Noname manuscript No.
(will be inserted by the editor)

Unsupervised and Scalable Subsequence Anomaly Detection in Large Data Series

Paul Boniol · Michele Linardi · Federico Roncallo · Themis Palpanas · Mohammed Meftah · Emmanuel Remy

Received: date / Accepted: date

Abstract Subsequence anomaly (or outlier) detection in long sequences is an important problem with applications in a wide range of domains. However, the approaches that have been proposed so far in the literature have severe limitations: they either require prior domain knowledge, or become cumbersome and expensive to use in situations with recurrent anomalies of the same type. In this work, we address these problems, and propose NormA, a novel approach, suitable for domain-agnostic anomaly detection. NormA is based on a new data series primitive, which permits to detect anomalies based on their (dis)similarity to a model that represents normal behavior. The experimental results on several real datasets demonstrate that the proposed approach correctly identifies all single and recurrent anomalies of various types, with no prior knowledge of the characteristics of these anomalies (except for their length). Moreover, it outperforms by a large margin the current state-of-the art algorithms in terms of accuracy, while being orders of magnitude faster.

Keywords Data series · Time series · Anomalies discovery

1 Introduction

Massive collections of data series[1] are becoming a reality in virtually every scientific and social domain, and there is an increasingly pressing need by relevant applications for developing techniques that can efficiently analyze them [42,8,44].

[Anomaly Detection in Sequences] Anomaly, or outlier detection is an old problem [9,55,64,30,16,65], finding applications in a wide range of domains. In the specific context of sequences, which is the focus of this paper, we are interested in identifying anomalous subsequences, that is, the outlier is not a single value, but rather a sequence of values. This distinction is crucial for the following reason: even though all individual values in a subsequence look normal when examined independently from one another, the sequence of these same values may be anomalous (e.g., the trend, or shape of the subsequence may not be normal).

Therefore, subsequence anomaly detection is a very useful and important operation for many real-world applications, because it enables the early identification of problems that would otherwise remain undetected until too late [7].

[Limitations of Previous Approaches] Existing techniques either explicitly look for a set of pre-determined types of anomalies [25,4], or identify as anomalies the subsequences with the largest distances to their nearest neighbors (termed *discords*) [64,52]. We observe that these approaches pose limitations to the subsequence anomaly identification task, for several reasons, explained below.

First, the anomalous behavior is not always known. Therefore, techniques that use specific domain knowledge for mining anomalies (e.g., in cardiology [25], and engineering [7]) involve several finely-tuned parameters, and do not generalize to new cases and domains. For ex-

P. Boniol · M. Meftah · E. Remy
EDF R&D
E-mail: firstname.lastname@edf.fr

M. Linardi · F. Roncallo · T. Palpanas
Université de Paris
E-mail: firstname.lastname@parisdescartes.fr

[1] If the dimension that imposes the ordering of the sequence is time then we talk about *time series*. In the rest of this paper, we will use the terms *sequence*, *data series*, and *time series* interchangeably.

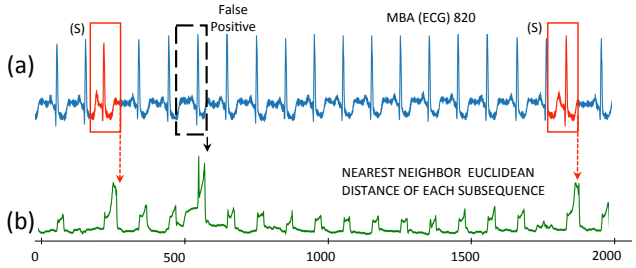


Fig. 1 (a) MBA ECG (*2000* points snippet from patient 820), with two anomalous Supraventricular premature beats (S). (b) Euclidean distances of each subsequence (length *75*) to its best non-trivial match in the full sequence: anomalies do *not* have the largest distance to their nearest neighbors.

ample, early detection of anomalies in bearings (rolling elements in rotating machines, such as an aircraft engine) is of great importance for engine manufacturers, such as Safran[2]. Even though existing techniques based on signal processing achieve good performance [4], Safran engineers have noted that these techniques require expertise and knowledge of the specific system's kinematic model, and would instead like to have an automated method, capable of detecting anomalies without expert knowledge [51].

Second, in the case of general, domain-agnostic techniques for subsequence anomaly detection, the state-of-the-art algorithms (e.g., [64,52]) have been developed for the case of a *single* anomaly in the dataset, or multiple different (from one another) anomalies. The reason is that these algorithms are based on the distance of a subsequence to its Nearest-Neighbor (NN) in the dataset: the subsequence that has the farthest NN is marked as an anomaly.

Figure 1 depicts this situation. We show a snippet of the MIT-BIH Supraventricular Arrhythmia Database (MBA) ECG recording [23,40] of patient *820*. This sequence includes *repeated* anomalous subsequences (*ventricular premature contractions*, marked by solid red rectangles). Following the state-of-the-art approaches [64, 52], we plot in Figure 1(b) the distance of each subsequence (of length *75*) to its NN, and we observe that the (known) anomalies do not correspond to the most distant NN (i.e., the highest peak in Figure 1(b)). This is because our dataset includes several anomalies that are similar to one another (i.e., of the same type). At the same time, these approaches mark as outliers subsequences that are normal (dotted black rectangle), resulting in (a large number of) false positives.

Third, in order to remedy this situation, the m^{th} *discord* approach has been proposed [61]. This approach takes into account the multiplicity, m, of the anomalous subsequences that are similar to one another, and marks as anomalies all the subsequences in the same group, by computing the m^{th} (instead of the 1^{st}) NNs for each subsequence. Nevertheless, this approach assumes that we know the multiplicity m, which is not true in practice (otherwise, we need to re-execute the algorithms for several different m values).

Fourth, another drawback of unsupervised methods for subsequence anomaly detection is the non-stationarity of data series: the data characteristics (e.g., basic statistics and trends) may change over time. These situations are hard to handle and confuse the *discord* and m^{th}-*discord* methods, since an anomalous subsequence may find a very near neighbor among the subsequences of a latter part of the series that involves a different set of normal (and anomalous) patterns.

[Proposed Approach] In this work, we address the aforementioned problems, and propose NormA, a novel approach suitable for subsequence anomaly detection. The proposed approach allows us to detect anomalies based on their (dis)similarity to a model that represents the normal (expected) behavior.

NormA starts by carefully selecting some of the subsequences of the dataset, based on a scoring mechanism. The selected set of subsequences are then used to build the normal behavior model, which is a set of sequences. This process is automatic (using the minimum description length principle), without the need for user intervention, and is effective even when the dataset contains multiple anomalies. We also propose a variant of NormA that is able to handle situations, where a single series exhibits multiple normal behaviors. This is an important case in practice, e.g., when the underlying data generation process changes among several normal states. At the end, NormA detects subsequence anomalies by comparing candidate subsequences to this normal behavior model. We note that NormA is unsupervised, and computes the normal behavior model based on the original (unlabeled) dataset, despite the presence of anomalies in it.

Using a large variety of real and synthetic datasets, we experimentally demonstrate that NormA is statistically significantly better than current state-of-the-art algorithms in detection accuracy, for both single and repeated anomalies. At the same time, NormA is one order of magnitude faster than the competition.

[Contributions] Our contributions can be summarized as follows[3].

- We summarize the state-of-the-art methods on subsequence anomaly detection, and discuss their practical shortcomings. To overcome these problems, we

[2] http://www.safran-group.com/

[3] A preliminary version of this paper, as well as a corresponding demo paper have appeared elsewhere [10,11].

propose a new definition of subsequence anomalies, based on the distance to normal behavior.

• We formalize the concept of *Normal Model*, which is a set of data series that represents the recurrent (normal) behavior in a sequence. The *Normal Model* can be the basis for anomaly detection, and can be instantiated in different ways.

• We describe a new subsequence anomaly detection algorithm that automatically constructs the *Normal Model* series, based on the principles of *frequency*, *coverage* and *centrality*. Subsequently, the algorithm uses the *Normal Model* in order to identify anomalies in an unsupervised and domain-agnostic manner. We propose two flavors of this algorithm: NormA-SJ that is based on full computation, and NormA-smpl, based on sampling, that achieves almost the same accuracy, but is considerably faster.

• Furthermore, we propose NormA-mn, an extension of our approach, that is able to effectively handle cases where a single series exhibits multiple normal behaviors.

• Finally, we conduct an extensive evaluation with the largest set of real datasets tested in the literature (including all datasets that have been used in the past), as well as several synthetic datasets. The results demonstrate that NormA is significantly more accurate than the state-of-the-art approaches proposed in the data series and multi-dimensional outliers literature, including a supervised method, even in the presence of many (similar and/or diverse) anomalies. At the same time, NormA is up to orders of magnitude faster.

[Paper Structure] The rest of this paper is organized as follows. We discuss the background and relevant challenges in Section 2. Section 3 formulates the problem. In Section 4, we describe our solution, and we report the results of our experimental analysis in Section 5. In Section 6, we discuss related work, and we conclude in Section 7.

2 Preliminaries

A data series $T \in \mathbb{R}^n$ is a sequence of real-valued numbers $t_i \in \mathbb{R}$ $[t_1, t_2, ..., t_n]$; $|T| = n$ is the length (or size) of T. We are typically interested in local regions of the data series, namely subsequences.

A subsequence $T_{i,\ell} \in \mathbb{R}^\ell$ of a data series T is a subset of contiguous values from T of length ℓ (usually $\ell \ll n$) starting at position i; formally, $T_{i,\ell} = [t_i, t_{i+1}, ..., t_{i+\ell-1}]$.

The problem we are addressing in this work is the identification of anomalous subsequences (of a given length) within a long data sequence.

Given two sequences, A and B, of the same length, ℓ, we can calculate their Z-normalized Euclidean distance, $dist$, as follows [18,41,56,58,63]: $dist(A,B) = \sqrt{\sum_1^l (\frac{A_{i,1}-\mu_A}{\sigma_A} - \frac{B_{i,1}-\mu_B}{\sigma_B})^2}$, where μ and σ represent the mean and standard deviation of the sequences. For the rest of this paper, we will simply use the term *distance*.

Given a subsequence $T_{i,\ell}$, we say that its m^{th} Nearest Neighbor (m^{th} NN) is $T_{j,\ell}$, if $T_{j,\ell}$ has the m^{th} shortest distance to $T_{i,\ell}$, among all the subsequences of length ℓ in T, excluding trivial matches [66]; a trivial match of $T_{i,\ell}$ is a subsequence $T_{a,\ell}$, where $|i-a| < \ell/2$ (i.e., the two subsequences overlap by more than half their length).

2.1 Data Series Discord

The state-of-the-art solutions for subsequence anomaly detection use the following definition for the anomalies, also called *discords*:

Definition 1 (discord [64, 52, 27, 37, 21, 17, 38, 35]) Among all subsequences of length ℓ of series T, the subsequence $T_{i,\ell}$ that has the largest distance to its NN is called a (data series) discord.

This is an intuitive definition: a subsequence is a discord if its NN is very far away. Figure 2(a) depicts the discord $T_{i,\ell}$ and the distance, d_i, to its NN (note that for ease of exposition, we represent each subsequence as a point in 2-dimensional space). Observe that distance d_i is the largest NN distance among all other subsequences. However, this definition fails when we have two neighboring discords, with a small distance to each other, and a very large distance to all the rest of the subsequences. In order to capture these situations, the m^{th}-discord has been proposed:

Definition 2 (m^{th}-discord [61]) Among all subsequences of length ℓ of series T, the subsequence $T_{i,\ell}$ that has the largest distance to its m^{th} NN is called an m^{th}-discord.

Naturally, in anomaly detection we are not only interested in the most significant anomaly. We now propose a definition that extends the previous two for the case of the k most significant anomalies:

Definition 3 (Top-k m^{th}-discord) A subsequence $T_{i,\ell}$ is a *Top-k* m^{th}-discord if it has the k^{th} largest distance to its m^{th} NN, among all subsequences of length ℓ of T.

Note that this definition subsumes the previous two: the simple discord (Definition 1) is equivalent to Top-1 1^{st}-discord, and the m^{th}-discord (Definition 2) is equivalent to Top-1 m^{th}-discord.

Example 1 Figures 2(a) and (b) illustrate these notions. This example depicts the Top-1 1^{st}-discord ($T_{i,\ell}$ in Figure 2(a)): its 1-NN is the furthest away than the NNs of all other subsequences. Figure 2(a) also shows two groups with 3 and 5 anomalous subsequences (containing $T_{j,\ell}$ and $T_{k,\ell}$). These two groups cannot be detected using Top-1 1^{st}-discord. However, using the m^th-discord definition, these groups can be detected. For instance in Figures 2(b), Top-1 3^{rd}-discord distance $d_{k.3}$ of $T_{k,\ell}$ is large enough to identify it (and its corresponding group) as anomalous. Similarly, for the group of 5 anomalies, we need to use the Top-1 5^{rd}-discord definition in order to correctly identify subsequence $T_{j,\ell}$ and the other subsequences in the same group as anomalies.

Even though discords have been extensively studied and used in the literature, they have shortcomings that can limit their practical use. Therefore, we argue for the need of a new, different approach on subsequence anomaly detection. We elaborate on these issues in the following sections.

2.2 Shortcomings of Discords

Subsequence anomaly detection based on discords has attracted lots of attention in the past years. There exist several studies that have proposed fast and scalable discord discovery algorithms in various settings [52,27,37, 21,64,17,61,38], including simple and m^{th}-discords[4], in-memory and disk-aware techniques, exact and approximate algorithms.

Nevertheless, we claim that the way discords are defined may in some situations complicate the discovery of anomalies. The reason is twofold: (i) the number of anomalies present in a dataset is usually more than one, and is not known in advance; and (ii) often times anomalous subsequences repeat themselves (approximately the same) in the same dataset.

