An Experiment Evaluation of Anomaly Detection in Time Series[Experiment, Analysis & Benchmark]

Aoqian Zhang, Shuqing Deng, Dongping Cui, Ye Yuan, Guoren Wang
Beijing Institute of Technology, China
{aoqian.zhang,shuqing.deng,dongping.cui,yuan-ye,wgr}@bit.edu.cn

ABSTRACT

Anomaly detection for time series data has been studied for decades in both statistics community and computer science community. Various algorithms have been proposed to work in different scenarios. However, there is a lack of comparative evaluation of these state-of-the-art approaches, especially under the same test environment with the same benchmark, making it hard for users to select a proper method in real-world applications. In this work, we classify anomaly detection methods into three facets consisting of six classes and empirically compare nine representative methods in our testbed over fourteen real-world and more than ten synthetic datasets. In addition to intra-class comparisons, we also provide inter-class comparisons and observe non-trivial and interesting findings, such as point anomaly methods can also perform well in datasets with subsequence anomaly. We propose a practical guide for anomaly detection and suggest future research directions based on these findings.

PVLDB Reference Format:
Aoqian Zhang, Shuqing Deng, Dongping Cui, Ye Yuan, Guoren Wang. An Experiment Evaluation of Anomaly Detection in Time Series[Experiment, Analysis & Benchmark]. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/zaqthss/experiment-tsad.

1 INTRODUCTION

Time series is one of the most widely used data types in the current decades. Time series analysis comprises methods for analyzing time series data to extract valuable knowledge from the data. Anomaly detection aims to find rare observations which deviate significantly from the majority of the data [12] and is the major part of time series analysis (mining). It has been studied in various applications, such as fraud detection in financial markets [5], alert in environment monitoring [9] and industrial manufacturing [6], identification of cyber-attacks [21] etc.

Unfortunately, an anomaly is rather challenging to define, especially in the context of time series data [29]. It is hard to evaluate an existing anomaly detection method. Due to the following reasons, it is difficult for users to choose a proper algorithm (with appropriate parameters) to handle the anomaly cases in real scenarios.

The first reason is the complexity and diverse nature of time series data [16]. Time series data can be univariate and multivariate, and the latter may possess many dimensions. In addition, there are many anomaly types without a clear classification. Various algorithms have been proposed to detect anomaly behaviors using different techniques in recent studies. However, these algorithms often focus on a specific case. For example, [14] is only able to find anomaly data points in a univariate time series naturally. [3] is capable of univariate data steams but can only detect anomaly patterns of a certain length. Thus, after analyzing the state-of-the-art anomaly detection algorithms, *we extract their key features and classify them into three facets, i.e., data dimension, processing technique, and anomaly type*. We can further divide them into sub-classes under a particular facet, as shown in Figure 1, to help make a clear intra-class comparison in a specific domain.

The second reason is that existing experiments always focus on intra-class comparisons but leave inter-class comparisons only in the theory analysis stage. For example, univariate detection algorithms are only compared to other univariate methods. Algorithms that focus on point anomalies are never applied to time series data with subsequence anomalies. However, *we propose inter-class comparisons to understand existing methods better*. For instance, what will a univariate perform if we run it on each data dimension of a multivariate dataset? Will the potential correlations among dimensions in a multivariate time series affect the final result?

The third reason is a lack of thorough tests using the same datasets and the same platform with the same metrics. Each existing method provides an experimental comparison in the proposed paper with different metrics. The compared baselines may be written in a different language or use different data structures, leading to unfair comparison. *A systematic comparison of various datasets can help us find clearer results other than pure statements.*

As mentioned above, in our work, we classify and re-implement nine state-of-the-art unsupervised algorithms (shown in Table 1) and propose a systematic empirical study in our test-bed using real-world and synthetic benchmarks. To the best of our knowledge, this is the first in-depth experimental study providing both *intra-class and inter-class* evaluations.

Contributions. Our major contributions in this paper are summarized as follows.

