Matrix Profile XXVIII: Discovering Multi-Dimensional Time Series Anomalies with *K* of *N* Anomaly Detection†

Sadaf Tafazoli* Eamonn Keogh*

Abstract

In recent years there has been significant progress in univariate time series anomaly detection. However, efforts to generalize this success to the multi-dimensional case have met with limited progress. The main difficultly appears to be that in any N-dimensional time series, the anomaly will generally only manifest itself on K of the time series, with $K < N$. This leads to a chicken-and-egg problem. If we knew which K time series exhibited the anomaly, it would be easy to discover its location. However, we do not know this in advance, and the search space is of size 2^N and not obviously amiable to greedy search. In this work we show a novel, simple algorithm that allows us to quickly find the best K of N anomaly subset for any value of K. Moreover, we show a simple metric that can rank the top anomaly subsets for all values of K from 1 to N. While our methods are mostly agnostic to the anomaly scoring model, for concreteness we use the Matrix Profile, and show that we can discover multi-dimensional anomalies that would escape detection by all current rival methods.

Keywords: Time Series, Anomalies, Matrix Profile methods.

1 Introduction

In the last few years, there has been significant progress in univariate time series anomaly detection. The state-of-the-art algorithms can detect anomalies that are so subtle that they would defy discovery by careful human inspection [21]. However, efforts to generalize these successes to multi-dimensional time series have not seen similar progress. The main problem appears to be that in any N-dimensional time series, the anomaly will generally only manifest itself on K of the time series with $K < N$ (typically $K \ll N$), and the inevitable small amounts of noise on the remaining N-K dimensions will tend to swamp the signal provided by the anomalous time series.

To make this concrete, consider the following simple example. We created a dataset that contains N time series, all of which are simply slightly noisy sine waves. As shown in Figure 1, we modeled a fault which induced a spike in one time series, and an unusual shape in the other, and did not affect the rest of the data. Can we detect this anomalous region?

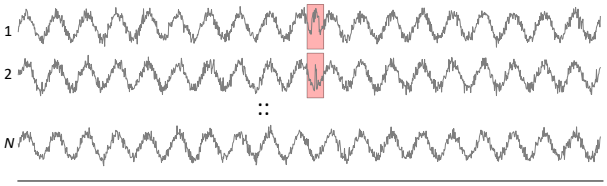

Figure 1: A set of N time series, which are slightly noisy sine waves. At one location, we have embedded anomalies in time series 1 and 2.

Given the success of anomaly scoring algorithms for the univariate case, the obvious solution to discover the anomalous region is to simply compute the anomaly scores for each time series individually, then combine those scores, in this case by adding them (although as we shall see, there are other ways). As shown in Figure 2. bottom, if $N = 2$, that is, we only consider these two-time series, then the problem is trivial.

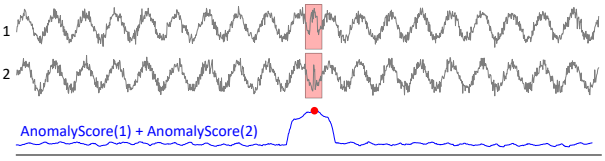

Figure 2: If $K = N$, then by simply summing the individual time series anomaly scores and finding the maximum (indicated by the red dot) we correctly predict the true anomalous region.

However, suppose that, in addition to the two above time series, we consider the others that do not contain any anomaly. Would the summed anomaly curve still reveal the correct location of the anomaly? In Figure 3 we test this question.

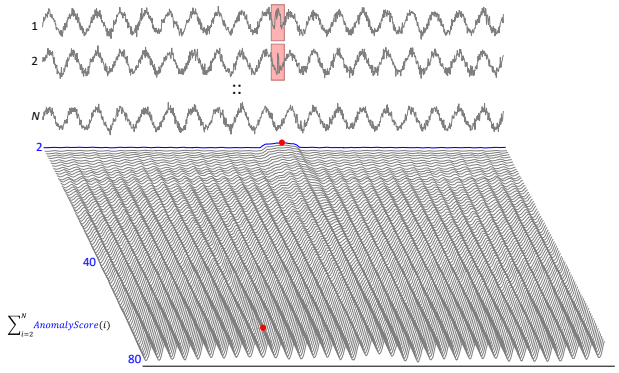

Figure 3: *top*) The time series we encountered in Figure 1. *bottom*) The anomaly scores for increasing large sets of time series, that include the two anomalous examples. Note that by the time we have seen 75 examples, the predicted location of the anomaly has moved, and is now a false positive.

If we have, say, $N = 40$, the anomaly curve has flattened out somewhat but we can still just about detect a peak at the right place. However, by the time we have $N = 75$, the anomaly location has changed to the wrong location by the cumulative effect of small amounts of noise in the N-K normal time series.

The fact that this approach does not fail until N is 75 is a testament to the robustness of the underlying anomaly scoring function, here we are using the Matrix Profile [22]. However, this is a very contrived problem with short time series. For longer time series, or for real-world time series, the swamping effect will occur much earlier. We will demonstrate our ideas on real-world problems and show that we can significantly outperform rival methods that consider only one modality.

An apparent solution to this issue is to search over all combinations of anomaly scores to find the subset that maximizes the anomaly score. However, it is clear that we must

*Computer Science and Engineering Department, University of California, Riverside, USA. {stafa002, eamonn}@ucr.edu

penalize subsets for their cardinality. In the example above, any anomaly score that we add to the correct two scores will slightly increase the maximum. In addition, it is important to realize that the nature of the problem precludes a greedy or dynamic programming search. In particular, under any reasonable anomaly score aggregation function, the highest scoring subset of size K is not necessarily a superset of the highest scoring subset of size K-1.

In this work we make the following contributions:

- We show that there are at least three different anomaly scoring aggregation functions that may be useful, depending on the circumstances.
- We demonstrate a novel search algorithm, TSADIS (Time Series Anomaly Detection through Incremental Search) that can discover the best K of N solutions, for all values of K from 1 to N, in just $O(N \times logN)$ time. In contrast, a brute-force algorithm would take $O(2^N)$.
- In many circumstances a user may wish to browse all K subsets, of size 1 to N. However, the user (which may be a person or an algorithm) may insist on selecting just one subset, the most *natural* subset that reflects the anomaly. Thus, we introduce a function that can rank and compare different sized subsets.

The rest of this paper is organized as follows. In Section 2 we review our motivations and assumptions before considering related work. In Section 4 we introduce the necessary definitions and notations, then introduce our algorithms in Section 5. Section 6 contains an extensive empirical evaluation. Finally, we offer directions for future work in Section 7.