Example 2 Assume the dataset depicted in Figures 2(a) and (b). Running the algorithm with $m = 1$ will only identify the Top-1 1^{st}-discord (that is, $T_{i,\ell}$). In fact, anomalies colored in light red in Figure 2(a) are not identified with $m = 1$. In order to identify the Top-1 3^{nd}-discord (T_k, ℓ), we will need to rerun the algorithm with $m = 3$. Figure 2(b) represents this case ($m = 3$): the group in the bottom of the plot is identified as anomalous. However, the anomalous group in the middle of the plot remains undetected (depicted in light red in Figure 2(b)). We will need to execute the algorithm up to $m = 5$ in order to identify correctly all the anomalies colored in red in Figure 2. Since we do not know when to stop increasing the parameter m, we may end up in a situation where the algorithm starts reporting false positives, i.e., erroneously identifying normal subsequences as anomalies (this happens for $m = 13$ in our example).

[4] The authors of these papers define the problem as k^{th}-discord discovery.

3 Problem Formulation

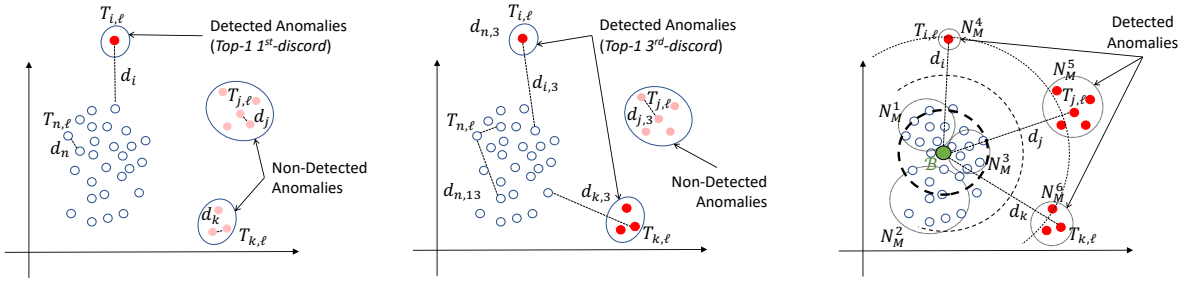
We now formulate a new approach for subsequence anomaly detection, based on the notion of normal (expected) behavior. Since we are interested in subsequence anomalies, we first define the set of all subsequences of length ℓ in a given data series T: $\mathbb{T}_\ell = \{T_{i,\ell} | \forall i.0 \leq i \leq |T|-\ell+1\}$. In general, we assume that $\mathbb{T}_\ell$ contains both normal and anomalous subsequences. We then need a way to characterize normal behavior:

Definition 4 (Normal Model, N_M) Given a data series T, N_M is a model that represents the normal (i.e., not anomalous) trends and patterns of T.

The above definition is not precise on purpose: it allows several interpretations, which can lead to different kinds of models. Nevertheless, subsequence anomalies can then be defined in a uniform way: anomalies are the subsequences that have the largest distances to the expected, normal behavior, N_M (or their distance is above a set threshold).

There are several ways to create N_M. In this work, we propose a formalization for N_M as follows: N_M is a set of sequences, $N_M = \{(N_M^0, w^0), (N_M^1, w^1), ..., (N_M^n, w^n)\}$, where N_M^i is a subsequence of length ℓ_{N_M} (the same for all N_M^i) that corresponds to a recurring behavior in the data series T, and w^i is its normality score (as we explain later, the highest this score is, the more usual the behavior represented by N_M^i is). In other words, this model averages (with proper weights) the different recurrent behaviors observed in the data, such that all the normal behaviors of the data series will be represented in the normal model, while unusual behaviors will not (or will have a very low weight).

Figure 2(c) is an illustration of a Normal Model. As depicted, the Normal Model N_M is a weighted combination of a set of subsequences (points within the dotted circles). The combination of these subsequences and their related weights returns distances d_i, d_j, d_k that are high enough to be differentiated from the normal points/subsequences. These distances can be seen



(a) d defined as the *discord* distance (*Top-1 1st-discord*)

(b) d defined as the m^{th}-*discord* distance (*Top-1 3rd-discord* for $d_{[i,j,k,n],3}$ and *Top-1 13rd-discord* for $d_{n,13}$)

(c) d defined as the distance to the Normal Model $N_M = \{(N_M^1, w^1), \ldots, (N_M^6, w^6)\}$; $\mathcal{B}$ is the weighted barycenter of N_M

Fig. 2 Illustration of the different subsequence anomaly definitions: (a) discord; (b) m^{th}-discord; (c) NormA.

as the distance between subsequences and a weighted barycenter $\mathcal{B}$ (in green) that represents N_M. Note that we do not actually compute this barycenter; we illustrate it in Figure 2(c) for visualization purposes.

We choose $\ell_{N_M} > \ell$ in order to make sure that we do not miss useful subsequences, i.e., subsequences with a large overlap with an anomalous subsequence. For instance, for a given subsequence of length ℓ, a normal model of length $\ell_{N_M} = 2\ell$ will also contain the subsequences overlapping with the first and last half of the anomalous subsequence.

In the experimental section, we demonstrate the effectiveness of the above formalization of N_M, using all datasets that have been used in the literature for subsequence anomaly discovery.

Definition 5 (Subsequence Anomaly) Assume a data series T, the set $\mathbb{T}_\ell$ of all its subsequences of length ℓ, and the Normal Model N_M of T. Then, the subsequence $T_{j,\ell} \in \mathbb{T}_\ell$ with *anomaly score*, i.e., distance to N_M, $d_j = \sum_{N_M^i} w^i * min_{x \in [0, \ell_{N_M} - \ell]} \{dist(T_{j,\ell}, N^i_{M_{x,\ell}})\}$ is an anomaly if d is in the $Top\text{-}k$ largest distances among all subsequences in $\mathbb{T}_\ell$, or $d > \epsilon$, where $\epsilon \in \mathbb{R}_{>0}$ is a threshold.

Note that the only essential input parameter is the length ℓ of the anomaly (which is also one of the inputs in all relevant algorithms in the literature [52,27,37,21, 64,17,61,38]). The parameter k (or ϵ) is not essential, as long as the algorithm can *rank* the anomalies.

We stress that in practice, experts start by examining the most anomalous pattern, and then move down in the ranked list, since there is (in general) no rigid threshold separating anomalous from non-anomalous behavior [9]. All anomaly discovery processes function this way.

As we mentioned above and will detail later on, we choose to define N_M as a set of sequences that summarizes normality in T, by representing the average behavior of a set of normal sequences. Intuitively, N_M is the set of data series, which tries to minimize the sum of Z-normalized Euclidean distances between itself and some of the subsequences in T. The Normal Model and subsequence anomaly definition is illustrated in Figure 3. Last but not least, we need to compute N_M in an unsupervised way, i.e., without having normal/abnormal labels for the subsequences in $\mathbb{T}_\ell$.

Symbol	Description
T	a data series
$\|T\|$	cardinality of T
ℓ	subsequence length
$\mathbb{T}_\ell$	set of all subsequences of length ℓ in T
N_M	Normal Model of T
N_M^i	the i^{th} sequence of Normal Model of T
w^i	Normality score of N_M^i
ℓ_{N_M}	length of Normal Model N_M
$N_M^i \bowtie_\ell T$	join between N_M^i and T with subsequence length ℓ
$T \bowtie_\ell T$	self-join of T with subsequence length ℓ_{N_M}
$\mathbb{S}$	a subset of subsequences of T, of length ℓ_{N_M}
$\mathbb{C}$	a set of clusters of subsequences of length ℓ_{N_M}
c	one cluster in $\mathbb{C}$
$Center(c)$	the centroid of cluster c

Table 1 Table of symbols

Observe that this definition of N_M implies the following challenge: even though N_M summarizes the normal behavior only, it needs to be computed based on T, which may include (several) anomalies. We address these challenges by taking advantage of the fact that anomalies are a minority class.

We can now define the problem we want to solve.

Problem 1 (Subsequence Anomaly Detection) Given a data series T, and the set $\mathbb{T}_\ell$ of all its subsequences of length ℓ, define a function $f : \mathbb{T}_\ell, k \rightarrow \mathcal{A}$ that returns $\mathcal{A}$, a set containing the k most important subsequence anomalies in $\mathbb{T}_\ell$.

In this work, we focus on the $Top\text{-}k$ anomalies; using instead a threshold ϵ to detect anomalies is a straightforward extension.

Table 1 summarizes the symbols we use in this paper.

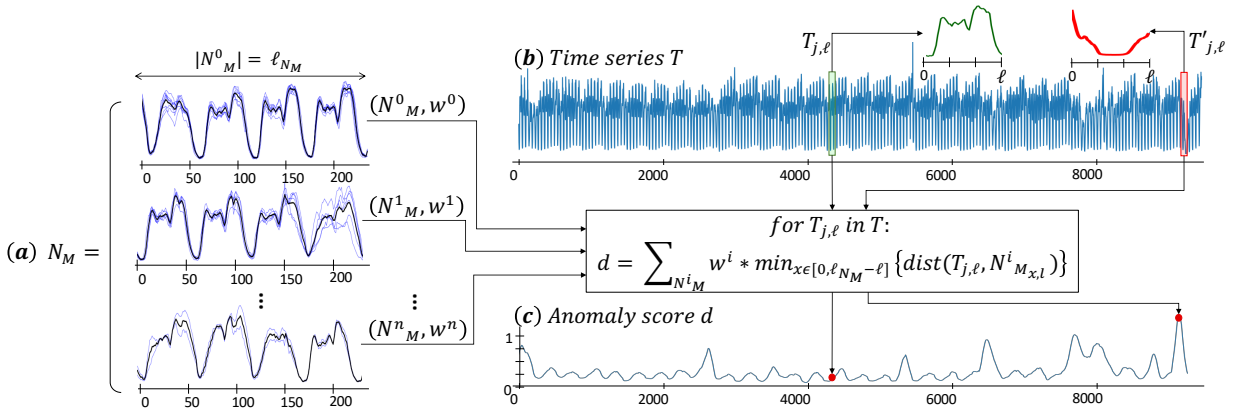


Fig. 3 (a) Normal Model of series T (shown in (b)), composed of n cluster centroids (thick lines) of subsequences (thin lines) of T. Each subsequence of T is compared to all centroids N^i_M weighted by w^i (black box). (c) Anomaly score d of all subsequences of T: subsequence $T_{j,\ell}$ is normal (low score), while $T'_{j,\ell}$ is an anomaly (high score).

Algorithm 1: NormA Subsequence Anomaly Detection.

input : data series T, anomaly length ℓ
output: *Anomalies* - list of anomalous subsequences

1 $N_M \leftarrow CompNM(T, \ell_{N_M})$; `// compute Normal Model`
2 $Anomalies \leftarrow CompAnom(T, N_M, \ell)$;`// detect anomaly`

4 Proposed Approach

In this section, we describe NormA, our solution for automated subsequence anomaly detection.

Algorithm 1 summarizes our approach, which detects anomalies based on their distance from the (set of) *Normal Model* (sequences). It takes as input a data series T, and the length ℓ of the candidate anomalies. The algorithm first computes the Normal Model N_M based on T, and subsequently detects and returns a ranked list of the anomalous subsequences in T based on N_M.

We note that the length of the anomalies, ℓ, is the *only* user-defined parameter in the subsequence anomaly detection techniques we propose in this work, and can be set by the domain expert (e.g., in the case of electrocardiogram data, cardiologists are interested in analyzing heartbeats, which have a known length). This parameter appears in all subsequence anomaly detection methods [61,64,52,28] as well as in outlier techniques [14,36]. All the other parameters described in the the rest of this section are internal parameters, and are set automatically to their default values. For instance, the length of the Normal Model sequences, ℓ_{N_M}, needs to be larger than ℓ. In our experiments, we use the default value $\ell_{N_M} = 3\ell$ (the results show stable performance as ℓ_{N_M} varies). We further discuss this issue in our experimental evaluation.

In the rest of this section, we describe in detail these two steps: computation of the Normal Model, and detection of anomalies.

4.1 Normal Model Based Anomaly Detection

We first discuss the problem of how to identify the anomalous subsequences in a series T, assuming that we have already computed the Normal Model $N_M = \{(N^0_M, w^0), (N^1_M, w^1)..., (N^n_M, w^n)\}$. Remember that N_M (ideally) represents the expected, normal behavior of the data. Intuitively, the anomalous subsequences are the ones that are far away from most of the subsequences in N_M.

Our technique starts by considering the pairwise distances between each subsequence of length ℓ in T to subsequences of the same length in each of N^i_M in N_M. For each subsequence N^i_M in N_M, this operation results in a meta-sequence, $N^i_M \bowtie_\ell T$ (the *join* sequence), that contains at position j the nearest neighbor distance between subsequence $T_{j,\ell}$ and any subsequence of the same length, ℓ, in N_M. We formally define the join sequence.

Definition 6 (Data Series Join) Given two data series A and B, and a subsequence length ℓ, the *Join* between A and B denoted by $(A \bowtie_\ell B)$, is a meta data series, where: $|A \bowtie_\ell B| = |B| - \ell + 1$, and $\forall i . 1 \leq i \leq |A \bowtie_\ell B|$, $(A \bowtie_\ell B)_{i,1} = min(dist(B_{i,\ell}, A_{1,\ell}), ...$
$, dist(B_{i,\ell}, A_{|A|-\ell+1,\ell}))$.