(1) We classify anomaly detection methods in time series based on their key features into three different facets and six inner classes.
(2) We create a common test-bed for comparative intra-class and inter-class evaluation of algorithms under different scenarios

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

using real-world and synthetic datasets. We evaluate the precision/recall, efficiency, and sensitivity of critical parameters for each algorithm.
(3) We re-implement the algorithms with JAVA and build an extendable data structure to make the comparison fair. The source code [1] is available to the research community.
(4) We provide a practical guide for choosing a proper anomaly detection method under a given scenario.

Highlight Results. We list our major nontrivial findings in the study as follows:

- For time series data, some dimensions can have a larger impact on anomaly detection. NETS utilizes such sub-dimension and have slight different results from CPOD.
- Distance-based methods can still get high accuracy under a number of dimensions. The sparsity problem can be coped with by a good parameter-search, shown in Section 4.3.4
- Running algorithms on each dimension of a multivariate time series and union of the results together can promote the recall, which can reduce the miss alert cases, shown in Section 4.4.1.
- For the online method, the trade-off of accuracy and efficiency is not trivial over window/slide size. Larger window size and slide size does not necessarily get better results, though they cost more time. That is due to the streaming feature of time series and the potential existence of change point, shown in Section 4.4.2.
- Point anomaly methods can also perform well under subsequence anomaly case, which may help get rid of the requirement of anomaly length or adaptively set it, shown in Section 4.4.3.

Related Work. An experimental analysis over distance-based outlier detection in data streams is proposed in [30]. However, only the efficiency factors such as CPU time and leak memory are evaluated under different parameters. [18] revisits the outlier definitions and types but provides only the intra-class comparisons varying different anomaly rates. Our study compares algorithms in a wider variety and evaluates accuracy and efficiency.

There also exist various surveys on anomaly detection for time series data. [11] introduces methods upon the specific data type and scenario. [8] shows a wide variety of IoT (Internet of Things) applications and challenges faced by anomaly detection. [4] focuses methods in univariate time series while [2] presents introduction of outlier detection techniques based on outlier type. However, these works consider theory analysis, but we focus on the interesting insights from experimental results.

Recently, approaches using various deep neural network architectures have been proposed for outlier detection for time series data. [7] presents a structured and comprehensive overview of research methods in deep learning-based anomaly detection. We do not involve these deep learning techniques in our study for two main reasons: 1) Most of them are supervised or semi-supervised, which is not fair to compare with the unsupervised algorithms in this paper. 2) Some of their results are not easy to explain or explicitly shown [21].

[1] https://github.com/zaqthss/experiment-tsad

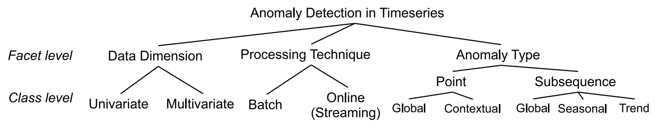


Figure 1: Facets and classes of anomaly detection algorithms in time series data

The original papers that propose anomaly detection algorithms compared in our paper [3, 10, 13–15, 19, 31, 33, 34] consist empirical comparison as well. However, these experiments are limited in terms of datasets, competitors and a thorough analysis over precision/recall, efficiency and sensitivity compared to our study. We use datasets from these papers and benchmark provided by Numenta [19], Exathlon [16] and SEQ [18] as well.

2 PRELIMINARIES

2.1 Problem Definition

We first introduce some key definitions about anomaly detection in time series.