2 Motivation

Our work is predicated on the assumption that an anomaly will generally not exhibit itself on all the time series that monitor that system (at least not initially). Consider the following examples:

- A distillation column may be monitored by 1,000+ sensors. However, the overall system can be envisaged as being comprised of multiple subsystems. These subsystems may correspond to different physical regions in the column (i.e., trays at different levels), to different physical devices (i.e., pumps or valves), or to different logical processes (i.e., heat recovery or drainage). These subsystems may be weakly or strongly coupled [9].
- In an ICU setting there may be as many as thirty sensors monitoring a patient's health. But most serious issues only manifest themselves on a subset of these time series, at least initially. For example, cardiac tamponade may present itself only on respiration and blood pressure. Hyperglycemia typically presents itself on respiration and glucometer.

3 Related Work

The topic of time series anomaly detection has seen a dramatic explosion of interest in recent years, as such it is a difficult area to survey in limited space. We refer the interested reader to[1][2][3][10][17][20] and the references therein.

There are two important points that we have gathered from our survey of the literature. The first is mostly due to a single paper [21], that forcefully suggests some of the apparent success of recently proposed algorithms may be questionable, due to severe problems with the commonly used benchmarks in this area. The second issue is noted in [6], which claims that a flaw in the most common evaluation metrics means that a "*random guess method can outperform state-of-the-art detectors*[1]". We do not weigh in on these issues, other than to state that we have not assumed the correctness of previous work, and have made an effort to avoid these issues in our work.

We have chosen to extend *time series discords*[17][22], to the K of N case, rather than one of the many other possibilities. The reason for this is that there is an increasing evidence that discords remain competitive with the state-of-the-art. Among the hundreds of time series anomaly detection algorithms proposed in the last two decades, only time series discords could claim to have been adopted by more than one hundred independent teams to actually solve a real-world problem. For example, a group of climatologists at France's UMR Espace-Dev laboratory use discords to find anomalies in climate data [14]. A team of researchers at NASA's JLP lab have applied discord discovery to planetary data, noting that "(*discords*) *detect Saturn bow shock transitions well*" [5]. There are several other time series anomaly detection algorithms that are well cited [8][4], but most of the citations are from rival methods comparing these algorithms on a handful of benchmarks [21].

4 Definitions and Notation

Our data type of interest is *time series*.

DEFINITION 1. A *time series* $T \in \mathbb{R}^C$ is a sequence of real-valued numbers $t_i \in \mathbb{R}$: $T = [t_1, t_2, \ldots, t_C]$, C is the length of T.

Typically we are not interested in global properties of a time series but rather shapes of small regions called *subsequences*:

DEFINITION 2. A *subsequence* $T_{i,m}$ is a contiguous subset of values from T starting at index i with length m.

We can take any subsequence from a time series and compute its distance to all subsequences. We call an ordered vector of such distances a *distance profile*:

DEFINITION 3. A *distance profile* D_i for time series T refers to an ordered array of distances between a given query subsequence $T_{i,m}$ and all subsequences in time series T.

As noted above, we are assuming that the distance is measured using the Euclidean distance between the z-normalized subsequences. This distance can be computed very efficiently using the MASS algorithm [16]. For a distance profile D_i of query $T_{i,m}$ the i^{th} position represents the distance between the query and itself, so the value must be 0. The values

[1] At the time of writing this paper in on arxiv.org and is not peer reviewed. However, its claims seem irrefutable, and, in any case, we have independently confirmed them.

before and after position i are also close to 0, because the corresponding subsequences have overlap with query. We need our algorithm to ignore these trivial matches of the query and itself, and instead focus on *non-self matches:*

DEFINITION 4. *Non-self match*: Given a time series T containing a subsequence $T_{p,m}$ of length m starting at position p and a matching subsequence $T_{q,m}$ starting at q, $T_{p,m}$ is a *non-self match* to $T_{q,m}$ with distance $d_{p,q}$ if $|p - q| \geq m$.

As we noted, our ideas and algorithms for combining individual anomaly scores into a multi-dimensional anomaly score are agnostic to the choice of individual anomaly scoring technique. However, for concreteness, and because there is increasing evidence that it is among the state-of-the-art, we will explicitly ground our ideas with time series discords. We can use the definition of non-self match to help define *time series discord*:

DEFINITION 5. *Time series discord*: Given a time series T, the subsequence $T_{d,m}$ of length m beginning at position d is said to be a discord of T if $T_{d,m}$ has the largest distance to its nearest non-self match. That is, $\forall$ subsequences $T_{e,m}$ of T if M_D represents all non-self matching subsequences of $T_{d,m}$, and M_E represents all non-self matching subsequence of $T_{e,m}$, $min(d_{d,M_D}) > min(d_{e,M_E})$.

Although there are many ways to locate time series discords, the most effective methods exploit a proposed data structure called the *Matrix Profile* [22]:

DEFINITION 6. A *Matrix Profile* (*MP*) of a time series T is a vector storing the z-normalized Euclidean distance between each subsequence and its nearest non-self match. Formally, $MP = [min(D_1), min(D_2), \ldots, min(D_{C-m+1})]$, where D_i $(1 \leq i \leq C - m + 1)$ is the distance profile of query $T_{i,m}$ in time series T. The highest value of the MP is the time series discord.

We generalize the Matrix Profile to *multi-dimensional time series*, which we define as:

DEFINITION 7. A *multi-dimensional time series* $T \in \mathbb{R}^{N \times C}$ is a set of co-evolving time series $T^{(i)} \in \mathbb{R}^C$: $T = [T^{(1)}, T^{(2)}, \ldots, T^{(N)}]^T$ where N is the dimension of T and C is the length of T.

Figure 4 illustrates this notation with a toy dataset. Suppose we have a three-dimensional time series.

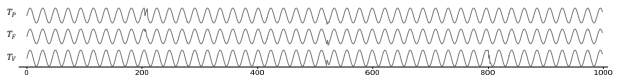

Figure 4: A toy three-dimensional time series that we will use as a running example. P = pressure, F = flowrate, V = viscosity.

We are interested in multi-dimensional time series that have anomalies that may be present on a subset of dimensions; thus, we call such anomalies a *K-dimensional anomaly*.

DEFINITION 8. A *K-dimensional-anomaly* (*KDA*) is an anomaly that is manifest on at least *K* time series.

The toy dataset in Figure 5 has three anomalies each occurring at a unique time. The anomaly marked in green is a 2-dimensional-anomaly (2*DA*) since it is present on at least two time series. The anomaly marked with blue is a 3-dimensional-anomaly (3*DA*) and finally the anomaly marked with red is a 1-dimensional-anomaly (1*DA*).

Notice that any iDA will also be a i-1DA. For example, a 3*DA* is also a 2*DA* and 1*DA*, and a 2*DA* is also 1*DA*.