In Algorithm 2, we report the pseudo-code of the anomaly detection procedure. First we compute all the

Algorithm 2: CompAnom Normal Model Based Anomaly Detection

input : data series T, N_M, anomaly length ℓ
output: *Anomalies* - ranked list of anomalous subsequences of length ℓ

```
allJoin ← [];
foreach (N_M^i, w^i) in N_M do
    allJoin ← N_M^i ⋈_ℓ T ;
end
AnomalyScore d ← [];
foreach j ∈ [0, |T| − ℓ] do
    d ← Σ_{(N_M^i, w^i) ∈ N_M} w^i join[j]
end
Anomalies ← subsequences with top-k d values;
// number of anomalies k ∈ [1, |N_M ⋈_ℓ T|]
Anomalies ← sort subsequences in Anomalies
    in order of decreasing values in d;
```

join sequences $N_M^i \bowtie_\ell T$ (with $(N_M^i, w^i) \in N_M$), which contains the distances between each subsequence of T and their nearest neighbor in N_M^i. As described in Definition 5, we then compute the anomaly score for each subsequence. This score corresponds to the nearest neighbor distance between the subsequence to score and all the subsequences in each N_M^i in N_M. Given a subsequence $T_{j,\ell}$, we retrieve the nearest neighbor distance between $T_{j,\ell}$ and every $N_M^i \in N_M$, $(N_M^i \bowtie_\ell T)_j$. We then weigh these distances with w_i and sum them. Formally, for each subsequence in T at position j, the anomaly score is computed as:

$$d_j = \sum_{(N_M^i, w^i) \in N_M} w^i (N_M^i \bowtie_\ell T)_j \quad (1)$$

These scores represent the degree of abnormality: the larger the score is, the more abnormal the subsequence is. We then have to extract the k subsequences of length ℓ, which have the highest scores, and rank them. Algorithm 2 can also operate in an iterative fashion. This means that the algorithm can report the first (top) anomaly, and then the user can ask the algorithm to calculate and report the next anomaly, according to the sorted (in descending order) anomaly scores. The user can stop this process at any point.

[Complexity Analysis] In Algorithm 2, the anomaly extraction step is defined by the computation of $N_M^i \bowtie_\ell T$, which is bounded by $O((|T| - \ell + 1) * \ell_{N_M} * |N_M|)$, where $|N_M|$ is the number of subsequences in N_M (remember that $|N_M| << |T|$). Then it costs $O(|T| - \ell + 1)$ if we use a threshold to select the anomalies: we simply make a pass over the anomaly scores and report all subsequences that have a value greater than the threshold. If we use a Max Heap to select the subsequences with the k largest values, this becomes O(k*log(k)). Therefore, the anomalies extraction step is negligible and the complexity is $O((|T| - \ell + 1) * \ell_{N_M} * |N_M|)$.

The distance measure we use for $N_M^i \bowtie_\ell T$ is the Z-normalized Euclidean distance. Though, we can replace it with other distance measures, e.g., Dynamic Time Warping (DTW) in applications where local misalignments do not constitute anomalies.

4.2 Computing the Normal Model

So far we have assumed that we know the Normal Model, N_M. In this section, we explain how we can derive it in an automated way.

Recall that N_M should capture (summarize) the normal behavior of the data. This may not be very hard to do for a sequence T that does *not* contain any anomalous subsequences. In practice however, we would like to apply the NormA approach in an unsupervised way on any sequence, which may contain several anomalies. The challenge is then how to compute N_M based on a sequence T that contains anomalies, without user intervention and no prior knowledge of the anomalies (except for their length), and then identify the anomalous subsequences in this same sequence T.

Note that the N_M legnth, ℓ_{N_M} is larger than the anomaly length ℓ, so that we do not miss subsequences with a large overlap with an anomalous subsequence: given a subsequence of length ℓ, if we choose a normal model of length $\ell_{N_M} = 2\ell$, it will contain the subsequences overlapping with the first and last half of the anomalous subsequence, which is desirable.

We compute the N_M sequences in three steps. First, we extract the subsequences, which can serve as candidates for building the N_M. Then, we group these subsequences according to their similarity, adopting a hierarchical clustering strategy, augmented by automated identification of the right number of clusters, based on the Minimum Description Length principle. The last step consists of scoring the clusters computed in the previous step. Finally, we set the Normal model $N_M = \{(N_M^0, w^0), (N_M^1, w^1) ..., (N_M^n, w^n)\}$, with N_M^i the centroid of the i^{th} cluster, and w^i its score.

We now elaborate on these N_M computation steps.

4.2.1 Candidate Subsequences Selection

Remember that we are interested in describing the normal behavior of a system. Hence, we need to identify the subsequences (of the data series in which we wish to detect anomalies) that occur approximately the same

along the data series. These subsequences are a form of recurrent patterns, and should represent the normal behavior. Good candidate subsequences are those that satisfy the following properties: (i) they are similar to one another (normal behavior corresponds repeats approximately the same); (ii) they cover a large percentage of the data (not all extracted from the same part of the series); and (iii) they have high cardinality (appear frequently in the series).

We note that recurrent pattern discovery has been studied under the name *motif* discovery. Supervised and unsupervised motif discovery techniques assume that the user knows how to set this range threshold [41, 24], or otherwise define the target cardinality of the motif set [34]. One can thus use a motif method to extract good candidates (we show in the experimental analysis that this strategy is accurate). However, these solutions are in general very expensive (quadratic complexity). In this work, we propose a different strategy requiring less computational time.

[**Proposed Approach**] In order to discover groups of *recurrent patterns* we adopt a strategy that groups similar subsequences, without knowing beforehand their range and frequency. Since subsequence clustering has high time and memory complexity, considering every possible subsequence of a large input data series would not be a suitable solution, both in execution time efficiency, and in accuracy [26]. We thus decide to ignore some subsequences [49] and select only a subset of them in the original data series.

We describe two variations of our candidate subsequence selection strategy, one motif based strategy, and one random selection strategy. In the first strategy, we select subsequences from T that have high similarity to T (excluding overlapping subsequences). To that extent, we sort the subsequences of T according to the distances to their $1^{st}NN$ in T. We can achieve this with the self-join:

Definition 7 (Data Series Self-Join) Given a data series T, the self-join of T with subsequence length ℓ, denoted by $T\bowtie_\ell T$, is a meta data series, where: $|T\bowtie_\ell T| = |T| - \ell + 1$ and $\forall i.1 \leq i \leq |T\bowtie_\ell T|$, $(T\bowtie_\ell T)_{i,1} = dist(T_{i,\ell}, 1^{st}NN \text{ of } T_{i,\ell})$.

For each position i, the self-join sequence contains the nearest neighbor distance of the subsequence $T_{i,\ell}$ (an example is shown in Figure 1(b)). Given the self-join of T, we can discard the isolated occurrences, namely, the subsequences that do not have a close match, and thus have the highest self-join values.

Given an input data series T and its self-join ($T\bowtie_\ell T$), we define the set of the clustering candidate patterns (subsequences), $\mathbb{S}^{selfjoin}$, selected by means of the self-join:

Definition 8 (Motif Set: $\mathbb{S}^{selfjoin}$) Given a data series T and a subsequence length ℓ, $\mathbb{S}^{selfjoin} = \{T_{i,\ell_{N_M}} | 1 \leq i \leq |T| - \ell_{N_M} + 1 \wedge (T\bowtie_\ell T)_i < \epsilon\}$, where $\epsilon \in \mathbb{R}^+$. Moreover, If $T_{i,\ell_{N_M}}, T_{j,\ell_{N_M}} \in \mathbb{S}^{selfjoin} \implies |i-j| \geq \ell_{N_M}$.

The $\mathbb{S}^{selfjoin}$ set contains non-overlapping subsequences of T, which are not isolated occurrences.

In the second selection strategy, we use a random sampling strategy. Even though random motif selection could be performed [32], we decide to use uniform random sampling as a first baseline. We sample from T a subset of non-overlapping subsequences, generating the candidate set as follows:

Definition 9 (Random Set: $\mathbb{S}^{sample}$) Given a data series T, a subsequence length ℓ_{N_M}, and a sampling rate $0 < r < 1$, $\mathbb{S}^{sample} = \{T_{i,\ell_{N_M}} | 0 \leq i \leq |T - \ell_{N_M} + 1|\}$, such that $|\mathbb{S}^{sample}| < r * |\mathbb{T}_{\ell_{N_M}}|/\ell_{N_M}$. Moreover, If $T_{i,\ell_{N_M}}, T_{j,\ell_{N_M}} \in \mathbb{S}^{sample} \implies |i-j| \geq \ell_{N_M}$.

In $\mathbb{S}^{sample}$ we place the subsequences that are randomly chosen until we reach the maximum size of $|\mathbb{S}^{sample}|$ that respect the constraint in Definition 9. Thanks to the uniform distribution of the random sampling, the subsequences in $\mathbb{S}^{sample}$ also cover the entire length of the data series T.

Note that in the optimal case, where T is a *periodic* data series, we know that there are at most $|\mathbb{T}_{\ell_{N_M}}|/\ell_{N_M}$ non-overlapping recurrent patterns, assuming that ℓ_{N_M} is the length of the period. We thus consider this value as an upper bound for the $\mathbb{S}^{selfjoin}$ cardinality. This value also represents the maximum number of fixed length cycles occurring in an *aperiodic* data series. Among the datasets we consider in the empirical evaluation, the maximum value of $|\mathbb{T}_{\ell_{N_M}}|/\ell_{N_M}$ corresponds to the *1.3%* of $|\mathbb{T}_{\ell_{N_M}}|$. Moreover, we notice that setting the threshold $\epsilon = \mu(T\bowtie_\ell T)$ in $\mathbb{S}^{selfjoin}$ always allows to filter isolated subsequences in T.

4.2.2 Candidate Subsequences Clustering

At this point, we are ready to present the adopted clustering technique to group subsequences in $\mathbb{S}$ ($\mathbb{S}^{selfjoin}$, or $\mathbb{S}^{sample}$). In that regard, we consider their complete-linkage (dendrogram), resulting from the agglomerative hierarchical clustering [15]. Following previous work, we select a dendrogram cut by applying the *Minimum Description Length* principle [50, 49].

We define description length as the total number of bits used to represents a subsequence, namely its

entropy. Given a data series T, we measure its entropy $H(T)$ as:

$$H(T) = -\sum_{i=1}^{|T|}(P(T = T_{i,1})log_2 P(T = T_{i,1}) \quad (2)$$

The notation $P(T = T_{i,1})$, denotes the probability of finding the value $T_{i,1}$ in T. The description length DL of T is then defined as $DL(T) = |T| * H(T)$, and quantifies the storage requirement of a sequence. It is minimized, as a data series contains the highest number of repeated values. In this case, bits compression reduces the space.

Once the subsequences are grouped, we can represent them by using their distances to the cluster centers. If the clustering is optimal, we expect that the sequences have high similarity to their cluster centers. We consider the subsequences at the clustering stage in their SAX form (Symbolic Aggregate approXimation), where each real value is assigned a discrete label [54].

We introduce the conditional description length of a data series T (that quantifies the bits needed to store it), when knowing its cluster center sequence $Center(c)$:

$$DL(T|Center(c)) = DL(T - Center(c)) \quad (3)$$

Given a cluster of subsequences, c, (with the centroid $Center(c)$), we compute the conditional cluster description length DLC, namely the amount of bits used to encode the cluster using its center:

$$DLC(c|Center(c)) = DL(Center(c)) + \sum_{d \in c}(DL(d|Center(c))) \quad (4)$$

where the non-conditional $DLC(c) = \sum_{d \in c}(DL(d))$. Given a set of clusters A, in order to quantify the compression achieved by A, we compare the bits needed to store all the subsequences, with and without knowing $Center(c)$. We thus apply the *bitsave* measure:

$$bitsave(A) = \sum_{c \in A} DLC(c) - DLC(c|Center(c)) \quad (5)$$

In Algorithm 3, we report the clustering procedure, which selects and outputs the clusters of a dendrogram cut. The subsequences linkage is computed in Line 1. Subsequently, we iterate over the cuts in a top-down manner (Line 4). Therefore, we start by considering the cuts that produce the least number of clusters. We expect that the highest bitsave is attained grouping subsequences in the smallest amount of groups, if cluster intra-similarity is maximized. Hence, we iterate the cuts until their clusters *bitsave* stops to increase (Line 6). We thus pick the clusters resulting from the last encountered cluster. This permits to group the subsequences, maximizing their *similarity* and *frequency*.

Algorithm 3: SubsequencesClustering

input : subsequences set $\mathbb{S}$
output: a cluster set $\mathbb{C}$

```
1  Dendogram ← CompleteLinkage(𝕊);
2  ℂ ← ∅;
3  lastBitsave ← −∞;
4  foreach cut in Dendogram in top-down order
     do
5      ℂ′ ← get subsequences clusters from cut;
6      if bitsave(ℂ′) > lastBitsave then
7          ℂ ← ℂ′;
8          lastBitsave ← bitsave(ℂ′);
9      else
10         break;
11     end
12 end
```

4.2.3 Candidate Clusters Scoring

Each cluster we compute in Algorithm 3, becomes the candidate group of subsequences (candidate cluster), that are considered to build the Normal Model. We now propose a scoring function, which permits to compute w^i (that can be seen as the normality degree) for each candidate clusters i. Intuitively, the cluster and subsequences with the top score are the most representative of the *different*, *recurring* patterns in the entire data series; the next cluster is less representative (but still contains subsequences that are close to normal behavior).

Let $\mathbb{S} \subseteq \mathbb{T}_{\ell_{N_M}}$ be a subset of subsequences in T of length ℓ_{N_M}. We can then compute the *coverage* of $\mathbb{S}$, $Coverage(\mathbb{S}) = MaxOffset(\mathbb{S}) - MinOffset(\mathbb{S})$, which measures the distance between the maximum and minimum offsets in T (of two $\mathbb{S}$ subsequences), and corresponds to the span of T from where the subsequences in $\mathbb{S}$ were extracted. We will also refer to the *frequency* of $\mathbb{S}$, $Frequency(\mathbb{S}) = |\mathbb{S}|$ (equal to the cardinality of $\mathbb{S}$).

Moreover, we want to consider an *inter-clustering* property, namely the *centrality*. We borrow this definition from the graph analysis literature [60], which states that the most central node in a graph denotes its influence. Given a cluster set $\mathbb{C}$ and a cluster $c \in \mathbb{C}$, we define *centrality* as:

$$centrality(c, \mathbb{C}) = \frac{1}{\sum_{x \in \mathbb{C}} dist(Center(c), Center(x))} \quad (6)$$

Recall that a cluster of subsequences, denoted by c, formally coincides with a set of subsequences $\mathbb{S}$. The

Center function we adopt in our work is the *centroid*, which is the arithmetic mean vector of the subsequences in a cluster c.