Definition 1 (Time series). *A time series is a series of data points indexed in time order. Consider a time series of n observations, $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. The i-th observation (data point) $\boldsymbol{x}_i \in \mathbb{R}^D$ consists of D dimensions $\{x_i^1, \ldots, x_i^D\}$ and is observed at timestamp t_i. A subsequence $\boldsymbol{X}_{i,\ell} = \{\boldsymbol{x}_i, \ldots, \boldsymbol{x}_{\ell-i+1}\}$ is a subset of consecutive time points from time series $\boldsymbol{X}$ starting at position i with length ℓ.*

Definition 2 (Anomaly Detection in Time Series). *Anomaly is generally understood to be rare items, events or observations which deviate significantly from the majority of the data. Anomaly detection in time series is to find the unusual data points $\boldsymbol{x}_i$ or subsequences $\boldsymbol{X}_{i,\ell}$ in the given time series.*

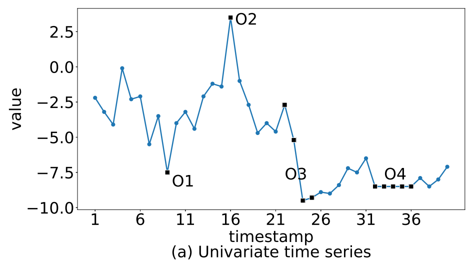
Figure 2(a) illustrates a time series with four anomalies where $O1$ and $O2$ are data points while $O3$ and $O4$ are subsequences consist of four and five consecutive data points, respectively.

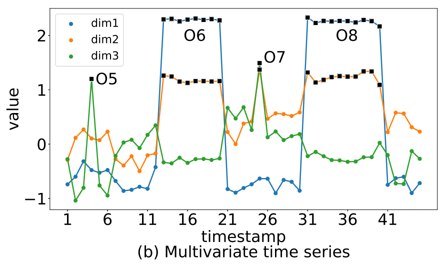
2.2 Analysis Facets

As suggested in [11], the anomaly analysis problems can be categorized in various ways, which represent different facets of the analysis. In this work, we try to analyze the anomaly detection problem from the perspective of algorithm classes. Here, we categorize these algorithms into three facets, i.e., data dimension, processing technique, and anomaly type, which are shown in Figure 1. For each facet, we further divide the detection algorithms into classes. The details of these facets and classes will be introduced in this section, and experimental results in the same class and across different classes (under the same facet) can be found in Section 4.

2.2.1 Data dimension. The first facet is the data dimension. Some algorithms aim to process univariate time series while others are able to handle multivarate time series. Following Definition 1, $\boldsymbol{X}$ is called as a **univariate** time series if $D = 1$ while $\boldsymbol{X}$ is a **multivariate** time series if $D > 1$.

Example 1. *A* ***univariate*** *and a* ***multivariate*** *time series with three dimensions are shown in Figures 2(a) and (b), respectively. It is*





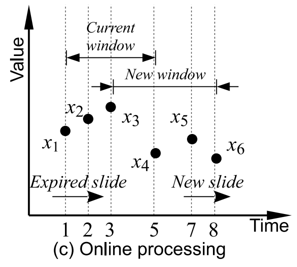


Figure 2: Examples of (a) univariate time series (b) multivariate time series and (c) online processing technique

noted that, for multivariate time series, if there exists anomaly in any of its dimension(s), such data point/subsequence should be identified as anomaly. Thus, $O5$ (anomaly exists only in dimension 3) and $O7$ (exist in both dimension 1 and 2) are considered as anomalies.

Univariate detection methods can be applied by detecting the anomalies dimension by dimension and finally union all the results on a multivariate time series. In contrast, multivariate detection methods can take the whole series as an input. Though we consider the time series as a series of points, there are correlations in the time dimension and among feature dimensions. In theory, multivariate detection methods can advantage such relationships among dimensions, leading to better detection accuracy. Thus, we wonder whether multivariate methods always win univariate ones under various cases.

2.2.2 Processing Techniques. Time series (or data stream) can also have an infinite number of data points. Algorithms that are capable of processing the infinite time series are called **online** algorithms while those that require a whole data are called **batch** algorithms. The sliding window is the most commonly used technique by online algorithms, that processes data points window by window.