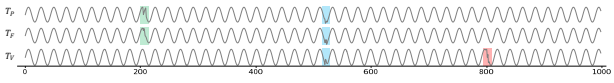

Figure 5: In this multi-dimensional time series we have three anomalies at index 205, 520 and 800 respectively. The anomaly marked by green is a 2DA, the second anomaly marked in blue is a 3DA and finally the anomaly marked by red is a 1DA anomaly.

We are ultimately interested in detecting the *natural anomalies* in a dataset:

DEFINITION 9. *Natural anomaly*: for a given timestamp t and the list of the *KDA*s at t, iDA is a natural anomaly if i is equal to anomalies' natural dimension.

The natural dimension of an anomaly is the maximum number of the time series that the anomaly is presented on. In our running example the natural dimension of the anomaly marked in green is two. The natural dimension of the anomalies marked in blue/red is three/one respectively.

Why are we interested in defining and finding the *natural anomaly*? It might be imagined that it is sufficient to simply declare that there was an anomaly at time t. However, anomaly detection is more actionable if we know *which* sensors are involved. For example, in petrochemical processing, a distillation column may be monitored by 1,000 sensors, and these may be spread over a 10,000 m^2 plant. If an anomaly is detected, the plant manager may need to dispatch a response team. Knowing which subsystem is experiencing failure could help her to quickly direct her team to the right location [19][7].

It is important to note that our natural anomaly definition makes no claim about causality or redundancy. For example, an over-pressurized boiler may produce an anomaly in a pressure^{Pa} time series, and that, in turn, may cause anomalies in both the temperature^{Fahrenheit} and temperature^{Celsius} time series[2]. We would expect the natural anomaly to discover an anomaly featuring these three traces. The task of discovering the direction of causality and the redundancy the two temperature measurements, is something for a downstream algorithm.

We will show in Section 5 how we can score *KDA*s. In Section 6 we show how we can distinguish the *natural anomalies* from other *KDA*s by comparing their scores.

We have now defined the task-at-hand, to discover *K-dimensional-anomalies* and *natural anomalies*. We propose to do this by computing a Matrix Profile for each time series, and

[2] The example may seem frivolous. However, we have seen petrochemical datasets that record temperatures in both Kelvin and Fahrenheit. Moreover, because of rounding policies the correlation between these two time series was not 1.0

then "reasoning" about combinations of these *MP*s to see which combination is most likely to contain the anomaly.

The calculated Matrix Profiles for each time series in the T can be saved in an array called *all-matrix-profiles* (*MPs*):

DEFINITION 10. *All-matrix-profiles* (MPs): Given a multi-dimensional time series $T \in \mathbb{R}^{N\times C}$, all-matrix-profile $MPs \in \mathbb{R}^{N\times C}$ is an array containing the Matrix Profiles of all N time series in T. Formally, $MPs^{(i)} \in \mathbb{R}^{C}: MPs = [MP^{(1)}, MP^{(2)}, \dots, MP^{(N)}]^T$.

Figure 6 illustrates the *MPs* for the toy dataset we are using as our running example.

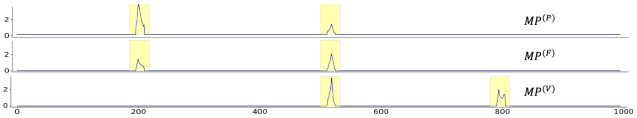

Figure 6: Matrix Profiles for the three time series in our running examples. Notice how the "bumps" reflect the anomalies in Figure 5.

In an N-dimensional time series, an anomaly may present on 1 or 2 … or K time series ($K \leq N$). Since we do not know in advance on which set of time series an anomaly is preserved, we need to (in principle) extract all the possible *anomaly-score sets*:

DEFINITION 11. *Anomaly-score set* (*S*): given an all-matrix-profiles $MPs \in \mathbb{R}^{N\times c}$ there are 2^N-1 possible combinations of the anomaly scores in MPs. Any possible combination of anomaly scores in MPs is an *anomaly-score set*.

We group all the anomaly-score sets of the same size in a list called *K-dimensional-anomaly-score*:

DEFINITION 12. *K-dimensional-anomaly-score* (*KS*): given an all-matrix-profile $MPs \in \mathbb{R}^{N\times c}$ there are $\frac{N!}{(N-K)!K!}$ possible combination of anomaly-scores set of size K. The *KS* is a list of sets, the sets that contain all anomaly-scores with size K, where $K > 0$. Formally,

$$1S = [\{MP^{(1)}\}, \dots, \{MP^{(N)}\}]$$
$$2S = [\{MP^{(1)}, MP^{(2)}\}, \{MP^{(1)}, MP^{(3)}\}, \{MP^{(2)}, MP^{(3)}\}, \dots \{MP^{(N-1)}, MP^{(N)}\}]$$
$$3S = [\{MP^{(1)}, MP^{(2)}, MP^{(3)}\}, \{MP^{(1)}, MP^{(2)}, MP^{(4)}\}, \dots \{MP^{(1)}, MP^{(N-1)}, MP^{(N)}\}]$$
$$\dots$$
$$NS = [\{MP^{(1)}, MP^{(2)}, MP^{(3)}, MP^{(4)}, \dots, MP^{(N-1)}, MP^{(N)}\}]$$

Using the toy dataset as an example, the K-dimensional-anomaly-scores are as follows:

$$1S = [\{MP^{(P)}\}, \{MP^{(F)}\}, \{MP^{(V)}\}]$$
$$2S = [\{MP^{(P)}, MP^{(F)}\}, \{MP^{(P)}, MP^{(V)}\}, \{MP^{(F)}, MP^{(V)}\}]$$
$$3S = [\{MP^{(P)}, MP^{(F)}, MP^{(V)}\}]$$

To be clear, these are all the possible subsets of anomaly-scores, upon which an anomaly could manifest itself. Thus, if we wish to discover an anomaly that is present on, say, two dimensions, it must be one of the sets listed in 2*S* above.

The reader will appreciate that the K-dimensional-anomaly-score contains all the data we need to locate anomalies that are preserved on at least K time series.

While one could imagine many ways to mine a K-dimensional-anomaly-score and locate the appropriate *KDA*, we take a direct approach. We aggregate the anomaly-score sets (S) by taking the *min* value at each timestamp to generate the *Min Matrix Profiles* (MMP):

DEFINITION 13. *Min Matrix Profile* (MMP): given an anomaly-score set $S \in \mathbb{R}^{K\times C}$, MMP is a vector storing the aggregated anomaly score of Matrix Profiles in S obtained by taking the min value at each timestamp. Formally, $MMP = \min_{j\in K} S_{i,j}$, $j = 1, \dots, K$, where $1 \leq i \leq C$. The min method acts as an AND operator, so we would get a high value if and only if *all* Matrix Profiles have a high value at the given timestamp.