Intuitively, in order to set the weights w^i for all clusters i, we need to consider the subsequences that most often occurs along the largest part of the data. This translates to identifying the cluster with the highest frequency and the largest coverage. In order to account the most recurrent subsequence, we also adopt the centrality measure. If a subsequence is the most recurrent, we expect that all its occurrences are grouped in the cluster with the highest centrality.

We are now ready to score the candidate clusters, taking into account the *frequency* and *coverage* of the subsequences in each cluster, and its *centrality* as well. After normalizing *Frequency(c)*, *Coverage(c)*, and *Centrality(c)* so that each lies in the $[1, 2]$ interval for all $c \in \mathbb{C}$ (normalization is needed so that all three criteria have equal weight), the score we assign to a cluster c, given also the complete clusters set $\mathbb{C}$, is the following:

$$Norm(c, \mathbb{C}) = Frequency(c)^2 \times Coverage(c) \times centrality(c, \mathbb{C}) \quad (7)$$

The $Norm$ function provides an index, with regards to the Normal Model properties we take in consideration. Since high *coverage* values might erroneously be assigned to clusters with low *frequency*, we favor clusters that have high *frequency*. For this reason, it appears squared in Equation 7.

4.2.4 Normal Model Extraction

In Figure 4(a), we report the cluster scores we obtain for the MBA ECG recordings (patient 803). In the plot, we report each cluster $Norm$ score (the size of the red point is proportional to $Frequency(c)$) coupled with their *coverage* (blue line), which starts and ends respectively at the smallest and largest offset of the cluster subsequences. In the right part of Figure 4, we depict the subsequences in each cluster. The x-axis value assigned to each red point is the arithmetic mean of its subsequences offsets in the corresponding cluster. This set of clusters $\mathbb{C} = \{c_0, ..., c_n\}$ will be used in the normal model $N_M = \{(N_M^0, w^0), ...(N_M^n, w^n)\}$, with $N_M^i = Center(c_i)$ and $w^i = Norm(c_i, \mathbb{C})$.

In this example, the subsequences contained in the cluster with the highest $Norm$ score, represent correct Heartbeat Ventricles contracts. The centroid of this cluster will be the most influential in N_M. On the other hand, clusters with low scores contain subsequences that do not represent any known features (they may be noise, or even repeated anomalies) and therefore, will not have a real influence in N_M.

Algorithm 4: CompNM Compute *Normal Model*

input : data series T, Normal Model length ℓ_{N_M}
output: Normal Model N_M

1 compute $\mathbb{S}^{selfjoin}$ (or $\mathbb{S}^{sample}$) from T;
`// compute the set of subsequences clusters in` T `(`$\mathbb{C}$`)`
2 $\mathbb{C} \leftarrow SubsequencesClustering(\mathbb{S}, \ell_{N_M})$;
3 $N_M \leftarrow \{\}$;
4 **for** c *in* $\mathbb{C}$ **do**
5 | add $(centroid(c), Norm(c, \mathbb{C}))$ in N_M;
6 **end**

4.2.5 Overall Algorithm

The overall procedure for computing the Normal Model is then structured as shown in Algorithm 4. In Line 1, we select a subset of subsequences, $\mathbb{S}$, applying one of the two strategies we discussed earlier (i.e., $\mathbb{S}^{selfjoin}$, or $\mathbb{S}^{sample}$), which take into consideration several desired characteristics of the correct (non-anomalous) part of the data. Subsequently we cluster them in Line 2. In Line 4, we iterate each cluster that is assigned to the $Norm$ score (Line 5), and then added to the normal model as a tuple composed of its centroid and its score. The assigned score quantifies how much a group of similar subsequences (cluster) supports the properties we define over correct data. We use NormA-SJ to refer to the algorithm that uses $\mathbb{S}^{selfjoin}$, and NormA-smpl for the variation with $\mathbb{S}^{sample}$.

[Complexity Analysis] The complexity of Algorithm 4 depends on the choice of the subsequence selection strategy, performed in the initial part. We can compute $\mathbb{S}^{selfjoin}$, using the state-of-the art algorithm *Stomp* [66] in $O(|T|^2)$ time. On the other hand, computing $\mathbb{S}^{sample}$, takes linear time in the worst case ($O(|T|)$). In the experimental evaluation we test the two selection strategies in isolation to assess their accuracy separately. Subsequently, the subsequences linkage computation takes $O(\ell|\mathbb{S}|^2)$.

It is important to note that the space of $|\mathbb{S}|$ is in general two order of magnitude smaller than the original space of T. In turn, selecting a dendrogram cut has worst case time complexity of $O(\ell|\mathbb{S}|^2)$, when all the cuts need to be evaluated. As we show in the experimental evaluation, the number of cuts considered in Algorithm 4 is very small in practice.

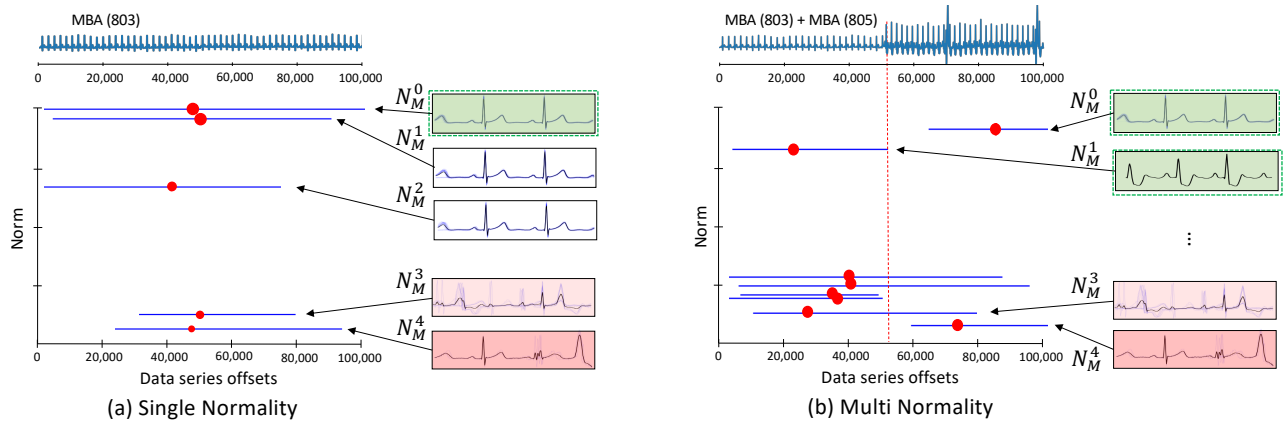


Fig. 4 (a) *Norm* cluster scoring of MBA ECG recordings (patient 803); (b) *Norm* cluster scoring of the concatenation of two MBA ECG recordings (patient 803 and 805).

4.3 Multiple Normal Behaviors

In general, a system may be characterized by several (i.e., more than one) different recurrent patterns that all correspond to normal behaviors. This may happen when the underlying generation process changes among multiple different normal states of operation (e.g., when a machine has two, or more operating states). In such cases, we would expect the occurrence of multiple different and valid Normal Models subsequences as well.

Thanks to the Normal Model structure, multiple normal patterns can be identified. Assume a data series that is composed of two segments (partitions) corresponding to two different sets of normal behavior subsequences (patterns). If these two subsequence sets have the same cardinality (i.e., the two segments are similar in size), then both of them will be represented by one of the Normal patterns N_M^i, N_M^j in the Normal Model N_M, and both N_M^i, N_M^j will have similar weights w_i, w_j. In principle, NormA is capable to handle data series composed of different segments. Figure 4(b) depicts the scoring step on a data series composed of two MBA ECG datasets. As we can see, the clusters are distributed between the two parts that correspond to the offsets of the two segments. Moreover, the two normal patterns (N_M^0, N_M^1) have similar scores (i.e., value on the y-axis), and thus, will have the same significance on the distance computation to the normal model.

However, since the normal subsequences of each segment may be significantly different from one another, it may be the case that an anomalous subsequence in one of the segments is similar to the normal subsequences of some other segment. In this case, the algorithm will not be able to detect this anomalous subsequence, which is obviously not desirable. In order to remove this undesirable effect, we define NormA-mn, a variant of NormA-smpl, where we use a different method to compute the distance to the Normal Model. For each subsequence $T_{j,\ell}$ of T, the anomaly score is defined as the distance $\tilde{d}_j$ of that subsequence to the Normal Model, computed as follows:

$$\tilde{d}_j = \left(\sum_{N_M^i} w^i (N_M^i \bowtie_\ell T)_j \right) - \beta_j \tag{8}$$

In the equation above, $(N_M^i \bowtie_\ell T)_j$ represents the distance of $T_{j,\ell}$ to its nearest neighbor in N_M^i, while the role of parameter β_j is to suppress the aforementioned noise. Assuming that S is the changing point of the two segments of T, β_j should be equal to:

$$\beta_j = \begin{cases} \dfrac{\sum_{k\in[0,S]} d_k}{S} \text{ if we have: } j \in [0, S] \\ \dfrac{\sum_{k\in[S,|T|]} d_k}{|T| - S} \text{ if we have: } j \in [S, |T|] \end{cases} \tag{9}$$

However, the changing point S is usually not known in practice (and remains a challenging research problem [22]). Moreover, it becomes even more difficult if there is more than one changing point to find (i.e., for triple and quadruple normalities). Thus, we compute β_j as the average distance of the Normal Model to subsequences in a time interval of length τ around the subsequence $T_{j,\ell}$:

$$\beta_j = \frac{\sum_{k\in[I^b_{j,\tau}(T), I^e_{j,\tau}(T)]} d_k}{2\tau} \tag{10}$$

with:

$$I^b_{j,\tau}(T) = max(0, max(0, j - \tau) - max(j + \tau - |T|, 0))$$
$$I^e_{j,\tau}(T) = min(|T|, min(|T|, j + \tau) + max(0, \tau - j))$$

Note that in the above equation, $\tau \in [1, |T|]$. In the specific case when $\tau = |T|$, we have $I^b_{j,\tau}(T) = 0$ and $I^e_{j,\tau}(T) = |T|$ and thus $\tau = \mu(T)$, with $\mu(T)$ the mean of the entire data series. As a matter of fact, using $\tau = |T|$ is similar to using the classical $NormA$ method. In practice, β_j is accurate if τ is large enough to consider a representative time neighborhood of the segment with mostly normal subsequences (and maybe also a few anomalies). In the rest of this paper, we set $\tau = 2|N_M|$. Our experimental evaluation shows that varying this parameter does not have a strong influence on the performance of our approach.

5 Experimental Evaluation

In this section, we present the experimental results with real datasets from different domains, including *all* annotated datasets that have been used in the *discord* discovery literature. To ensure reproducibility, we created a web page [3] with the source code and datasets.

The experiments we conduct demonstrate the effectiveness of NormA. We test the accuracy of anomaly detection in datasets characterized by the presence of repeated (similar) anomalies, but also in datasets, where anomalous occurrences correspond to rare patterns, namely discords. We also study the scalability of NormA, considering data series of different and increasing size, and a real-use case dataset containing 20M of points.

[Summary of Results] In summary, the experimental analysis demonstrates the superiority of NormA against the current state of the art approaches, both in terms of accuracy and scalability. In particular, over a wide variety of datasets, NormA significantly outperforms (overall) all the competitors used in our analysis, including time series discord discovery algorithms, outlier algorithms for multidimensional data, and a deep learning technique. Moreover, the results show that NormA is up to one order of magnitude faster than the competitors, irrespective of the anomaly length considered (ℓ), or the dataset characteristics (number of anomalies, dataset length). Finally, we showcase the meaningful results that NormA produced for two diverse real use-cases.

5.1 Setup

We implemented our algorithms in C (compiled with gcc 5.4.0) and Python 3.5. The evaluation was conducted on a server with Intel Xeon CPU E5-2650 2.20GHz and 250GB RAM.

[Datasets] We benchmark our system using real and synthetic datasets, for all of which a ground truth of annotated anomalies is available (Table 2). Following previous work [53], we use several synthetic datasets that contain sinusoid patterns at fixed frequency following a random walk trend (Figure 5). We then inject different number of anomalies, in the form of sinusoid waveforms with different phases and higher than normal frequencies (Figure 5(a)), and add various levels of Gaussian noise on top (Figure 5(b)). We refer to those datasets using the label SRW-[# of anomalies]-[% of noise]-[length of anomaly], and use them in order to test the performance of the algorithms under different, controlled conditions.

Our real datasets are the following. Simulated engine disks data (SED) from the NASA Rotary Dynamics Laboratory [5]. representing disk revolutions recorded over several runs (3K rpm speed). MIT-BIH Supraventricular Arrhythmia Database (MBA) [23,40], which are electrocardiogram recordings from 5 patients, containing multiple instances of two different kinds of anomalies. Five additional real datasets from various domains that have been studied in earlier works [28,52], and their anomalies are simple *discords* (usually only 1): aerospace engineering (Space Shuttle Marotta Valve [28]), gesture recognition (Ann's Gun dataset [52]), medicine (Patient's respiration measured by the thorax extension [28], ECG recordings qtb/sel102 [28]), and electrical consumption study (Dutch Power Consumption data [28]).

Finally, we use the following two non-annotated datasets. The Nasa Bearing dataset [1] that consists of individual files that are 1-second vibration signal snapshots of bearings installed on a shaft, and the New York City Taxi and Limousine Commissions dataset (NTC) [2] that records the number of New York City taxi passengers every for every 30 minutes from July 2014 to January 2015.

We note that the largest datasets used in the literature have length of 15,000 and 36,000 points [28, 52]. In contrast, we use sequences that are 2 and 3 orders of magnitude larger, with a maximal length up to 20,000,000 points (Nasa Bearing).