In theory, batch algorithms will have a higher accuracy due to their ability to compare all the data points simultaneously. In contrast, online algorithms can run much faster since they only process data points in the current window, far smaller than the whole data. Thus, *we are interested in the trade-off between the effectiveness and efficiency of these two types of algorithms under various scenarios.*

Example 2. *As shown in Figure 2(c), the data points in univariate time series $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ are observed at time $1, 2, 3, 5, 7, 8$, respectively. Let us set the window size $\theta_W = 4$ and slide size $\theta_S = 2$. The online algorithm will first process the data points $\{x_1, x_2, x_3, x_4\}$ in the current window. After the processing, $\{x_1, x_2\}$ in the first slide will be expired and $\{x_5, x_6\}$ in the new slide will come in, forming a new window containing $\{x_3, x_4, x_5, x_6\}$ that to be processed.*

2.2.3 Anomaly Types. Given a time series, one can find particular data points as anomalies, i.e., differ significantly from other observations. These data points are considered **point** anomalies. Subsequences (consecutive points) of the time series with unusual behavior compared to other subsequences are identified as **subsequence** anomalies. Most anomaly detection methods can only detect a specific type of anomaly.

Inspired by [18], we focus on the *Global* and *Contextual* anomaly patterns for point anomaly. The former pattern indicates that the identified anomalies are significantly different from the other points. The latter means that the reported anomalies are different from their neighbors in a specific time range. As for subsequence anomalies, *Global*, *Seasonal* and *Trend* anomaly patterns are considered. *Global* means that the basic shapes of the subsequence are different. *Seasonal* refers to the subsequence with unusual seasonality compared to the whole series. *Trend* indicates the subsequence that significantly alters the trend of the time series, leading to a permanent shift in the mean of the data.

Example 3. *Point anomalies can be univariate or multivariate. Figure 2(a) presents two univariate point anomalies, $O1$ and $O2$, while Figure 2(b) shows one univariate point anomaly $O5$ and a multivariate one $O7$. According to the above description, $O1$ is a contextual anomaly and $O2$ is a global anomaly.*

Similarly, subsequence anomalies can also be univariate (like $O3$ and $O4$) and multivariate (like $O5$ and $O7$). As for patterns, $O3$ is a trend anomaly since it leads a permanent shift with a lower mean, $O4$ is a seasonal anomaly and $O6$, $O8$ are global anomalies, respectively. It is noted that individual points in the subsequence anomaly may not be point outliers (see points in $O4$), they only form an anomaly subsequence when they gather together.

State-of-the-art subsequence methods require the *length* of the target subsequence as input. However, it is pretty hard to determine the length given a time series without sufficient domain knowledge. On the contrary, methods that focus on point anomalies do not need such extra input (the length is 1). *We wonder 1) what the performance point methods will have on subsequence anomaly; 2) the sensitivity of the parameter length; 3) the capability of algorithms to detect various patterns of anomalies.*

3 ALGORITHMS

In addition to the analysis facets, algorithms can also be classified by their detection techniques. Here we briefly introduce them by distance-based and pattern-based categories. Table 1 categorizes each algorithm by data dimension, process type, and anomaly type. The "threshold" column indicates how to set the anomaly threshold. We will then describe these algorithms from aspects of overview, slide processing (for online methods)/data processing (for batch methods), and outlier reporting in the following.

Table 1: Anomaly detection algorithms considered in empirical comparison

	Algorithm	Mul	Process	Anomaly	Threshold	Code	Speedup
Distance	CPOD [31]	✓	online	point	θ_k	-	-
	NETS [33]	✓	online	point	θ_k	JAVA	-
	STARE [34]	✓	online	point	top-θ_K	JAVA	-
	NP [13]	-	batch	subsequence	top-θ_K	python	1.5
Pattern	PBAD [10]	✓	batch	subsequence	top-θ_K	python	2.5
	LRRDS [15]	✓	batch	subsequence	cluster	r	1
	SAND [3]	-	online	subsequence	top-θ_K	python	0.3
	Luminol [19, 25]	-	batch	point	top-θ_K	python	28.75
	SHESD [14]	-	batch	point	θ_k, top-θ_K	JAVA	-

3.1 Distance-based Algorithms

Distance-based algorithms are the majority of the literature [30]. Such a technique can detect both point and subsequence anomalies, but we will only mention point anomaly in the following definitions for simplicity.