The high values (i.e., peaks) in the MMPs indicate the location of *KDA*s. Note that for a given *KS*, we have $\frac{N!}{(N-K)!K!}$ MMPs. Figure 7 illustrates this concept.

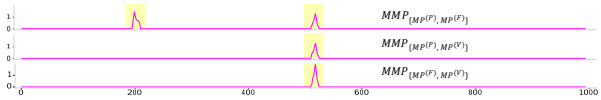

Figure 7: The toy data's *2S* has three MMPs ($\frac{3!}{(3-2)!2!} = 3$). Note that we only have peaks in the locations where anomalies are present on at least two dimensions. Moreover, $MMP_{\{MP^{(P)}, MP^{(V)}\}}$ and $MMP_{\{MP^{(F)}, MP^{(V)}\}}$ give us a partial view of *2DA*'s location.

Each MMP gives us a partial view of the *KDA*s' location. In order to find *all possible KDA*s we take max function over all MMPs at each timestamp and generate a single vector to show the location of anomalies for that *KS*. We call this vector *KD-profile*:

DEFINITION 14. *KD-profile* (*KDP*): given the list of the MMPs of a *KS*, *KDP* is a vector storing the aggregated anomaly score of the MMPs obtained by taking the max value at each timestamp. Formally,

$$KDPs = \max_{j\in K} MMPs_{i,j}, \; j = 1, \dots, K, \text{ where } 1 \leq i \leq C.$$

The max method acts somewhat like an OR operator, so we will get a high value if *any* of the MMPs has a high value in the given timestamp. Figure 8 illustrates this notation.

Figure 8: The two-dimensional-profile (2*DP*) of the 2*S* of our toy example. Notice that 2DP has peaks only where data has 2*DA* and there is no peak at index 800 where data has 1*DA*.

The calculated *KD*-profiles for all K-dimensional-anomaly-scores are saved in an array called *all-KD-profiles* (*KDPs*):

DEFINITION 15. *All-KD-profiles KDPs* $\in \mathbb{R}^{N\times C}$ is an array, containing *KD*-profiles of the K-dimensional-anomaly-scores for $K = (1, 2, \dots, N)$.

Given the above definitions, we are now in a position to formalize our two problem statements.

Problem Statement 1: Given an N-dimensional time series T, a user selected subsequence length m, and selected scoring method, find the most significant *KDA*s for $K = 1$, $K = 2$, … , $K = N$. The output is a list L of length N, where each element of the

list consists of a triple containing: the *location* (timestamp/index) of the *KDA*, the *set* indicating which time series are contributing to the *KDA*, and the last value is the *significance* score of the *KDA*.

We can refer to the three elements of the i^{th} list item with $L_{i.location}$, $L_{i.set}$, and $L_{i.significance}$.

It is important to recognize that, in general, the elements of the list do not have to be *nested*. For example, we may have $L_{1.set}$ ={pressure}, and $L_{2.set}$ ={flow-rate | viscosity}. The fact that we do not require the nesting property is important to allow full expressiveness of representation, but it unfortunately does preclude certain search mechanisms for efficiently computing *L*, including branch and bound or dynamic programming.

Depending on the downstream application, a user can explore all *N* elements in the list *L*, or they can ask for only the natural anomalies in *L*:

Problem Statement 2: Given the list *L* of *KDA*s, return the most natural anomaly.

Recall the example used in Figure 2. The true anomaly is $L_{2.set}$ ={1|2}. If we measure maximum 2*DA'* score on any *two* dimensions only in this dataset, the score of 8.33 reflects the embedded anomaly on {1|2}. Suppose we insisted on finding any anomaly on only one dimension. In this case maximum 1*DA'* score reflects time series {2} with a lower score of just 6.9. In the other direction, suppose we insisted on finding any anomaly on only three dimensions. In this case the set returned is {74|18|20}, and we know from our creation of the data that these time series are spurious. Critically, however, note that the *3DA*'s score has decreased to 0.

Thus, our definition of *KDA*'s score allows a simple and obvious direction of detecting the "*natural anomalies*"

5 Algorithms

5.1 *KDA* Detection Algorithm

In the previous section we defined what we wish to compute. Here we discuss algorithms to actually compute these definitions. For concreteness, we begin with the brute-force algorithm to compute the *KDP*s as shown in Table 1.

In line 1 we obtain *N*, the number of time series in multi-dimensional time series *T*. Note that number of time series is equivalent to the maximum possible dimensions that an anomaly can manifest itself on. We iterate through all the time series in *T* and calculate their Matrix Profile to make the all-matrix-profile array in 4 to 5. A list containing all the possible combinations for a list with size *N* is calculated in line 7. We use this list to query all the possible anomaly-score sets from the all-matrix-profile array. We save the anomaly-score sets with respect to their sizes in a dictionary in line 10 to 13. Then we iterate through each *K*-dimensional-anomaly-score to generate its *KD*-profile in lines 15 to 22. When calculating *KDP*s we also save the indices of time series contributing to each *KD*-profile.

Once we calculate the *KDP*s, the detection of *KDA*s is trivial. The peaks on each *KDP* indicate the location of a *KDA*. Figure 9 shows an example of the output of this algorithm on the toy example. Note that they are sorted from top to bottom. The top figure shows the 1*DP*, notice that it is the most "busy" reflecting the fact that it shows all anomalies that appear on even a single dimension.

Table 1: Brute-Force Algorithm to compute *KDP*s

```
Function:  Brute_Force_KDP (T, m)
Input:
    T: {Array-like} of shape (n_samples, n_time_series)
      Multi-dimensional Time series
    m: int
      Subsequence length
Output:
    KDPs: {Array-like} of shape {n_samples, n_time_series}
      All-kd-profiles
1  N = T.shape[1] // get the number of time series in T
2                 // calculating all-matrix-profile (def. 10)
3  MPs=empty_like(T)
4  For i, ts in enumerate(T):
5      MPs[:,i] = matrix_profile(ts, m)
6         // get all possible set of combination of time series
7  C = get_combinatiosn(N)
8                // get all the K-dimensional-anomaly-scores(def. 12)
9  A = defaultdict(list)
10 For this_set in C:
11     k= len(this_set)      // get the size of the set
12     S = MPs[:,this_set]   // get the anomaly-score set(def. 11)
13     A[k].append(S)
14                           // get all-KD-profiles(def. 15)
15 KDPs = []
16 For k, v in A.items():
17     MMPs = []             // calculate the MMPs (def. 10)
18     For S in v:
19        MMP = min(S, axis = 1)
20        MMPs.append(MMP)
21     KDPs.append(max(MMPs, axis = 1))
22 Return KDPs
```

The second from the top figure shows the 2*DP*, note that it is less "busy", as there are (generally) fewer anomalies that appear on (at least) two dimensions, and so on.