[Algorithms] We compare NormA to the current state-of-the-art algorithms. We consider two techniques that enumerate *Top-k* 1^{st} discords, GrammarViz (GV) [52] and STOMP [64]. Moreover, we compare NormA against the Disk Aware Discord Discovery algorithm (DAD) [61], which finds m^{th} discords. We also compare to Local Outlier Factor (LOF) [14] and Isolation Forest [36]. These two methods are not specific to subsequence anomaly detection, but constitute strong baselines from the literature on multi-dimensional data outlier detection. Finally, we include in our comparison LSTM-AD [39], a *semi-supervised* deep learning technique. Note that the

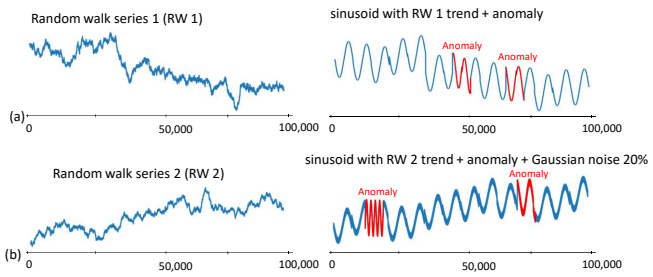


Fig. 5 Synthetic datasets. *(a)* Random walk sequence (left), and sinusoid signal following the same trend (right) with injected anomalies (red/bold subsequences). *(b)* A second example, with 20% of Gaussian noise added on top.

Datasets	Length	ℓ	$\mathbf{N_A}$	Domain
Annotated				
SED	100K	75	50	Electronic
MBA (803)	100K	75	62	Cardiology
MBA (805)	100K	75	66	Cardiology
MBA (806)	100K	75	27	Cardiology
MBA (820)	100K	75	76	Cardiology
MBA (14046)	100K	75	142	Cardiology
Marotta Valve	20K	1K	1	Aerospace engineering
Ann Gun	11K	800	1	Gesture recognition
Dutch Power Consumption	35K	800	3	Elect. cons. study
Patient Respiration	24K	800	1	Medicine
SRW-[20-100]-[0%]-[200]	100K	200	var.	Synthetic
SRW-[60]-[5%-25%]-[200]	100K	200	60	Synthetic
SRW-[60]-[0%]-[100-1600]	100K	var.	60	Synthetic
Non-Annotated				
NYC Taxi (NTC)	10K	100	-	Transport
Nasa Bearing	20M	20K	-	Bearings

Table 2 List of dataset characteristics: series length, anomaly length (ℓ), number of annotated anomalies ($\mathbf{N_A}$), domain.

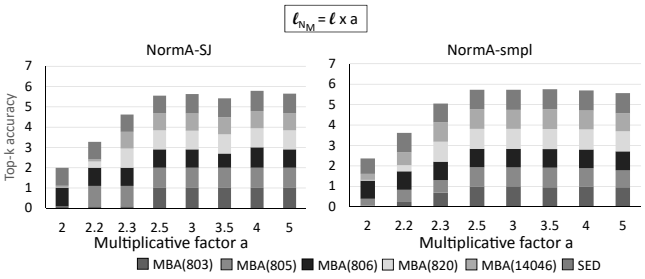


Fig. 6 Cumulative P@k anomaly detection accuracy for NormA-SJ (*left*) and NormA-smpl (*right*) on the 6 real annotated datasets with multiple anomalies, when varying the Normal Model length.

comparison to LSTM-AD is not fair to all the other techniques: LSTM-AD has to first train on labeled normal data, which gives it an unfair advantage; all the other techniques are *unsupervised*. We include it to get an indication as to how the unsupervised techniques compare to a state-of-the-art supervised anomaly detection algorithm. In practice, we train LSTM-AD on the longest subsequence without anomalies: 4109-10846 points (7000 on average).

[Measures] We use the precision-at-k (P@k) accuracy measure to evaluate the effectiveness of the methods. P@k accuracy is defined as the number of correctly identified anomalies among the k answers of the algorithm, divided by k. (This corresponds to precision on the anomaly class $TP_A/(TP_A + FP_A)$, where TP_A is the number of detected true anomalies, and FP_A the number of false positives.) Note that we use k only for evaluation purposes: none of the algorithms tested in the following section require k as a parameter. In our accuracy evaluation, we set k to the number of anomalies in the sequence ($k = N_A$ of Table 2). Recall that the annotated datasets we use in this work have *all* their anomalies annotated.

We also measure time, in order to evaluate the efficiency and scalability of the methods.

5.2 Normal Model Tuning

In this section, we evaluate the sensitivity of the Normal model N_M, as a function of its length ℓ_{N_M} (relevant for NormA-SJ and NormA-smpl), and of the sampling rate r (relevant for NormA-smpl).

First, we measure the performance for P@k anomaly detection, setting k equal to the number of anomalies contained in each one of our six real annotated datasets with multiple anomalies, and we vary the length of the Normal Model (ℓ_{N_M}), using a multiplicative factor ranging between 2-5 times the anomalous pattern length ℓ. Figure 6 shows the cumulative accuracy for each Normal Model length we tested (the results for NormA-smpl are averages over 100 runs). We compute accuracy as the ratio of correctly identified anomalies over the total number of anomalies in each dataset.

We observe that the accuracy values become stable once the Normal Model length is at least 2.5x larger than the anomaly length. We also note that this behavior is the same for both NormA-SJ and NormA-smpl, and moreover, absolute accuracy values are in both cases almost the same. In all following experiments, we set the Normal Model length to the default value of 4x the anomaly length.

Second, we computed accuracy as we vary the sampling ratio r (see Definition 9) for computing the Normal Model for NormA-smpl. We varied r between 0.1-0.6, and observed that accuracy remained always (almost) stable (graphs omitted for brevity). In all following experiments, we use the default value $r = 0.4$.

Overall, we note that NormA *only* needs ℓ as an input parameter (all the rest are set as discussed above). Note that ℓ is also an input parameter for all other subsequence anomaly detection algorithms, which nevertheless also need additional user-defined parameters (e.g., LOF and DAD require the number of similar anomalies, m, that we want to detect).

5.3 Distance Measure Impact

In this section, we evaluate the impact of the distance measure used in the NormA framework (we use NormA-smpl as our baseline). For this purpose, we use in the *dist* function of 5 the *Euclidean* distance (i.e., the core distance measure for our proposed method), the *Shape-Based Distance* (*SBD*) [45], and the *Dynamic Time Warping* (*DTW*) distance.

Figure 7(a) depicts the NormA-smpl score for the three distance measures for a 6000 points snippet of the MBA(803). In Figure 7(b), we depict the averaged accuracy results over 10 different runs for the SED and all the MBA datasets. The results show that the *SBD*, *DTW* and *Euclidean* distances lead to similar results (with no clear winner). Overall, *Euclidean* provides accurate results. Moreover, through the use of the MASS algorithm [64] it is significantly faster than the other two distance measures. We thus use this distance for the rest of the experimental section.

5.4 Anomaly Detection Evaluation

In this section, we report the the anomaly detection accuracy results.

[Anomalies Detection Accuracy] In Table 3, we show the P@k accuracy (correctly identified anomalies among the k retrieved divided by k), with k equal to the number of anomalies. These experiments test the capability of each method to correctly retrieve the k anomalous subsequences in each dataset. For NormA, we simply have to report the P@k anomalies that the algorithm produces. In the same manner, we compute accuracy for Isolation Forest and LOF, considering the k subsequences assigned with the highest scores by these two approaches. For the *discord* based techniques, we have to consider the *Top-k* 1^{st} *discord* and the m^{th} *discord* (with $m = k$). Finally, LSTM-AD marks as anomalies the subsequences that have the largest errors (distances) to the sequences that the LSTM-AD algorithm predicts; we compute accuracy considering the subsequences with the k largest errors.

In the first section of Table 3, we report the results of all techniques on the annotated real datasets with multiple (diverse and similar) anomalies. NormA is clearly the winner, with the exception of MBA(14046), for which its performance is still very close to the best performer. As expected, *Top-k* 1^{st} *discord* techniques (GV and STOMP) achieve low accuracy, since anomalies do not correspond to rare subsequences (i.e., isolated discords). We also observe that the m^{th} discord technique (DAD), which is able to detect groups of m similar anomalous subsequences, does not perform well, either. This is due to the many false positives produced by the algorithm.

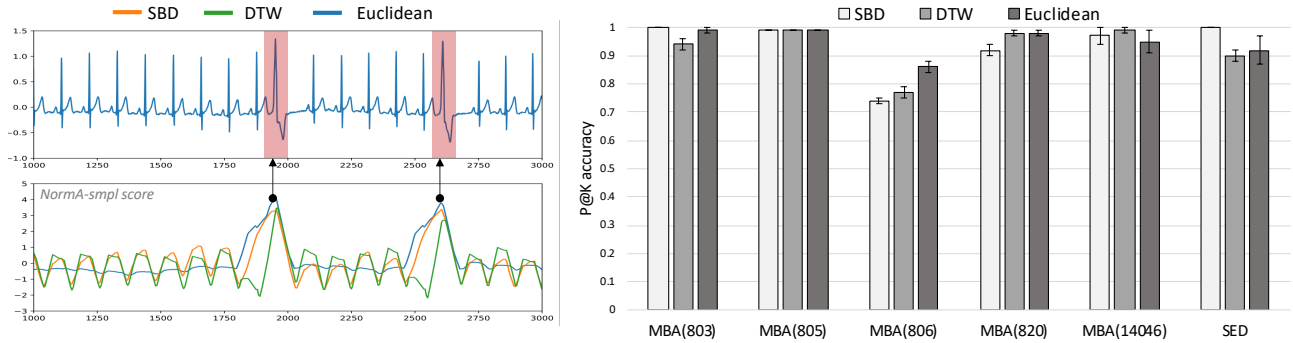
In the other three sections of Table 3, we report the accuracy of the evaluated methods on all the synthetic datasets (where we vary the number of anomalies, the % of Gaussian noise, and the anomaly subsequence length ℓ). We note that the accuracy of the *discord* discovery techniques substantially improves, since in this case most anomalies correspond to rare and isolated subsequences (i.e., different from one another). Even in these cases, NormA is clearly superior to the competitors. In contrast to GV, STOMP and DAD, NormA's performance is stable for increasing noise.

Regarding LSTM-AD, we note that in general it is more accurate than the *discord* based algorithms. Nevertheless, we stress that LSTM-AD only achieves this performance, because (contrary to the rest of the techniques) it benefits from a training phase on labeled data. However, in several situations labeled data are not available (and extremely expensive to generate). Even as such though, LSTM-AD cannot match the performance of NormA. Since we would expect a supervised algorithm to perform at least as good as an unsupervised one, these results suggest that supervised methods still have lots of potential for improvement.

Regarding LOF, we observe that it does not perform well in our context. Isolation Forest achieves better performance, but not as good as NormA.

Overall, we observe that NormA is more accurate than all competitors (with very few exceptions, for which its performance is still very close to the best one), in all the settings we used in our evaluation. Furthermore, we note that the performance of NormA-smpl is in almost all cases equal to that of NormA-SJ, or very close to it.

[Critical Difference Diagram] After rejecting the null hypothesis using the Friedman test, we use the pairwise Post-Hoc Analysis using a Wilcoxon signed-rank test [59] to test and produce the critical difference diagram for the algorithms and datasets of Table 3. The critical difference diagram with $\alpha = 0.05$ (Figure 8) shows that NormA-SJ and NormA-smpl are the overall winners, with NormA-SJ and NormA-smpl being significantly better than all previous algorithms.



(a) NormA-smpl result for MBA(803) using SBD, DTW and Euclidean distances

(b) NormA-smpl accuracy on MBA datasets using SBD, DTW and Euclidean distances

Fig. 7 Distance measure impact experiment. (a) NormA-smpl accuracy score for MBA(803) for *sbd*, *DTW* and *Euclidean* distances. (b) Overall accuracy for all the MBA datasets.

Datasets	GrammarViz	STOMP	DAD	LSTM-AD	LOF	Isolation Forest (stddev)	NormA-smpl (stddev)	NormA-SJ
SED	0.46	0.57	0.44	0.10	0.65	0.65 (0.02)	**0.92 (0.05)**	0.91
MBA (803)	0.15	0.72	0.01	0.35	0.08	**1.00 (0.00)**	0.99 (0.01)	**1.00**
MBA (805)	0.09	0.10	0.03	0.85	0.42	0.99 (0.01)	**0.99 (0.00)**	**0.99**
MBA (806)	0.01	0.59	0.66	0.10	**0.92**	0.75 (0.06)	0.86 (0.02)	0.85
MBA (820)	0.05	0.92	0.04	0.09	0.42	0.92 (0.03)	**0.98 (0.01)**	**0.98**
MBA (14046)	0.09	0.54	0.71	**1.00**	0.64	0.99 (0.01)	0.95 (0.04)	0.93
SRW-[20]-[0%]-[200]	**1.0**	0.77	0.55	0.94	0.74	0.75 (0.05)	**1.00 (0.00)**	**1.00**
SRW-[40]-[0%]-[200]	0.97	**1.0**	0.05	**1.00**	0.89	0.92 (0.02)	0.97 (0.01)	0.97
SRW-[60]-[0%]-[200]	0.96	0.88	0.10	0.92	0.76	0.87 (0.02)	0.99 (0.01)	**1.00**
SRW-[80]-[0%]-[200]	0.96	0.43	0.14	0.95	0.82	0.86 (0.01)	**0.98 (0.00)**	**0.98**
SRW-[100]-[0%]-[200]	0.95	0.99	0.11	**1.00**	0.75	0.92 (0.02)	**1.00 (0.00)**	**1.00**
SRW-[60]-[5%]-[200]	**1.0**	0.73	0.21	0.96	0.88	0.89 (0.01)	**1.00 (0.00)**	**1.00**
SRW-[60]-[10%]-[200]	0.83	**0.98**	0.01	0.94	0.70	0.80 (0.01)	**0.98 (0.00)**	**0.98**
SRW-[60]-[15%]-[200]	0.76	0.62	0.17	0.94	0.66	0.82 (0.01)	0.99 (0.01)	**1.00**
SRW-[60]-[20%]-[200]	0.73	**1.0**	0.01	0.96	0.73	0.85 (0.02)	**1.00 (0.00)**	**1.00**
SRW-[60]-[25%]-[200]	0.63	0.64	0.09	0.83	0.67	0.80 (0.01)	**0.99 (0.01)**	0.94
SRW-[60]-[0%]-[100]	0.98	**1.0**	0.23	**1.00**	0.74	0.88 (0.02)	**1.00 (0.00)**	**1.00**
SRW-[60]-[0%]-[200]	0.96	0.60	0.19	**1.00**	0.85	0.83 (0.01)	**1.00 (0.00)**	**1.00**
SRW-[60]-[0%]-[400]	0.98	**1.0**	0.63	0.88	0.76	0.88 (0.01)	0.98 (0.01)	**1.00**
SRW-[60]-[0%]-[800]	0.91	0.86	-	0.76	0.69	0.87 (0.01)	0.97 (0.02)	**0.98**
SRW-[60]-[0%]-[1600]	**1.0**	**1.0**	-	0.90	0.52	0.64 (0.02)	0.92 (0.04)	0.97
average	0.62	0.73	0.24	0.78	0.68	0.85	0.97	**0.98**

Table 3 P@k accuracy for DAD, STOMP, GrammarViz, LSTM-AD, NormA-smpl (standard deviation over 100 runs shown in parenthesis), and NormA-SJ. We set k equal to the number of anomalies, and ℓ to the length of the (annotated) anomalies.