Definition 3 (Neighbor). *Given a distance threshold θ_R, a data point x_i is a neighbor of data point $x_j (x_i \neq x_j)$ if the distance $d(x_i, x_j) \leq \theta_R$.*

Definition 4 (Distance-based Outlier). *Given a time series X, a count threshold θ_k and a distance threshold θ_R, distance-based outliers in X are set of data points that have less than θ_k neighbors.*

The critical factors of distance-based algorithms are distance threshold θ_R and count threshold θ_k. We can compute neighbors for each data point in the dataset with such factors and finally find the outliers. For online methods, the window size θ_W and the slide size θ_S factors will affect the performance, and they are usually set according to the size of the dataset.

3.1.1 *CPOD.*

CPOD [31] uses the core point structure to speed up neighbor searches. A core point is a special data point that stores its distances to other data points. Each data point is linked to at least one core point and divided into four different lists, i.e., $\{\theta_R/2, \theta_R, 3/2\theta_R, 2\theta_R\}$, according to the distances between the core point, respectively. Following triangle inequality of distances among data points, the search range is reduced.
Slide processing. CPOD first processes the expired slide and decreases the related neighbor counts. It then finds a supporting core point set for data points in the new slide and forms the data list linked to each core point. For the data points that do not have enough neighbors (less than θ_k), neighbor searches are executed.
Outlier reporting. Outlier candidates who still do not have enough neighbors after neighbor search will be reported as anomalies.

3.1.2 *NETS.*

NETS [33] employs a set-based approach which groups the data points at similar locations into a set A cell is a hyper-cube in a high-dimensional space formed by a grid-based index structure. The diagonal of each cell is θ_R, making all the data points in one cell become neighbors of each other.
Slide processing. NETS processes the expired and new slides concurrently and maintains the cardinality of each grid cell. The cardinality of grid cells containing data points in the expired and new slides are updated when the window slides. After that, the net effect is calculated by subtracting the changed cardinalities.
Outlier reporting. A cell can be categorized as an outlier cell if all the data points in it are outliers. NETS performs neighbor searches to find all neighbors of the data points in a non-determined cell. Outliers detected on the cell and point level are reported.

3.1.3 *STARE.*

STARE [34] is a density-based method based on the observation that in many regions of the data space, data distributions hardly change across window slides. Thus, it skips updating the densities in those local regions, namely stationary regions, by implementing data distribution approximation and cumulative net-change-based skip techniques.
Slide processing. STARE approximates the data space into a set of fixed small regions. When a new slide arrives, it firstly keeps track of the change by counting the number of data points (weight) in each region. Secondly, it examines the weights changes between consecutive sliding windows and identifies stationary regions where the local densities estimated in previous windows are reused. Thirdly, it updates the density of other regions.
Outlier reporting. The data point that has a relatively lower density than its neighbors is identified as an outlier. Similar to NETS, with the computed densities of all the regions, STARE takes the top-θ_K outliers on the cell and point level.

3.1.4 *NP.*

NP [13] uses bagging to discover frequent and rare subsequences from time series robustly. It addresses unsolved issues in *matrix profile* framework [32] by leveraging bagged nearest neighbors and estimates each subsequence density from multiple subsamples by averaging.
Data processing. NP firstly learns models from multiple sub-samples. The nearest neighbor ball is proposed to adjust the nearest neighbor distance for robust estimation. When a subsequence falls inside the nearest neighbor ball, its nearest neighbor distance will be automatically adjusted as the radius of such ball. After aggregating the density estimated from each sub-sample, the frequent/rare subsequences can be found according to the learned neighbor profile.
Outlier reporting. To be consistent with *matrix profile*, NP takes *np* to represent the aggregated density where larger *np* indicates lower estimated density. As a result, subsequences with the top θ_K largest *np* are reported as anomalies.