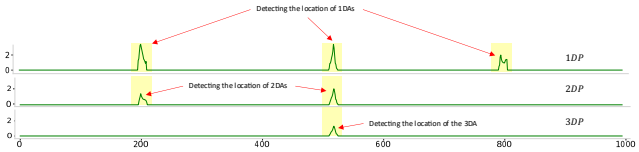

Figure 9: The all-*KD*-profile for the toy dataset. The peaks in each *KDP* represent the location of *KDA*s.

The information in the *KDP*s array can be seen as an index that can be used by downstream algorithms to interpret results and rank *KDA*s.

For example, it can be used to answer questions like "What is the most significant anomaly that appears on exactly three dimensions?". In Table 2 we will present a simple explicit algorithm to allow the user to answer such questions.

The number of the time series is extracted in line 1. Lines 3 to 7 extract the indices/timestamps of the *KDA*s that are preserved only on the *K* time series. The indices of the time series that contributed to the *KDA*s are extracted in line 8 using *KDP_idx* array. *KDP_idx* contains the indices of each time series that contributed to *KDP*s. The anomaly scores from each *KDP* are saved in array s. This information is used in lines 11 to 16 to calculate the *KDA*'s score using a user selected scoring method. Line 17 shows how the index of the most significant *KDA* is obtained. Finally, the *location* (timestamp/index) of the most significant *KDA*, the *set* contributing to it, and its significance *score* are returned.

For example, if we want to query the *KDP*s array shown in Figure 9 with "*What is the most significant anomaly that*

appears on just two dimensions?", the output would be (200, {0,1}, 4.53). Note that in these examples we envision the queries coming directly from an end-user, however, these queries may also come from a downstream algorithm.

Depending on the downstream application different scoring methods can be used. In this work we introduce three different scoring methods: min, sum, mean. In sum the scores from each *KDP* contributing to the *KDA* are summed together. We can normalize the sum score for a given anomaly by dividing it by the amount of dimensions that given anomaly has. This is the exact approach of the mean method. With the mean method we penalize a subset for their cardinality. Finally, in the min method, the score for each *KDA* is equivalent to the anomaly score on the corresponding *KDP*. For example, the min score for a 2*DA* is the value of the peak on 2*DP* where that 2*DA* was located. We call this the min method since it reads the scores from the Min Matrix Profiles building up the *KDP*.

Table 2: Algorithm to compute Query *KDP*s

```
Function:  Query_KDP (KDPs, KDP_idx, method, th)
Input:
    KDPs: {Array-like} of shape {n_samples, n_time_series}
      All-kd-profiles
    KDP_idx: {Array-like} of shape {n_samples, n_time_series}
      indexes of the time series contributing to KDPs
    method: {'min', 'sum, 'mean'}, default= 'sum'
      method to score anomalies
     k: int          Number of the dimensions of the anomaly
     th:             Threshold to detect anomaly, default =0
Output:
    KDA: tuple
      a triple (X,Y,S)containing:
      X: int                    The timestamp/index of the KDA
      Y: array of int           The name/index of time series
      S: float                  Significant of KDA
1   N = KDPs.shape[1] // get the number of the time series
2   //get the location of the KDA that are on exactly K dimensions
3   if k < N-1:
4      x = set(where(KDPs[:,k]>th)-set(where(KDPs[:,k+1]>th))
5      x = list(x)
6   else:
7      x = where(KDPs[:,k])
8   y = KDP_idx[x,:k+1] //get time series' contributing to KDA
9   s = KDP[x,:k+1] // get anomaly score from each dimension
10  // get the scores for each candidate KDA
11  if method == 'min'
12     score = s[:,k]
13  if method == 'sum'
14     score = sum(s[:,:k+1],axis=1)
15  if method == 'mean'
16     score = mean(s[:,:k+1], axis=1)
17  top = argmax(score)//get the idx of the KDA with highest score
18  most_significant_kda = (x[top], y[top], score[top])
19  return most_significant_kda
```

The reader will appreciate that the algorithm in Table 3 is only O(n) where n is the length of the timeseries, however Table 1 requires O(n×2^N) memory and O(n×2^N) time. This memory requirement is inconsequent, but the time requirement is intractable for all but the smallest time series. In the next section we will show how we can exploit sorting to dramatically improve the scalability of this algorithm.

5.2 Fast *KDP* Algorithm

In In the previous section, we used the brute force algorithm to detect *KDAs*. As we will later show, it is extremely effective at detecting K of N anomalies, but there is a problem: it is simply too slow to compute. For example, if we have twenty time series with a length of 1,000, it takes about fifteen minutes. This may be tenable, but with just fifty time series of that same length it takes several millennia [24].

However, we have a simple way to make it faster. It is based upon the following observations. We do not actually need to enumerate and score every combination of *MP*s. That approach *does* solve the task-at-hand but produces spurious information and requires many redundant calculations. Instead, we can exploit the independence of values at each time point. Consider a single time point i on the time series. We can look at the values of all the Matrix Profiles at location i and sort them into a list. Then the value of the 1*DP* at the i^{th} location is just the largest value in our sorted list, the value of the 2*DP* at the i^{th} location is the second largest value in our sorted list, and so on. Table 3 formalizes this idea.

Table 3: TSADIS: Fast KDP Algorithm

```
Function:  Fast_KDP (T, m)
Input:
    T: {Array-like} of shape (n_samples, n_time_series)
      Multi-dimensional Time series
    m: int                    Subsequence length
Output:
    KDPs: {Array-like} of shape {n_samples, n_time_series}
      kd-profiles
    KDP_idx: {Array-like} of shape {n_samples, n_time_series}
      The indexes of the time series contributing to KDPs
1  // calculating all-matrix-profile (def. 9)
2  MPs=empty_like(T)
3  For i, ts in enumerate(T):
4    mp = matrix_profile(ts, m)
5    MPs[:,i]=mp
6  // get all-kd-profiles(def. 15)
7  KDPs, KDP_idx = sort(MPs, axis= 1, order = descending)
8  return KDPs, KDP_idx
```

In lines 2 to 5 we iterate through every time series and calculate their Matrix Profile. We sort the values in the *MPs* across the y-axis in descending order. We also save the indices that sort the *KDP*s in an array of the same size as *KDP*s.