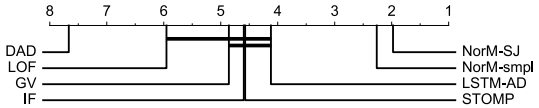


Fig. 8 Critical difference diagram ($\alpha = 0.05$) for the data series of Table 3.

[**Varying k in P@k**] In this part, we measure P@k accuracy for different values of k (*1,5,10,50,100*). The objective of this experiment is to evaluate the anomaly detection, testing the ability of each technique to assign and place the real anomaly in the first k places of the ranking, for a variable k. The results shown in Table 4 show that NormA is the technique with the best and most stable performance. Figure 9 helps us understand why. In this figure, we depict on the *left*, the distribution of the distances of each subsequence in the MBA(805) dataset to their NN in the Normal Model, built by NormA. On the *right*, we show for the same dataset, the distribution of the distances between each subsequence and their NN in the dataset itself (excluding trivial matches). In both diagrams, each bar is gradually colored according to the number of distances that belong to annotated anomalous subsequences, from dark/black (many) to gray/light (few). We observe that on the right plot, the subsequences with the k largest NN distances are not the annotated anomalies, whereas on the left plot the subsequences with the k largest distances are the true anomalies, which are also the P@k anomalies discovered by NormA. These results demonstrate that NormA is able to correctly rank the real anomalies, according to the highest distances to the NN in the Normal Model, whereas in the *discord* ranking there are many subsequences with high NN distance that are not anomalous (false positives).

[**Rare Subsequence Anomalies**] To further evaluate the quality of the Normal Model, we consider a collection of datasets, widely used in the data series anomaly (discord) literature. Those are datasets characterized by one (three for Dutch Power Consumption) anoma-

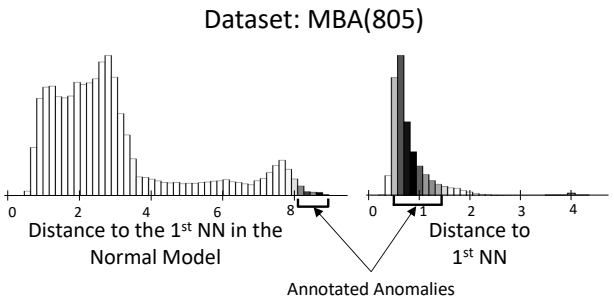


Fig. 9 Distribution of nearest neighbor (NN) distance of the MBA(805) subsequences. Bars are colored according the number of distances that belong to anomalous subsequences, from dark/black (many) to gray/light (few). *(left)* Distribution of distances to the NN in the Normal Model built by NormA. *(right)* Distribution of the distances to the NN in the dataset (excluding trivial matches).

Method	P@k					
	1	5	10	50	100	average
GV	0.25	0.13	0.20	0.18	0.23	0.15
DAD	0.50	0.30	0.30	0.34	0.41	0.37
STOMP	0.50	0.53	0.58	0.58	0.46	0.44
LSTM	0.75	0.61	0.58	0.62	0.61	0.64
LOF	0.83	0.86	0.81	0.72	0.56	0.64
IF	**1.00**	**0.96**	0.95	0.82	0.62	0.72
NormA-smpl	**1.00**	**0.96**	**0.98**	0.91	0.65	0.75
NormA-SJ	**1.00**	**0.96**	**0.98**	**0.92**	**0.66**	**0.75**

Table 4 Anomaly detection accuracy (average value of all annotated datasets in Table 2) for different P@k, of NormA and the competitors.

lous subsequences, which correspond to the P@k 1^{st}-discord. In Figure 10(*left)*, we report the excerpts of those datasets, whereas in Figure 10(*right*), we depict the Normal Model subsequence with the largest $Norm$ score (weight w^i) computed by NormA in green/light color, and the discord in red/dark color. The Normal Model subsequence with the largest $Norm$ score is in all cases very different than the discords, which are always correctly identified by NormA as the Top-1 anomalies.

5.5 Multi Normality

In this experiment, we demonstrate the ability of NormA-mn to capture anomalies in data series that have more than one normal behavior patterns. By concatenating real datasets enumerated in Table 2 (SED and MBA datasets), we evaluate the P@k accuracy of NormA-mn and some state-of-the-art methods for datasets with 2-4 different normal patterns, for a total of points equal to 200,000 (for two-normality), 300,000 (for three-normality), and 400,000 points (for four-normality). Note that apart from having different normal patterns, the concatenated data series also have different value ranges in each segment. These are challenging cases for our problem.

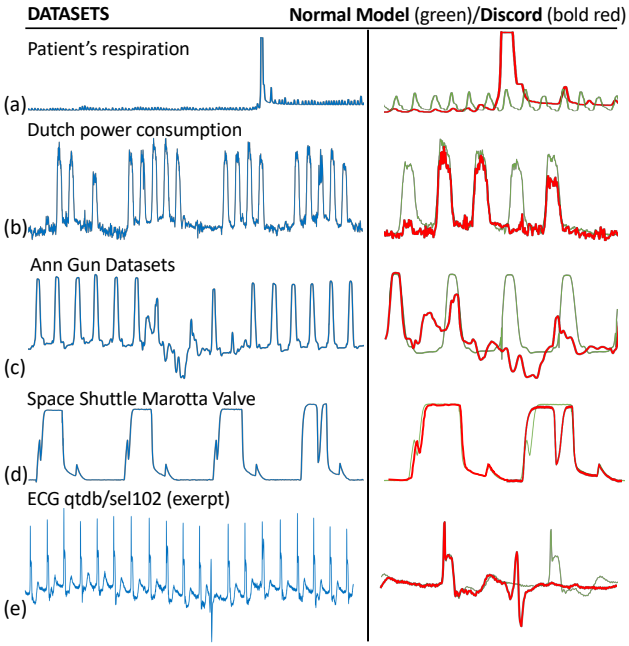


Fig. 10 *(left)* Excerpts of 5 datasets used in the literature. (a) Patient's respiration [28]. (b) Dutch Power Consumption [28,52]. (c) Ann Gun centroid dataset [52]. (d) Space Shuttle Marotta Valve dataset [28]. *(right)* Normal model subsequence with the largest $Norm$ score extracted (green/light), and anomalous pattern (red/dark).

In this experiment, we only consider the three best competitors according to Table 3, namely, STOMP, Isolation Forest (IF) and Local Outlier Factor (LOF). We assume that the segmentation is not known: thus, Norma and the other methods are run on the entire data series, without any information on where each segment starts and ends. (We do not include LSTM-AD in this experiment, because it would need to be trained on normal subsequences from each different segment, and thus require prior knowledge of the segments as well.)

Table 5 shows the P@k accuracy (average results over 10 executions). The results show that the change of normal behavior by the different segments of the series does not have a strong impact on the anomaly discovery accuracy of NormA-mn. On the contrary, IF is significantly less accurate, which means that it sensitive to normality changes (compare to Table 3). Table 5 also shows that the accuracy of IF is getting significantly smaller as the number of the different normal behaviors increase. This does not affect much the other methods.

Figure 11 summarizes all the above results, and compares the accuracy between NormA-mn and IF/LOF/STOMP (Figure 11(a,b,c), respectively) for datasets with single, double, triple and quadruple normality. These graphs show that the majority of points (representing the datasets of Table 5) are under the diagonal, which means that NormA-nm is more accurate for the major-

Data Series	Anomalies repartition	STOMP	LOF	Isolation Forest (stddev)	NormA-mn (stddev)
Double Normality					
MBA(803 + 805)	0.48/0.52	0.32	0.53	0.53(0.03)	**0.95(0.02)**
MBA(803 + 806)	0.69/0.31	0.58	0.64	0.75(0.00)	**0.89(0.03)**
MBA(803 + 820)	0.45/0.55	0.78	0.67	0.75(0.05)	**0.92(0.01)**
MBA(803 + 14046)	0.30/0.70	0.52	0.80	**0.97(0.02)**	0.41(0.09)
MBA(803) + SED	0.55/0.45	**0.67**	0.44	0.45(0.00)	0.60(0.14)
MBA(805 + 806)	0.83/0.17	0.20	0.54	0.68(0.04)	0.85(0.01)
MBA(805 + 820)	0.46/0.54	0.51	0.79	0.49(0.04)	**0.97(0.00)**
MBA(805 + 14046)	0.31/0.69	0.43	0.68	**0.88(0.07)**	0.64(0.10)
MBA(805) + SED	0.56/0.44	0.30	0.39	0.37(0.01)	0.87(0.05)
MBA(806 + 820)	0.26/0.74	0.83	0.68	0.78(0.00)	**0.92(0.01)**
MBA(806 + 14046)	0.16/0.84	0.59	**0.85**	0.84(0.00)	0.58(0.05)
MBA(806) + SED	0.35/0.65	0.62	0.62	0.57(0.01)	0.84(0.01)
MBA(820 + 14046)	0.34/0.66	0.70	**0.81**	0.66(0.00)	0.65(0.04)
MBA(820 + SED	0.60/0.40	0.82	0.40	0.38(0.01)	0.92(0.02)
MBA(14046) + SED	0.73/0.27	0.61	0.61	0.28(0.00)	0.83(0.08)
Average	0.47/0.53	0.56	0.63	0.63	0.79
Triple Normality					
MBA(803 + 805 + 806)	0.40/0.42/0.18	0.37	0.56	0.43(0.02)	**0.84(0.01)**
MBA(803 + 805 + 820)	0.30/0.32/0.37	0.54	0.67	0.37(0.02)	**0.86(0.06)**
MBA(803 + 805) + SED	0.35/0.37/0.28	0.41	0.41	0.26(0.00)	0.60(0.12)
MBA(803 + 805 + 14046)	0.23/0.24/0.53	0.44	0.71	**0.73(0.02)**	0.25(0.21)
MBA(803 + 806 + 820)	0.38/0.16/0.46	0.74	0.66	0.68(0.05)	**0.82(0.04)**
MBA(803 + 806) + SED	0.45/0.19/0.36	0.62	0.52	0.34(0.01)	**0.67(0.04)**
MBA(803 + 806 + 14046)	0.27/0.12/0.61	0.53	0.80	**0.88(0.00)**	0.32(0.22)
MBA(803 + 820) + SED	0.33/0.40/0.27	**0.75**	0.41	0.26(0.00)	0.64(0.02)
MBA(803 + 820 + 14046)	0.22/0.27/0.51	0.64	**0.80**	0.73(0.00)	0.66(0.07)
MBA(805 + 806 + 820)	0.39/0.16/0.45	0.53	0.69	0.41(0.02)	0.85(0.00)
MBA(805 + 806) + SED	0.46/0.19/0.35	0.35	0.49	0.31(0.00)	0.79(0.02)
MBA(805 + 820) + SED	0.34/0.40/0.26	0.54	0.44	0.24(0.00)	**0.94(0.01)**
MBA(806 + 820) + SED	0.18/0.50/0.32	0.78	0.48	0.31(0.00)	**0.82(0.01)**
Average	0.33/0.29/0.38	0.55	0.59	0.46	0.70
Quadruple Normality					
MBA(803 + 805 + 806 + 820)	0.27/0.29/0.12/0.33	0.53	0.67	0.32(0.02)	**0.86(0.03)**
MBA(803 + 805 + 806) + SED	0.30/0.32/0.13/0.24	0.44	0.53	0.23(0.00)	**0.74(0.06)**
MBA(803 + 806 + 820) + SED	0.29/0.13/0.35/0.23	**0.71**	0.50	0.23(0.00)	0.60(0.27)
MBA(805 + 806 + 820) + SED	0.30/0.12/0.35/0.23	0.55	0.50	0.21(0.00)	0.55(0.30)
Average	0.29/0.21/0.24/0.26	0.56	0.55	0.25	0.69

Table 5 P@k accuracy for STOMP, LOF, IF, and NormA-mn (with the default sampling rate $r = 0.4$) applied to multi-normal datasets. The repartition of anomalies is reported as the percentage of anomalies in each segment.