3.2 Pattern-based Algorithms

The so-called pattern represents the regularities in data, such as an ordered set of data points occurring in the data frequently or a particular distribution. Pattern-based algorithms are usually proposed to find subsequence anomalies which we use in the definitions below. (Point anomalies can be found if the pattern indicates a distribution.)

Definition 5 (Pattern-based Outlier). *Given a time series X, an anomaly length ℓ, pattern-based outliers in X are set of subsequences that dissimilar with the patterns extracted from X.*

The factors for pattern-based algorithms vary by how they define patterns, the support and the anomaly threshold. In general, the key factors are the support threshold θ_{sup}, an anomaly threshold θ_ϵ or a count threshold θ_K, representing the minimum occurrence of a pattern, the minimum/maximum score of an anomaly will have or the rank of the scores an anomaly will reach, respectively. The most time-consuming part is the extraction of the pattern, and thus those factors are carefully set.

3.2.1 PBAD. PBAD [10] can detect subsequence anomalies in mixed-type time series that consist of both continuous values and discrete event logs. Under such a structure, the information in both types of data can be embedded into a single feature vector that can be fit in classifiers to get anomaly scores.

Data processing. PBAD firstly normalizes and divides the data into windows and gets discrete representations. Secondly, the closed or maximal frequent itemsets and sequential patterns are minded from each dimension (whose support is larger than θ_{sup}). Thirdly, the distance-weighted similarity scores between the frequent patterns and each window are computed to construct a pattern-based embedding. Such embedding is the feature-vector representation of the window formed by the concatenation of the similarity scores overall dimensions.

Outlier reporting. The Isolation Forest classifier takes the feature-vector as input. We consider the length of the tree path as the anomaly score. The anomaly threshold is set to a specific percentile of the distribution of all scores. All subsequences with scores higher than θ_{ϵ} are reported as anomalies.

3.2.2 LRRDS. LRRDS [15] identifies the anomalies with a recurrence plot on multivariate time series based on the fact that the most unusual subsequences tend to have local regions of significantly different densities than normal ones. It reduces the dimensionality of the time series and allows the detection of variable-length subsequence anomalies.

Data processing. Firstly, a recurrence plot of the original time series is generated by the embedding technique. Secondly, the local recurrence rate (LREC) is used to represent the density of the local region in the recurrence plot, and the LREC curve is formed. Change points are then identified as points whose values are zero in the normalization of the LREC curve. Finally, LRRDS extracts subsequence sets from the LREC curve by separating it with change points determined in the previous step.

Outlier reporting. The global discord degree (gdd) of a given subsequence in the subsequence sets is computed by the *length* and *mean* of the subsequence in the LRCE curve and is utilized as the anomaly score. Using the K-means clustering method, all the computed gdds will form two clusters. The subsequences in the cluster with a higher average gdd value are reported as anomalies.

3.2.3 SAND. SAND [3] is an online algorithm that can detect subsequence anomalies in stream settings based on their distances to a model Θ that represents normal behavior consisting of a set of clusters paired with weights. It extends the k-Shape clustering method to enable the clustering result (model) to be updated incrementally without storing any expired subsequences.

Slide processing. SAND employs k-Shape clustering method on the first slide to build the initial model. When each new slide arrives, k-Shape is executed again, and the cluster results are compared with the current model. The newly generated clusters that can find similar clusters in Θ are merged, while clusters without matches are added to Θ. Finally, SAND updates the new centroid and weight of each cluster in Θ.

Outlier reporting. For a given subsequence in the current slide, SAND computes the weighted sum of the distances between such subsequence and the model as its anomaly score. These anomaly scores are then normalized by the mean and standard deviation in order to be fairly compared among different slides. Subsequences with the top θ_k (an integer determined by users) highest anomaly score are reported as anomalies.