5.3 Robustness to Noise

In real-world applications, multi-dimensional time series data often contains noise (see Figure 11). When noise becomes significant, it may effect anomaly detection models. We have based our algorithm on the Matrix Profile, which is a vector recording the smallest z-normalized Euclidean distance of each subsequence of a time series to all other subsequences. Therefore, if data is noisy, all the distances will generally become higher. We can remove this added base value to all distances in Matrix Profile before calculating the *KDP*s to suppress the high anomaly score caused by the noise. We estimate this base value by calculating the 75th percentile of the distances in Matrix Profile. This is a very intuitive parameter. We could also achieve essentially the same result just by *smoothing* the time series, before computing the Matrix Profile.

5.4 Online KDAs Detection

We have demonstrated the ability to detect *KDA*s in the previous section, however, we assumed that the entire time series was available. Here, we show how to detect *KDA*s online without that assumption. When working with streaming data, we need to take the new incoming single data point and compare its subsequence with the rest of the time series, compute the distance profile for this subsequence and update the existing Matrix Profile. This can be accomplished using the algorithm introduced by [22], which maintains the Matrix Profile in an incremental fashion by taking the existing data T and calculating Matrix Profile MP. When a new data point, t, arrives it appends

t to T and compares the new subsequence with all extant subsequences and updates the historical values. Further, it determines which of the existing subsequences is the nearest neighbor to the new subsequence and appends this information to the Matrix Profile, which continues as additional data stream in. Note that incremental Matrix Profile is different from batch Matrix Profile since it does not waste time re-computing any past pairwise distances, it only computes new distances and updates the appropriate arrays where necessary, making the algorithm very fast. This algorithm can be used to keep track of the extreme values of an incrementally-growing Matrix Profile and report a new discord when there is a new maximum value. This implies the ability to extract *KDA*s in streaming data by using incremental Matrix Profile for generating *KDP*s.

6 Empirical Evaluation

We have designed all experiments such that they are easily reproducible. To this end, we have built a webpage [24] that contains all datasets, code and random number seeds used in this work. This philosophy extends to all the expository examples in the previous sections. In our comparisons to other systems, unless otherwise stated, we take numbers directly from the original papers, rather than reimplement the approaches. This is not laziness on our part. Many of these approaches are very complex, and it would be difficult to ensure that we have reimplemented them to the authors' satisfaction. This ensures that we are comparing to the best possible implementation, on data that the authors think suits their approach.

In [24] we show an additional late-breaking result, comparing to one of the most cited papers of the last few years, which introduces the NeurIPS Benchmark [11]. We show that we beat ten state-of-the-art methods on the authors own datasets. We encourage the interested reader to review this.

6.1 Preamble: Metric of Success

Several recent papers have suggested that many techniques to evaluate anomaly detection algorithms are flawed [21]. For example, Kim et. al. show that a commonly used scoring metric would highly rank an algorithm that simply randomly guesses [13]. Keogh argues that many papers report results with unwarranted and misleading precision [15]. For example, suppose in a year of data, all of Xmas day is an anomaly because the sensor was turned off for maintenance. The correct way to report successes here would be binary, either 0/1 or 1/1. However, many papers report successes based on the sampling rate, something like 1440/1440. Or the algorithm may miss one datapoint, allowing the author to report 0.9993. There is a huge difference between the intellectually honest 0/1 and 0.9993. The latter implies an incredible precision that is just not warranted. The problem is compounded by the fact that in most cases it is impossible to say *exactly* where an anomaly begins or ends.

With this background in mind, we have taken great care to design a scoring function that produces sensible output, with only appropriate precision reported, and for which a random guess would score very poorly.

- If the ground truth for the anomaly is given as a region $T_i : T_j$, we score a prediction *correct* if it is anywhere within the range $T_{i-m} : T_{j+m}$. The reason for the "bracketing" of the region is that for some algorithm's prediction P, the location of the most anomalous region, P may be reporting the *beginning*, or the *end*, or the *middle* of a subsequence. Our scoring function will not penalize for this trivial detail.
- If an algorithm predicts that a set of K time series has an anomaly at location P, we count as a success each time that a prediction was true. For example, if we predict there was an anomaly at time P on time series {2,5,9}, and we examine the ground truth to discover that only {2,9} really had anomalies at P, we report our success as 2/3.

We believe that this metric of success is fair and intuitive. However, we note that the excellent results of TSADIS below is also apparent with other common metrics of success.

6.2 Comparison to MSCRED

Perhaps the most cited recent paper on multi-dimensional time series anomaly detection in recent years is [23]. Here the authors introduce a Multi-Scale Convolutional Recurrent Encoder-Decoder (MSCRED) and demonstrate its effectiveness on the dataset shown in Figure 10.

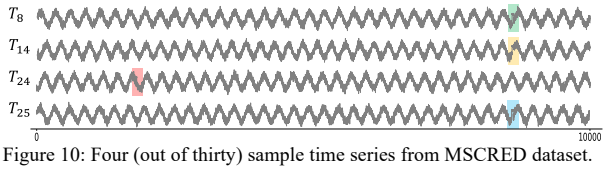

Figure 10: Four (out of thirty) sample time series from MSCRED dataset.

This dataset is designed to be challenging, with sinusoidal waves that differ in phase and frequency, and with "shock wave like" anomalies at different scales modeling three different types of root causes randomly inserted. Some of the anomalies are obvious, but some (see T_{24} in the above) are very subtle.

To evaluate TSADIS on this dataset we ask the following question of the data: *Find the Top-5 anomalies in Three Dimensions*. Table 4 summarizes the results.

Table 4: The results of TSADIS on the MSCRED dataset

ID	Ground Truth	TSADIS Prediction	TSADIS Score	Default Rate
A	11810, {24,15,28}	11744, {24,28,15}	3/3	0.012
B	12760, {21,26,5}	12715, {26,21,5}	3/3	0.012
C	14540, {3,16,2}	14418, {16,2,3}	3/3	0.012
D	17790, {9,5,20}	17682, {12,20,5}	2/3	0.012
E	18620, {25,14,8}	18546, {8,25,14}	3/3	0.012

These results are almost perfect, with a single error made on the case "D". The original authors use a different metric of success, average recall over five runs, obtaining 0.8. If we use *that* metric of success, we score 0.93.

Beyond the significant improvement in effectiveness, there are several other reasons to prefer TSADIS over MSCRED. The MSCRED approach required significant training data to achieve its results. In contrast, we can use TSADIS with *zero* training data. If we do so, our performance drops a little to 0.86, but we still beat MSCRED.