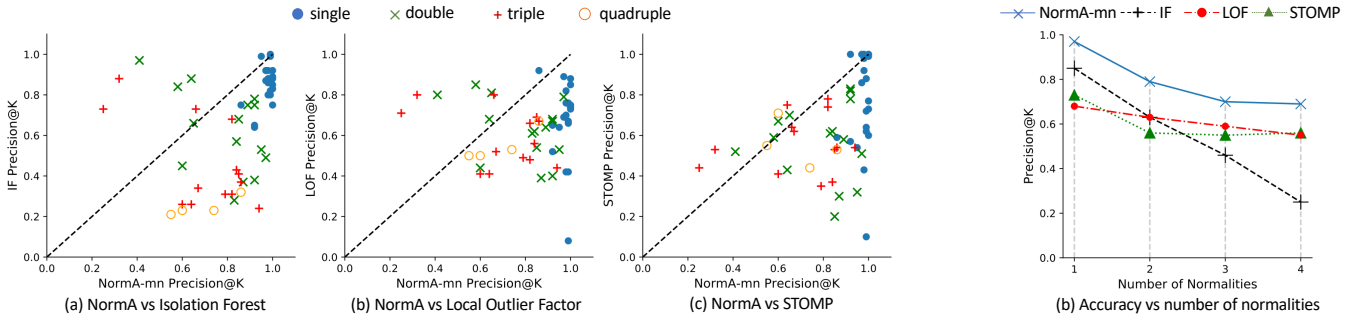
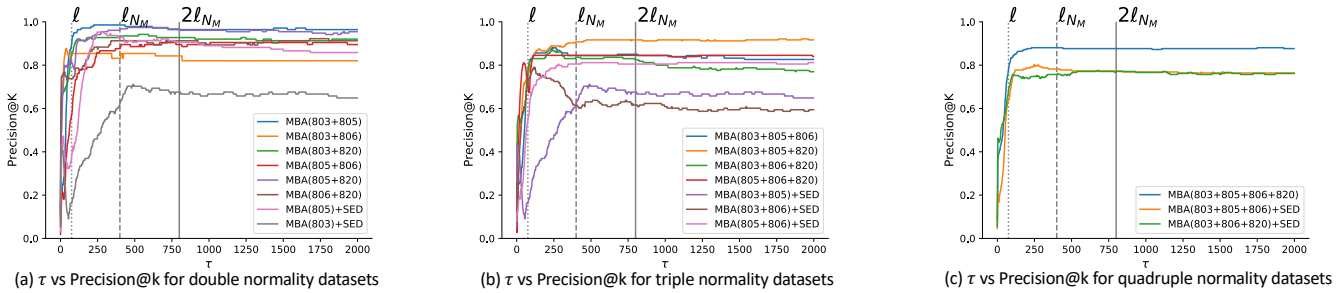


Fig. 11 NormA-mn P@k accuracy versus Isolation forest (a), Local Outlier Factor (b), and STOMP (c). Blue dots represent single normality datasets, green crosses represent double normality, and red crosses represent triple normality datasets. (d) depicts the P@k accuracy evolution for different number of normalities. Each point is the average accuracy for all datasets of the corresponding type (single, double, triple normality).

ity of the datasets. Moreover, Figure 11(d) depicts the average accuracy results for each algorithm as a function of the number of normal behaviors in the dataset. The results demonstrate that the accuracy of all methods decreases as the number of normal behaviors increase and the problem becomes harder, with IF (black dashed line) being the most sensitive of all.

[**Influence of** τ] We also evaluate the influence of parameter τ on the Precision@k of NormA-nm. Remember that in this work, we always use the default value of $\tau = 2\ell_{N_M}$ (see Section 4.3).

Figure 12 depicts the evolution of Precision@k for (a) double, (b) triple and (c) quadruple normality datasets, when we vary τ. As expected, Precision@k is low when τ is very small. In this case, the algorithm considers too



(a) τ vs Precision@k for double normality datasets (b) τ vs Precision@k for triple normality datasets (c) τ vs Precision@k for quadruple normality datasets

Fig. 12 Precision@k for (a) double, (b) triple, (c) quadruple normality datasets as a function of τ (refer to Section 4.3).

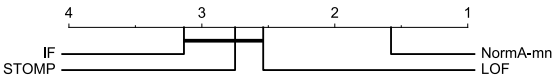


Fig. 13 Critical difference diagram ($\alpha = 0.05$) for the multiple-normality data series of Table 5.

few neighbors to have a representative local sample. We observe a convergence of the Precision@k for values of τ a bit larger than ℓ_{N_M}, which then remains stable as τ increases further.

[Critical Difference Diagram] We once again employ the pairwise Post-Hoc Analysis using a Wilcoxon signed-rank test [59] to test and produce the critical difference diagram for the algorithms and datasets of Table 5. The critical difference diagram with $\alpha = 0.05$ depicted in Figure 13 shows that NormA-mn is significantly better than all competitors.

5.6 Scalability Evaluation

We now present scalability tests (we do not consider LSTM-AD, since supervised methods have a completely different way of operation and associated costs, e.g., data labeling and model training.)

In Figure 14(a), we report the execution time (seconds in log scale) of NormA and all the competitors, when varying the size of the dataset. We use several prefix snippets (*50K, 100K, 500K, 1M, 2M* points) of the real dataset MBA(14406), and we set k equal to the number of anomalies that are annotated in each snippet. We observe that NormA-smpl is 1-2 orders of magnitude faster than the competitors, and gracefully scales with the dataset size. This is because the number of distance calculations performed by NormA-smpl in Algorithm 3 for each subsequence in the data (computation of join sequence) is limited to the subsequences contained in N_M.

NormA, performs a limited number of distance calculations during subsequence clustering (Algorithm 3), since only a small part of subsequences in the input series are selected to be clustered ($\mathbb{S}^{selfjoin}$, or $\mathbb{S}^{sample}$). Thus, NormA-SJ that uses the STOMP algorithm for the Normal Model computation stage, has a small additional time overhead (when compared to STOMP). GV, DAD and LOF adopt different pruning strategies in order to reduce the number of Euclidean distance computations, which prove to be less effective. DAD and LOF, in particular, reach the time-out point (8 hours in our experiments) for datasets $\geq$ 1M points.

In the next set of experiments, we measure the execution time (seconds in log scale) of the algorithms as we vary the number of anomalies; we use the MBA(14406) and instruct the algorithms to find *20,40,60,80,142* anomalies (Figures 14(b)), and the SRW-[20-100]-[0%]-[200] (Figures 14(c)) datasets. In all experiments, the algorithms compute the *Top-k* anomalies. We observe that the time performance of NormA is not influenced by the number of anomalies, since for every subsequence in the dataset we compute anyway the distance to its nearest neighbor in the Normal Model. Similarly, STOMP, IF and LOF enumerate in quadratic time all the *Top-k* 1^{st} *discords*, always consuming the same amount of time. In contrast, the performance of GV and DAD are negatively influenced by the number of anomalies. This confirms that the pruning strategies they use are influenced by the number of anomalies to discover.

Figure 14(d) depicts the time performance results as we vary the length of the anomalies between 100-1600 points (SRW-[60]-[0%]-[100-1600] datasets). The performance of STOMP is constant, because its complexity is not affected by the (anomaly) subsequence length. NormA remains relatively stable, since in Algorithms 2 and 4 the Euclidean distances are computed using the STOMP algorithm. In NormA, only the clustering operations are affected by the length of the subsequences to consider (Algorithm 3), which in all experiments we ran was always a very small number (∼1-2% of all subsequences). We observe that the execution time for NormA-SJ decreases as we move from anomaly length 100 to length 200. This decrease is explained by the reduction of the number of non-overlapping subsequences to cluster, which drops from *242* (anomaly

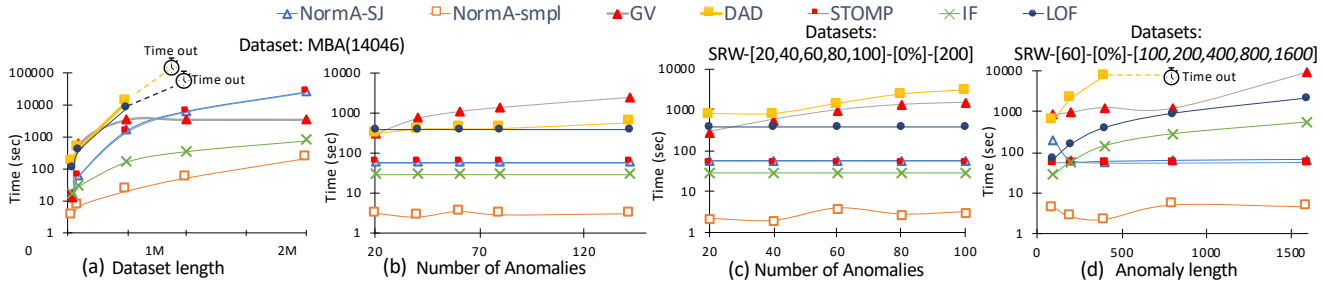


Fig. 14 Scalability: execution time vs *(a)* dataset size, *(b)* number of anomalies for MBA(14406), *(c)* number of anomalies for synthetic, *(d)* anomaly length. Timeout at 8 hours.

length *100*) to *128* (anomaly length *200*). Regarding NormA-smpl, we see a slight fluctuation in execution time, between *1.1-2.4* sec. LOF and IF are computing distances using all overlapping subsequences and the computational time is therefore affected by their length. As shown in Figure 14(d), both of these two methods perform orders of magnitude worse than STOMP and NormA. GV and DAD do not scale with the anomaly length, either.

5.7 NTC Dataset Use Case

We now consider the NTC dataset. As depicted in Figure 15, NormA correctly discovered anomalies that have been reported in earlier studies [6]: Daylight Saving Time (c), Thanksgiving (d), Christmas (e), New Years day (f), and snow storm of January 26-27, 2015 (h). In addition to the above anomalies though, NormA identified additional anomalous subsequences that were not reported by the earlier studies. These anomalies occurred during the Independence Day (a), Labor Day (b), and the bad weather of January 18, followed by the Martin Luther King (MLK) day (January 19) (g) that caused more than 400 accidents and flooding around the NYC area. These three events resulted to unusually low Taxi Traffic in NYC, which was detected by NormA. These results underline the effectiveness of NormA to discover anomalous subsequences.

5.8 Nasa Bearing Dataset Use Case

The Nasa Bearing dataset consists of *984* records of *20,480* points series each, measuring the vibrations of gear bearings. In total, this dataset contains (more than) *20* million points. The goal is to detect the records with failures (anomalous vibrations), which is slightly different than the problem we have considered so far (i.e., subsequence anomaly discovery). We adapt our method by concatenating all records, extracting the Normal Model N_M using subsequences in $\mathbb{S}^{sample}$.

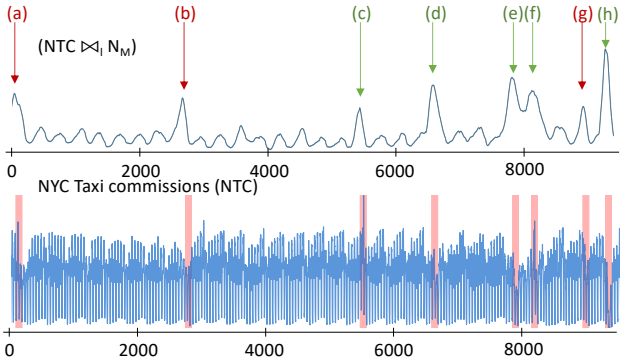


Fig. 15 NormA results on the NTC dataset. *(top)* The join with the Normal Model. Events in green (c,d,e,f,h): anomalies discovered by NormA and earlier studies. Events in red (a,b,g): new anomalies discovered by NormA. *(bottom)* NTC data series with the anomalies marked in red.

We then score anomalies by considering the join of each record R with the Normal Model and compute the average of the join. Summing up the Euclidean distances from the record subsequences to the ones in the Normal Model, permits to quantify the degree of anomalous activity of the record.

In Figure 16, we plot the series of the scores of all records in the Nasa Bearing dataset. Given C, the concatenation of records that do not contain anomalies (in our case the first 400 records), we set a threshold $T = \mu(C) + 3 * \sigma(C)$ (i.e., 3 standard deviations away from the mean), as commonly used in statistics to mark outliers. Based on the results of the analysis by Safran [51] and other experts [7], both of which are based on application-specific algorithms and make heavy use of domain knowledge, the failures start at record 534. NormA detects failures starting at record 533. These results demonstrate again the versatility of NormA, which successfully identifies anomalies in an unsupervised manner and no domain knowledge.

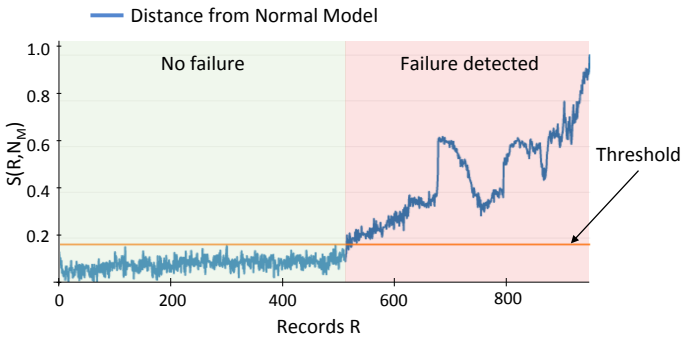


Fig. 16 NormA on the Nasa Bearing Dataset. In blue, data series S composed of all records R anomaly scores. In orange, the threshold computed on the first 400 records. When S is above *Threshold*, we flag a failure.

6 Related Work

The problem of subsequence anomaly discovery has been studied by several works that use the *discord* definition [64,52,27,37,21,17,38]. In these studies, anomalies are considered the isolated subsequences, that is, the ones that have the highest Euclidean distances to their NNs. In practice, these approaches (that are based on the *discord* definition) fail when the dataset contains multiple anomalies that are similar to one another. The notion of m^{th} *discord* has been proposed in order to resolve the problem of multiple similar anomalies [62]. The approach described in this study finds the sequence that has the farthest m^{th} NN in Euclidean space. During the search, a space pruning strategy based on the intermediate results of the simple *discord* discovery is applied. As we have already discussed, the m^{th} *discord* definition fixes the main problem of simple *discord* but is very sensitive to the m parameter, can lead to false positives, and is not scalable. NormA avoids all these shortcomings, because it is based on a new, different primitive for identifying anomalies.