3.2.4 Luminol. Luminol [19] follows the idea of time series bitmaps and takes advantage of SAX [22] representation to detect anomalies. Without the extensive effort of domain experts, it computes the frequency of SAX words with a given length and maps the normalized frequency to colors, which can help users judge patterns.

Data processing. Luminol firstly converts the time series into a SAX string with a symbol set of size n. Secondly, for each data point, a *lag* window denotes the memory of the past, and a *lead* window shows how far to look ahead for anomalous patterns are created. Finally, it generates two bitmaps using the normalized frequency of SAX subwords at the desired level (e.g., the frequency of "aa", "ab" etc. will be calculated at level 2) for the *lag* window and *lead* window, respectively.

Outlier reporting. The distance between those two bitmaps is measured and considered an anomaly score of each data point. Data points with the top θ_K most significant anomaly scores are reported as anomalies.

3.2.5 SHESD. SHESD [14] extends the Extreme Studentized Deviate (ESD) test method to find anomaly points more robustly. Time series decomposition and median absolute deviation are applied in the proposed method to cope with seasonality issues and eliminate anomalies' effect.

Data processing. SHESD firstly uses a modified STL decomposition to extract the seasonal component of the input time series. Secondly, the residual component is computed by subtracting the seasonal component and median of the time series from the original. Finally, the residual component is used as input of ESD method.

Outlier reporting. The data points in the output from ESD, i.e., θ_K most extreme values larger than the computed critical values λ_k, are reported as anomalies.

3.3 Implementation Notes

To present a fair comparison, we re-implement all the methods in Java 8 with our best effort based on their proposed papers and email communications with the authors. The "Original Code" column in Table 1 indicates the original language where each method was written, while the "Speedup" column shows efficiency promotion after our implementation. We also transfer the codes of NETS, STARE and SHESD into our testbed (with the same data structure) to avoid the potential effect on efficiency. Our implementation with modifications leads to gains in performance across most of the methods except SAND. The main problem lies in the *matrix eigenvalue decompositions* part, where there is no efficient package in JAVA compared to the *scikit-learn* package in Python. As for reproducibility, because some works do not take the same metric as ours, instead of checking the accuracy result, we compare the reported anomalies from the methods in the original version and after our implementation and find out that they are the same.

Table 2: Real-world datasets summary

	Dataset	Size	#Dim	Rate%	Avg Length
Point	Yahoo [20]	1.5k	1	0.7	-
	Twitter [14]	14k	1	0.7	-
	Stock [30]	10k	1	synthetic	-
	Tao [30]	568k	3	synthetic	-
	DLR [34]	23k	9	2.2	-
	SMTP [23]	95k	3	0.03	-
	ECG [34]	112k	32	16.3	-
Subsequence	Power [17]	35k	1	8.6	750
	Taxi [28]	10k	1	2.3	48
	Sed [3]	100k	1	3.0	64
	Exercise [10]	10k	3	15.1	140.2
	Exathlon [16]	3k	19	17.4	64.1
	SwaT [24]	50k	51	8.9	558.2
	SMD [27]	28k	33	9.5	336.8

Table 3: Synthetic datasets summary

	Dataset	Size	#Dim	Rate%	Pattern	Avg Length
Point	Uni-point-g	10k-100k	1	5 - 40	Global	-
	Uni-point-c	10k	1	5 - 40	Contextual	-
	Mul-point	10k-100k	32	5 - 40	Global	-
Subsequence	Uni-sub-g	5k-50k	1	5 - 40	Global	50
	Uni-sub-s	5k-50k	1	5 - 40	Seasonal	50
	Uni-sub-t	5k	1	10	Trend	50
	Mul-sub-g	5k-50k	3/50	5 - 40	Global	50
	Mul-sub-s	5k-50k	3	5 - 40	Seasonal	50
	Mul-cor-g	5k	3	10	Global	50
	Mul-ncor-g	5k	3	10	Global	50

4 EXPERIMENT

This section will first show the settings and datasets used in the experiments. Then, insights found over intra-class and inter-class comparisons are explained. Finally, we provide recommendations for selecting anomaly detection methods for applications.

