresponding region of the distance profiles, each with the length of *SubseqNum*. For notational clarity, we envision this *SubseqNum* × *SubseqNum* region as a matrix that we call **D**.

Perhaps surprisingly, **D** contains all the information needed to compute MPdist(**Q**, $\mathbf{T}_{i:i+n}$). The key observation is that using *just* **D** we can calculate $\mathbf{P}_{AB}$ and $\mathbf{P}_{BA}$, thus obtaining $\mathbf{P}_{ABBA}$.

The steps to calculate $\mathbf{P}_{AB}$ and $\mathbf{P}_{BA}$ are as follows:

- $\mathbf{P}_{AB}$: The row-wise minimum of **D** corresponds to $\mathbf{P}_{AB}$. Recall that the first value in $\mathbf{P}_{AB}$ is the minimum distance between the first subsequence in $\mathbf{T}_A$ compared to all the subsequences in $\mathbf{T}_B$, which is the minimum of the first row of **D**. In the same manner, the remaining values of $\mathbf{P}_{AB}$ can be obtained as the minimum of all the other rows in **D**.
- $\mathbf{P}_{BA}$: The column-wise minimum of **D** corresponds to $\mathbf{P}_{BA}$. $\mathbf{P}_{BA}$ is simply the minimum distance between the subsequences in $\mathbf{T}_B$ compared to all the subsequences in $\mathbf{T}_A$, which is just the column-wise minimum of **D**.

Thus, by concatenating $\mathbf{P}_{AB}$ and $\mathbf{P}_{BA}$, we can obtain $\mathbf{P}_{ABBA}$ and therefore the MPdist. Fig. 7 illustrates this.

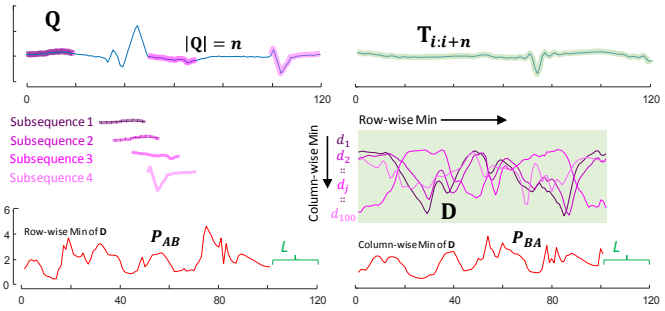


Fig. 7: Exploiting **D** to produce the $\mathbf{P}_{ABBA}$. See also Fig. 6. In this figure the length of subsequence is L = 21 this give us *SubseqNum* = 100 Euclidean distance profiles (d_1, …, d_{100}).

The time complexity for calculating a single Euclidean distance profile is $O(m\log m)$ [8]; so, the time complexity for calculating the distance profile for all subsequences is $O(SubseqNum \times m\log m)$.

At first blush, this method of computing MPdist seems to have gained us nothing. The time complexity for recreating $\mathbf{P}_{ABBA}$ in the region **D** is $O(SubseqNum^2)$; so to compute this for all sliding **D**'s in **T** would be $O(SubseqNum \times m\log m + m \times SubseqNum^2)$. However, we can optimize the algorithm to have an amortized time complexity of just $O(SubseqNum \times m)$.

The key to achieving this dramatic speed up is realizing that as we slide our query from location $\mathbf{T}_i$ to $\mathbf{T}_{i+1}$ to produce a new **D**, we do not need to recalculate everything from scratch; we just need to update a handful of values. As we slide our query one step, some points will *ingress* into **D** at the right and some points will *egress* from **D** at the left.

Concretely, for each step to the right, we have *SubseqNum* new points in distance profiles added to the **D**, and the same number removed from the **D**. Let us see how these incremental updates change $\mathbf{P}_{ABBA}$, and how we can address them:

- **Ingress:** For $\mathbf{P}_{BA}$ we can find the column-wise minimum of the last column of **D** for new arrival point in $O(SubseqNum)$. In addition, for $\mathbf{P}_{AB}$ recall that we must find the minimum of each row in **D**. This problem is equivalent to finding the classic sliding window minimum [14], which can be solved in O(1) for a single row, and in $O(SubseqNum)$ for all rows.
- **Egress:** We can easily remove the first point from $\mathbf{P}_{AB}$ and $\mathbf{P}_{BA}$ in time complexity O(1).

Thus, the amortized time complexity of obtaining the matrix profile for new arrival points is $O(SubseqNum)$. As a result, the

amortized time complexity for calculating MPdistvect between **T** and **Q** is O(*SubseqNum* × *m*).

IV. Experimental Evaluation

To ensure that our experiments are reproducible, we have built a website which contains all data/code/raw spreadsheets for the results, in addition to many experiments that are omitted here for brevity [7]. We perform many experiments that were excluded for brevity, we ask the interested reader to visit [7].

A. A Case Study in Power Demand Data

One of the areas in which time series data mining has been applied to the most in recent years is mining electrical power demand time series. In the absence of a benchmark dataset in this domain, we created one. While examining the REFIT dataset [12], we noticed that House 1 has two freezers that were individually metered. For each freezer, we extract 1,500 40-minute snippets, carefully aligning them (for the benefit of rival methods; MPdist is phase invariant) so that the increase in power demand happened at the third minute. Fig. 8 shows some examples, clustered by ED and MPdist.