Several methods have been proposed for efficient and scalable similarity (and nearest neighbor) search [43, 48,47,29,33,19,20], which can be used for subsequence anomaly detection. Nevertheless, even though such methods have the potential to speed up *discord*-based techniques (like the ones described above), they will not remove the drawbacks of the *discord* definition we have discussed in this work.

Wang et al. [57] proposed a framework for mining anomalies of different lengths. However, their algorithm (SLADE-TS) can only be applied in the specific context of a collection of several series, which need to be aligned and periodic. This requirement allows them to identify anomalies based on the behavior of the rest of the sequences, but cannot be applied in the case of subsequence anomaly detection in a single series, which is the focus of our work.

In multi-dimensional data, the Local Outlier Factor [14] is the degree of being an outlier assigned to each data instance, depending on how distant a data instance is from other points in its neighborhood. Similarly, Isolation Forest [36] is a machine learning technique that isolates anomalies instead of modeling normality. It first proceeds on building binary trees with random splitting nodes to partition the dataset. The anomaly score is defined as a function of the averaged path length between a particular sample and the root of the trees.

In outlier trajectory detection (than can be seen as a special kind of data series and a sub task of subsequence anomaly detection), relevant approaches have been proposed [30,16,65]. These approaches partition trajectories into smaller parts, cluster the resulting trajectories, and identify outlier trajectories with respect to these clusters (taking advantage of both distance-based and density-based approaches). We note that these methods identify as outliers individual trajectories within a trajectory dataset (following the partitioning phase), while in our case, we want to detect an anomalous subsequence within a single long series.

LSTM-AD [39] is a *supervised* subsequence anomaly detection algorithm, and as such not directly comparable to our (unsupervised) approach. LSTM-AD first trains an LSTM neural network using the data segments that do not contain anomalies, and then forecasts the values in the series: when the error between the forecast and the real value is above some threshold, the subsequence is classified as an anomaly. LSTM-AD learns the threshold in the validation set, picking the value that maximizes the F-score of the classification. The LSTM model has also been used in a zero positive learning framework, where the annotated anomalies are not necessary for the training phase [31]. The major drawback of this approach is that it is supervised, requiring a large amount of clean, normal data for training. In practice, this is not always possible to do.

7 Conclusions

Even though the problem of anomaly detection in data series has attracted lots of attention, the techniques that have been proposed so far fall short in terms of effectiveness and efficiency. In our work, we describe a novel approach that is based on the representation of normal behavior, which enables us to detect both single and recurrent anomalies, irrespective of the domain, and leads to superior accuracy and time perfor-

mance. As part of future work, we plan to study alternative ways for computing the Normal Model, as well as compare to the recently proposed Series2Graph approach [46,12,13].

References

1. http://data-acoustics.com/measurements/bearing-faults/bearing-4/ (2007)
2. http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml (2015)
3. https://tinyurl.com/ycbhkv6x (2019)
4. Abboud, D., Elbadaoui, M., Smith, W., Randall, R.: Advanced bearing diagnostics: A comparative study of two powerful approaches. MSSP **114** (2019)
5. Abdul-Aziz, A., Woike, M.R., Oza, N.C., Matthews, B.L., lekki, J.D.: Rotor health monitoring combining spin tests and data-driven anomaly detection methods. Structural Health Monitoring (2012)
6. Ahmad, S., Lavin, A., Purdy, S., Agha, Z.: Unsupervised real-time anomaly detection for streaming data. Neurocomputing (2017)
7. Antoni, J., Borghesani, P.: A statistical methodology for the design of condition indicators. Mechanical Systems and Signal Processing (2019)
8. Bagnall, A.J., Cole, R.L., Palpanas, T., Zoumpatianos, K.: Data series management (dagstuhl seminar 19282). Dagstuhl Reports (9(7), 2019)
9. Barnet, V., Lewis, T.: Outliers in Statistical Data. John Wiley and Sons (1994)
10. Boniol, P., Linardi, M., Roncallo, F., Palpanas, T.: Automated Anomaly Detection in Large Sequences. In: ICDE (2020)
11. Boniol, P., Linardi, M., Roncallo, F., Palpanas, T.: SAD: an unsupervised system for subsequence anomaly detection. In: 36th IEEE International Conference on Data Engineering, ICDE, pp. 1778–1781. IEEE (2020)
12. Boniol, P., Palpanas, T.: Series2graph: Graph-based subsequence anomaly detection for time series. Proc. VLDB Endow. **13**(11), 1821–1834 (2020)
13. Boniol, P., Palpanas, T., Meftah, M., Remy, E.: Graphan: Graph-based subsequence anomaly detection. Proc. VLDB Endow. **13**(12), 2941–2944 (2020)
14. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: Identifying density-based local outliers. In: SIGMOD (2000)
15. Bryant, P.G.: On the minimum description length (mdl) principle for hierarchical classifications. In: Data Science, Classification, and Related Methods (1998)
16. Bu, Y., Chen, L., Fu, A.W.C., Liu, D.: Efficient anomaly monitoring over moving object trajectory streams. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, p. 159–168. Association for Computing Machinery, New York, NY, USA (2009). DOI 10.1145/1557019.1557043. URL https://doi.org/10.1145/1557019.1557043
17. Bu, Y., Leung, O.T., Fu, A.W., Keogh, E.J., Pei, J., Meshkin, S.: WAT: finding top-k discords in time series database. In: SIAM (2007)
18. Chiu, B.Y., Keogh, E.J., Lonardi, S.: Probabilistic discovery of time series motifs. In: KDD (2003)
19. Echihabi, K., Zoumpatianos, K., Palpanas, T., Benbrahim, H.: The Lernaean Hydra of Data Series Similarity Search: An Experimental Evaluation of the State of the Art. PVLDB (2018)
20. Echihabi, K., Zoumpatianos, K., Palpanas, T., Benbrahim, H.: Return of the Lernaean Hydra: Experimental Evaluation of Data Series Approximate Similarity Search. PVLDB (2019)
21. Fu, A.W., Leung, O.T., Keogh, E.J., Lin, J.: Finding time series discords based on haar transform. In: ADMA (2006)
22. Gharghabi, S., Yeh, C.M., Ding, Y., Ding, W., Hibbing, P., LaMunion, S., Kaplan, A., Crouter, S.E., Keogh, E.J.: Domain agnostic online semantic segmentation for multi-dimensional time series. Data Min. Knowl. Discov. **33**(1), 96–130 (2019)
23. Goldberger, A.L., Amaral, L.A.N., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.K., Stanley, H.E.: PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation **101**(23), e215–e220 (2000 (June 13)). Circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215
24. Grabocka, J., Schilling, N., Schmidt-Thieme, L.: Latent time-series motifs. TKDD **11**(1), 6:1–6:20 (2016)
25. Hadjem, M., Naït-Abdesselam, F., Khokhar, A.A.: St-segment and t-wave anomalies prediction in an ECG data using rusboost. In: Healthcom (2016)
26. Keogh, E., Lin, J.: Clustering of time-series subsequences is meaningless: implications for previous and future research. KAIS **8**(2) (2004)
27. Keogh, E., Lonardi, S., Ratanamahatana, C., Wei, L., Lee, S.H., Handley, J.: Compression-based data mining of sequential data. DMKD (2007)
28. Keogh, E.J., Lin, J., Fu, A.W.: HOT SAX: efficiently finding the most unusual time series subsequence. In: ICDM (2005)
29. Kondylakis, H., Dayan, N., Zoumpatianos, K., Palpanas, T.: Coconut: sortable summarizations for scalable indexes over static and streaming data series. VLDBJ **28**(6) (2019)
30. Lee, J., Han, J., Li, X.: Trajectory outlier detection: A partition-and-detect framework. In: 2008 IEEE 24th International Conference on Data Engineering, pp. 140–149 (2008)
31. Lee, T., Gottschlich, J., Tatbul, N., Metcalf, E., Zdonik, S.: Greenhouse: A zero-positive machine learning system for time-series anomaly detection. CoRR **abs/1801.03168** (2018). URL http://arxiv.org/abs/1801.03168
32. Li, X., Lin, J.: Linear time motif discovery in time series. In: Proceedings of the 2019 SIAM International Conference on Data Mining, pp. 136–144. SIAM (2019)
33. Linardi, M., Palpanas, T.: Scalable, variable-length similarity search in data series: The ulisse approach. PVLDB (2019)
34. Linardi, M., Zhu, Y., Palpanas, T., Keogh, E.: Matrix profile x: Valmod - scalable discovery of variable-length motifs in data series. In: SIGMOD (2018)
35. Linardi, M., Zhu, Y., Palpanas, T., Keogh, E.J.: Matrix Profile Goes MAD: Variable-Length Motif And Discord Discovery in Data Series. In: DAMI (2020)
36. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: ICDM, ICDM (2008)
37. Liu, Y., Chen, X., Wang, F.: Efficient Detection of Discords for Time Series Stream. Advances in Data and Web Management (2009)
38. Luo, W., Gallagher, M.: Faster and parameter-free discord search in quasi-periodic time series. In: Advances in Knowledge Discovery and Data Mining (2011)

39. Malhotra, P., Vig, L., Shroff, G., Agarwal, P.: Long short term memory networks for anomaly detection in time series. In: ESANN (2015)
40. Moody, G.B., Mark, R.G.: The impact of the mit-bih arrhythmia database. IEEE Engineering in Medicine and Biology Magazine (2001)
41. Mueen, A., Keogh, E.J., Zhu, Q., Cash, S., Westover, M.B.: Exact discovery of time series motifs. In: SDM (2009)
42. Palpanas, T.: Data series management: The road to big sequence analytics. SIGMOD Rec. **44**(2), 47–52 (2015)
43. Palpanas, T.: Evolution of a Data Series Index. CCIS (2020)
44. Palpanas, T., Beckmann, V.: Report on the first and second interdisciplinary time series analysis workshop (ITISA). SIGREC **48**(3) (2019)
45. Paparrizos, J., Gravano, L.: K-shape: Efficient and accurate clustering of time series. SIGMOD Rec. **45**(1), 69–76 (2016). DOI 10.1145/2949741.2949758. URL `https://doi.org/10.1145/2949741.2949758`
46. Paul Boniol (advisor: Themis Palpanas): Unsupervised subsequence anomaly detection in large sequences. In: Proceedings of the VLDB 2020 PhD Workshop co-located with the 46th International Conference on Very Large Databases (VLDB 2020), *CEUR Workshop Proceedings*, vol. 2652 (2020)
47. Peng, B., Palpanas, T., Fatourou, P.: Messi: In-memory data series indexing. In: ICDE (2020)
48. Peng, B., Palpanas, T., Fatourou, P.: Paris+: Data series indexing on multi-core architectures. TKDE (2020)
49. Rakthanmanon, T., Keogh, E.J., Lonardi, S., Evans, S.: Time series epenthesis: Clustering time series streams requires ignoring some data. In: 2011 IEEE 11th International Conference on Data Mining, pp. 547–556 (2011)
50. Rissanen, J.: Modeling by shortest data description. Automatica (1978)
51. Safran: Personal communication with Dr. Dohy Hong (2018)
52. Senin, P., Lin, J., Wang, X., Oates, T., Gandhi, S., Boedihardjo, A.P., Chen, C., Frankenstein, S.: Time series anomaly discovery with grammar-based compression. In: EDBT (2015)
53. Senin, P., Lin, J., Wang, X., Oates, T., Gandhi, S., Boedihardjo, A.P., Chen, C., Frankenstein, S.: Grammarviz 3.0: Interactive discovery of variable-length time series patterns. TKDD (2018)
54. Shieh, J., Keogh, E.: iSAX: disk-aware mining and indexing of massive time series datasets. DMKD (2009)
55. Subramaniam, S., Palpanas, T., Papadopoulos, D., Kalogeraki, V., Gunopulos, D.: Online outlier detection in sensor data using non-parametric models. In: VLDB (2006)
56. Wang, J., Balasubramanian, A., de la Vega, L.M., Green, J., Samal, A., Prabhakaran, B.: Word recognition from continuous articulatory movement time-series data using symbolic representations. In: SLPAT
57. Wang, X., Lin, J., Patel, N., Braun, M.: A self-learning and online algorithm for time series anomaly detection, with application in CPU manufacturing. In: CIKM (2016)
58. Whitney, C., Gottlieb, D., Redline, S., Norman, R., Dodge, R., Shahar, E., Surovec, S., Nieto, F.: Reliability of scoring respiratory disturbance indices and sleep staging. Sleep (1998)
59. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin **1**(6), 80–83 (1945). URL `http://www.jstor.org/stable/3001968`
60. Wu, Q., Qi, X., Fuller, E., Zhang, C.Q.: Follow the leader: A centrality guided clustering and its application to social network analysis. The Scientific World Journal (2013)
61. Yankov, D., Keogh, E., Rebbapragada, U.: Disk aware discord discovery: Finding unusual time series in terabyte sized datasets. In: ICDM (2007)
62. Yankov, D., Keogh, E., Rebbapragada, U.: Disk aware discord discovery: finding unusual time series in terabyte sized datasets. KAIS **17**(2) (2008)
63. Yankov, D., Keogh, E.J., Medina, J., Chiu, B.Y., Zordan, V.B.: Detecting time series motifs under uniform scaling. In: KDD (2007)
64. Yeh, C., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H., Silva, D., Mueen, A., Keogh, E.: Matrix profile I: all pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In: ICDM (2016)
65. Yu, Y., Cao, L., Rundensteiner, E.A., Wang, Q.: Outlier detection over massive-scale trajectory streams. ACM Transactions on Database Systems (TODS) **42**, 1 – 33 (2017)
66. Zhu, Y., Zimmerman, Z., Senobari, N.S., Yeh, C.M., Funning, G., Mueen, A., Brisk, P., Keogh, E.: Matrix profile ii: Exploiting a novel algorithm and gpus to break the one hundred million barrier for time series motifs and joins. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 739–748 (2016). DOI 10.1109/ICDM.2016.0085