MSCRED is a very complicated approach, featuring a fully convolutional encoder, an attention based ConvLSTM, a temporal attention mechanism, a mini-batch stochastic gradient descent and several other elements. It is difficult to be sure how many parameters must be tuned here, but it seems to be at least

twelve. Moreover, because the output of the algorithm is stochastic, it takes significant effort to understand the effects of the parameters on the performance. In contrast, TSADIS only requires the setting of a single parameter, *m*. Here we set *m* to 250, which was, by visual inspection, a "whole" number approximately equal to the average period length. However, for examples B and C, we could have set *m* to any value between 150 and 650, a huge range, and still have obtained perfect results. The other examples have a slightly smaller range, but are still robust to *m*. In fairness, this robustness to the choice of *m* is a property of the Matrix Profile [22][17] that we inherit.

6.3 Comparison to Isolation Forest|AE-LSTM|Prophet

Another recent paper considers a real-world multi-dimensional time series dataset of photovoltaic (PV) systems [12], and compares three state-of-the-art approaches: AutoEncoder Long Short-Term Memory (AE-LSTM), Facebook-Prophet, and Isolation Forest. Eexamples of the data are shown in Figure 11.

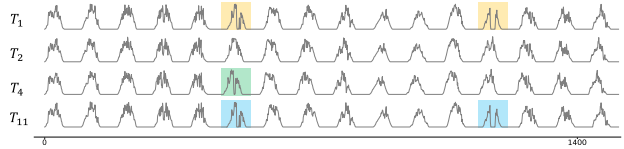

Figure 11: Four sample time series (out of twenty-two) selected from the photovoltaic (PV) systems dataset. Sixteen days are used.

The authors of this study actually solve an easier problem than the more general task we consider. They essentially ask: For a given *single*-dimensional time series, assuming we know it has an anomaly, can we detect where it is? Of the three approaches compared, only AE-LSTM obtained a perfect score. In contrast, we ask the more difficult multi-dimensional questions. *Find the Top-1 anomalies in K dimensions*. There is an objective ground truth provided only for the cases *K* is 1, 2, and 6, Table 5 summarizes the results.

Table 5: The results of TSADIS on the photovoltaic dataset

ID	Ground Truth	TSADIS Prediction	TSADIS Score	Default Rate
A	465, {0,4,11,16,18,19}	470, {0,4,11,16,18,19}	6/6	0.054
B	1140, {0,11}	1171, {0,11}	2/2	0.054
C	1320,{20}	1333,{20}	1/1	0.054

Here we achieved perfect results, again completely ignoring the training data that the three other methods relied upon. In this dataset dawn to dusk is about 60 datapoints, so we set $m = 60$. However, if we had set *m* to any value in the range 50 to 130, we would still have gotten perfect results.

6.4 Comparison to MGAB

In a recent work [20], the authors note some frustration with the community confining their interest to datasets containing anomalies that are readily apparent to the naked eye. This observation is an independent confirmation of the "triviality" argument of claims of Wu and Keogh [21]. However, the authors of [20], explicitly take action to redress the problem, by creating a dataset where the "*anomalies are for the human eye very hard to distinguish from the normal (chaotic) behavior*". Figure 12 suggests that they were successful.

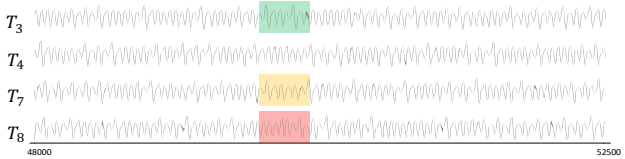

Figure 12: Four sample time series (out of ten) selected from the MGAB dataset. Only 4,500 datapoints (out of 97,600) are shown.

As with the previous example, the authors consider an easier problem than the more general task we consider, asking, for a given *single*-dimensional time series, can we detect where it is? We again will ask the harder multi-dimensional questions. *Find the Top-1 anomalies in K dimensions*. We do this for $K = 4$ and $K = 5$. Table 6 summarizes the results.

Table 6: The results of TSADIS on the MGAB dataset

ID	Ground Truth	TSADIS Prediction	TSADIS Score	Default Rate
A	84614, {0,1,2,5,6}	84697, {2,5,1,0,6}	5/5	0.082
B	49765, {2,3,7,8}	49802, {8,7,3,2}	4/4	0.082

Here we achieved perfect results. In the original paper, the authors compare to five anomaly detection methods (DNN-AE, LSTM-ED, NuPIC, LSTM-AD, TCN-AE [20]). As explained above, results are not exactly commensurate, however it is interesting to note that all five methods used training data, yet none was able to obtain perfect results. In our experiments we do include the training data (or rather, we make no effort to exclude it), but we in no way delineated it or labeled it as training data. Yet we were able to obtain perfect results.

6.5 Selecting the right *K*

In the experiments above, we evaluated our algorithm's accuracy, *given* that the user requested the correct value for *K*. Here we will evaluate our algorithms to predict the correct *K*.

As shown in Figure 13 we created a dataset that comprises of sine waves. We can add a simple anomaly to a time series, by taking the absolute value of a single period, and we can add noise to the time, which we measure in terms of a standard deviation of the original time series.

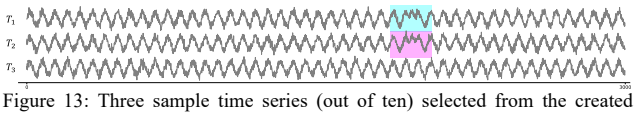

Figure 13: Three sample time series (out of ten) selected from the created dataset. Only 3,000 datapoints (out of 10,000) are shown. These examples are at noise level 0.3 × std.

We performed the following experiment. We created ten different ten-dimensional datasets, varying the number of dimensions that contained an anomaly from one to ten. We then tested to see if we could recover the correct number of anomalies, by plotting the *KDA*'s score (using the sum function) for all possible values of *K* from 0 to 10.

In order to count our prediction as correct, we insist that our algorithm must correctly predict:

- How *many* anomalies there are (what is true *K*?)
- On *which* of the ten time series there is an anomaly.
- The *location* of the anomaly.

Figure 14 shows the predicted number of anomalies for each

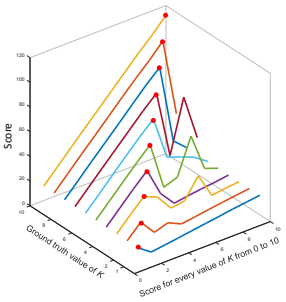

Figure 14: Our predicted number of anomalies (red dots) for datasets with a known ground truth number of anomalies that ranged from 1 to 10.

case, when we have a noise level of 0.1×std.

Given our success here, we can now ask how well we would do in the face of increasing noise levels. We repeated the experiment above, just for the case where the ground truth value of K was five. We did this ten times, and we successfully predicted $K = 5$ in each of the ten runs.