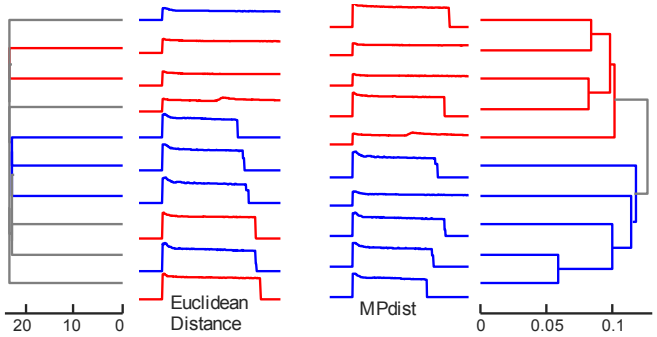
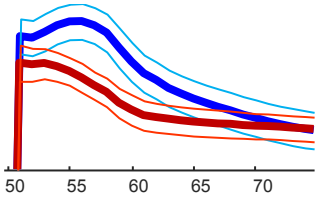


Fig. 8: Ten examples from the Freezer dataset clustered with Euclidean distance and $MPdist_{40}$.

The clustering results suggest that ED has great difficulties here. This is also true for *classification* of this data. With a 152/2848 train/test split, ED has a 35% error-rate, yet MPdist learns a parameter of 40 on the training data, and then $MPdist_{40}$ achieves a significantly better 5% error-rate. What explains such a drastic difference? In Fig. 9, we hint at the answer.



The mean values of sub-region of **class 1** and **class 2** of the Freezer dataset (bold lines), augmented by lines indicating ± one standard deviation (fine lines)

Fig. 9: For a just small sub-region of each class of the Freezer dataset, the shapes of the data is class conserved.

The ability of MPdist to focus on the relatively small amount of class conserved behavior and ignore everything else is critical in this domain, and, we suspect, in many other domains. This dataset was also donated to the UCR archive [5].

V. Conclusions

We have demonstrated that many real-word domains require a distance measure that is more robust than the current state of the art methods. We have introduced MPdist, a novel distance measure to repair this omission. We have shown that the MPdist is more robust to noise, irrelevant data, misalignment etc., than either ED or DTW. Moreover, these desirable features do not come at the cost of lethargy. Under typical assumptions, the MPdist can process data three orders of magnitude faster than real-time data streams from accelerometers or medical devices.

Finally, we have made all code and data freely available to the community (in perpetuity [7]), to allow the community to confirm and extend our findings.

ACKNOWLEDGEMENTS: We gratefully acknowledge NSF awards 1544969, 1544969, and 1510741.

References

[1] Abdoli, A., Murillo, A., Yeh, M., Gerry, A. and Keogh, E., 2018, December. Time Series Classification to Improve Poultry Welfare. In 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 2018. (forthcoming).

[2] Bagnall, A., et al, 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min. Knowl. Discov., 31(3), pp.606-660.

[3] Batista, G.E., Wang, X. and Keogh, E.J., 2011, April. A complexity-invariant distance measure for time series. In Proc' of the 2011 SIAM SDM pp. 699-710.

[4] Berndt, D.J. and Clifford, J., 1994. Using dynamic time warping to find patterns in time series. In KDD workshop (Vol. 10-16, pp. 359-370).

[5] Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A. and Batista, G., 2015. The UCR time series classification archive. URL *www.cs.ucr. edu/~ eamonn/time_series_data*

[6] Dau, H., Begum, N. and Keogh, E., 2016. Semi-Supervision Dramatically Improves Time Series Clustering under Dynamic Time Warping. In Proc' of 25th CIKM pp. 999-1008.

[7] E. Keogh, (2017). Supporting website for this paper: https://sites.google.com/site/mpdistinfo/

[8] Mueen, A. The MASS algorithm. (accessed 4-5-18) url: *www.cs.unm. edu/~ mueen/FastestSimilaritySearch.html*

[9] Murray, D., et al., 2015. A data management platform for personalised real-time energy feedback. Proc' of EEDAL'15.

[10] Paparrizos, J. and Gravano, L., 2015, May. k-shape: Efficient and accurate clustering of time series. In Proc' of the 2015 ACM SIGMOD (pp. 1855-1870). ACM.

[11] Rakthanmanon, T., et. al. 2012, August. Searching and mining trillions of time series subsequences under dynamic time warping. In Proc'of the 18th ACM SIGKDD pp. 262-70.

[12] Refitsmarthomes.org. 2018 REFIT Dataset. (Accessed 1-21-18) Available at: *www.refitsmarthomes.org/index.php/data*

[13] Rodriguez, et al. 2006. Rotation forest: A new classifier ensemble method. IEEE PAMI, 28(10), pp.1619-1630.

[14] Ruutu, J. and Kilkki, M., 2000. System and method employing last occurrence and sliding window technique for determining minimum and maximum values. U.S. Patent 6,023,453.

[15] Schäfer, P. The BOSS is concerned with time series classification in the presence of noise. Data Min. Knowl. Discov. 29(6): 1505-30 (2015).

[16] Willett, D.S., George, J., Willett, N.S., Stelinski, L.L. and Lapointe, S.L., 2016. Machine learning for characterization of insect vector feeding. PLoS computational biology, 12(11).

[17] Ye, L. and Keogh, E., 2009. Time series shapelets: a new primitive for data mining. Proc' of the 15th SIGKDD pp. 947-56.

[18] Yeh, C.C.M., et al. 2016. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View that Includes Motifs, Discords and Shapelets. Proc' of 16th IEEE ICDM, pp. 1317-22.

[19] Zhu, Y., et al. 2016. Matrix Profile II: Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins. In Proc' of the 16th IEEE ICDM, 2016 pp. 739-48.