As shown in Figure 15, we repeated the entire process for increasing amounts of noise, until we had at least one failure. As it happens, for noise level 1.5×std we had two failures out of the ten runs.

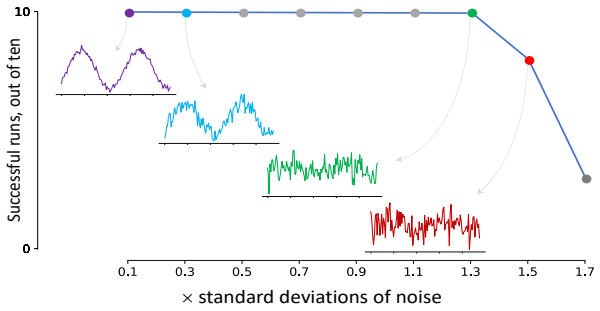

Figure 15: Testing our algorithms ability to predict the true number of time series that are involved in a five-dimensional anomaly in the ten-dimensional datasets, for increasing levels of noise.

In Figure 15.*inset* we show examples of the noise level at each setting. The reader will appreciate that the amount of noise our algorithm can tolerate before failure (shown in green), is considerable. To be fair, we are inheriting this robustness from the Matrix Profile, which is robust to noise [22].

6.6 Timing results

We have the luxury of being able to tersely summarize the time overhead for our algorithm. The time taken is dominated by the time needed to compute the Matrix Profiles. Thus, the only question is, how much *overhead* does our K of N approach incur? The answer, for all experiments in this paper, is less than 5% (Concretely, for MSCRED it is 0.17%, for MGAB 0.03%, and for photovoltaic is 3.7%).

7 Conclusions

We have shown that at least one state-of-the-art anomaly detection algorithm, the Matrix Profile, will generally fail if it is forced to consider all N dimensions of a N-dimensional time series. By casting the problem as a K of N anomaly detection we both improve the accuracy of anomaly detection and attribute the anomaly to the correct set of responsible time series. We compare the results of our algorithm on three datasets/approaches, showing, in each case, that we could match or improve upon the original author's approach despite framing the problem in a way that makes it more difficult for ourselves, for example, by completely ignoring the training data. We have made all code/data available to allow the community to confirm, exploit and expand our findings.

References

[1] F.X. Aubet, D. Zügner, and J. Gasthaus, "Monte Carlo EM for Deep Time Series Anomaly Detection," arXiv, Dec. 29, 2021.
[2] J. Audibert, S. Marti, F. Guyard, and M. A. Zuluaga, "From Univariate to Multivariate Time Series Anomaly Detection with Non-Local Information," in AALTD, 2021, pp. 186–194.
[3] P. Boniol, et. al, "Unsupervised and scalable subsequence anomaly detection in large data series," VLDB J., vol. 30, pp. 909–31, Nov. 2021.
[4] L. Bontemps, V.L.Cao, J. McDermott, and N.-A. Le-Khac, "Collective Anomaly Detection Based on Long Short-Term Memory Recurrent Neural Networks," in FDSE, 2016, pp. 141-152.
[5] A. Daigavane, et.al, "Unsupervised detection of Saturn magnetic field boundary crossings from plasma spectrometer data," Comput. Geosci., vol. 161, p. 105040, Apr. 2022.
[6] K. Doshi, S. Abudalou, and Y. Yilmaz, "TiSAT: Time Series Anomaly Transformer," arXiv , Mar. 10, 2022.
[7] P. Goel, A. Datta, and M. S. Mannan, "Industrial alarm systems: Challenges and opportunities," J. Loss Prev. Process Indust., vol. 50, pp. 23–36, Nov. 2017.
[8] Y. Guo, W. Liao, Q. Wang, L. Yu, T. Ji, and P. Li, "Multidimensional Time Series Anomaly Detection: A GRU-based Gaussian Mixture Variational Autoencoder Approach," vol. 95, pp. 97–112, Nov 2018.
[9] Y.Y. Haimes, Modeling and Managing Interdependent Complex Systems of Systems. John Wiley & Sons, 2018.
[10] K. Hundman, et. al, "Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding," in SIGKDD 2018, pp. 387–95.
[11] K. Lai, D. Zha, J.Xu, Y. Zhao, G. Wang, X. Hu: Revisiting Time Series Outlier Detection: Definitions and Benchmarks. NeurIPS Datasets and Benchmarks 2021
[12] M. Ibrahim, A. Alsheikh, F. M. Awaysheh, and M. D. Alshehri, "Machine Learning Schemes for Anomaly Detection in Solar Power Plants," Energies, vol. 15, no. 3, p. 1082, Feb. 2022.
[13] E. Keogh (2022). Irrational Exuberance Why we should not believe 95% of papers on Time Series Anomaly Detection. SIGKDD workshop talk. https://www.youtube.com/watch?v=Vg1p3DouX8w
[14] El Khansa, Gervet, and Brouillet, "Promient Discord Discovery with Matrix Profile: Application to Climate Data Insights," . Int. Conf. Ind. Instrum. Control, 2022.
[15] S. Kim, K. Choi, H.-S. Choi, B. Lee, and S. Yoon, "Towards a Rigorous Evaluation of Time-series Anomaly Detection," arXiv, Sep. 11, 2021.
[16] A. Mueen et al., "The fastest similarity search algorithm for time series subsequences under Euclidean distance," 2015. https://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html
[17] 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 ICDM, Nov. 2020, pp. 1190–95.
[18] J. Y. Park, E. Wilson, A. Parker, and Z. Nagy, "The good, the bad, and the ugly: Data-driven load profile discord identification in a large building portfolio," Energy Build., vol. 215, p. 109892, May 2020.
[19] N. Sanchez-Pi, L. A. P. Leme, and A. C. B. Garcia, "Intelligent agents for alarm management in petroleum ambient," J. Intell. Fuzzy Syst., vol. 28, no. 1, pp. 43–53, 2015
[20] M. Thill, W. Konen, and T. Bäck, "Time Series Encodings with Temporal Convolutional Networks," in Bioinspired Optimization Methods and Their Applications, 2020, pp. 161–173.
[21] R. Wu and E. Keogh,"Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress," IEEE Trans. Knowl. Data Eng., pp. 1–1, 2021.
[22] C. Yeh et al., "Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets," in 2016 IEEE ICDM, Dec. 2016, pp. 1317–1322.
[23] C. Zhang et al., "A Deep Neural Network for Unsupervised Anomaly Detection and Diagnosis in Multivariate Time Series Data," AAAI, vol. 33, no. 01, pp. 1409–1416, Jul. 2019.
[24] TSADIS webpage: https://sites.google.com/view/tsadis