4.1 Settings

We run our experiments on a Windows 10 desktop with a 3.79GHz AMD 3900X 12 Core CPU and 128GB of RAM.

4.1.1 Datasets. We employ five real-world datasets with labeled point anomalies and seven real-world datasets with labeled subsequence anomalies from different sources, with various sizes, dimensions, and anomaly rates (lengths), as shown in Table 2. The real-world datasets are in the following:

Yahoo[2] is a dataset from Yahoo! lab, which provides time series with annotated anomalies taken from real products traffic with 1.5K to 20K records. **Twitter**[3] is a dataset extracted and compared in SHESD. **TAO** consists of three oceanographic and surface meteorological variables and is available at Tropical Atmosphere Ocean project[4]. **Power**[5] describes the power consumption of a Dutch research facility in the year 1997. **Sed**[6] is a dataset from the NASA Rotary Dynamics Laboratory that records simulated engine disk revolutions at 3k rpm speed. **Taxi**[7] records the number of New York City taxi passengers for every 30 minutes from July 2014 to January 2015 with five anomalies: the NYC Marathon, Thanksgiving, Christmas, New Year's day, and a snowstorm. **Exercise**[8] contains tracking of movement during indoor exercises with around 90 repetitions of lunges and 8-12 repetitions of squats. **Exathlon** is systematically built based on real data collected in a use-case scenario implemented on Apache Spark. **SwaT**[9] is a scaled-down version of a real-world industrial water treatment plant producing filtered water, which contains 4 days with attack scenarios. **SMD**[10] is a 5-week dataset collected from a large Internet company.

We also generate synthetic univariate and multivariate datasets with various kinds of anomaly patterns following the guideline in [18]. For point anomaly, global and contextual outliers are generated, while shapelet (also called global), seasonal, and trend outliers are created for subsequence anomaly. In order to avoid the effect of randomness, we run experiments on synthetic datasets five times with different seeds and report the average results. The summary of synthetic datasets can be found in Table 3.

4.1.2 Evaluation Metrics. We evaluate the algorithms on the following two aspects, i.e., accuracy and efficiency.

Accuracy. We use different metrics according to the anomaly type. As for point anomaly, we take the classical measure [1]:

$$Precision = TP/(TP + FP) \tag{1}$$

$$Recall = TP/(TP + FN) \tag{2}$$

where TP, FP, FN are the number of true positive, false positive and false negative, respectively.

As for subsequence anomaly, the overlap of predicted anomaly and true anomaly is addressed [28]. Given a set of real anomaly ranges $R = \{R_1, \dots, R_{N_r}\}$ and a set of predicted anomaly ranges $P = \{P_1, \dots, P_{N_p}\}$, we have

$$Recall_T(R, P) = \frac{\sum_{i=1}^{N_r} Recall_T(R_i, P)}{N_r} \tag{3}$$

$$Precision_T(R, P) = \frac{\sum_{i=1}^{N_p} Precision_T(R, P_i)}{N_p} \tag{4}$$

where $Recall_T(R_i, P)$ consists of $ExistenceReward$ which indicates whether we find the real anomaly sebsequence R_i and $OverlapReward$ which presents the overlap of such finding, $Precision_T(R, P_i)$ only has the $OverlapReward$. Such rewards can be defined by users and we take the suggested rewards proposed in [28] with $Flat\ bias$.

Efficiency. Apart from loading the data from the file and result evaluation, the total time of all the other procedures will be summed and reported as the time cost of the test algorithm.

4.2 Parameter Search and Sensitivity

In order to compare algorithms on various datasets fairly and precisely, though all of them are unsupervised methods, we still try our best to employ a broad parameter search for every method.

[2]http://labs.yahoo.com/Academic_Relations
[3]https://github.com/ruananswer/twitter-anomalyDetection-java/
[4]http://www.pmel.noaa.gov
[5]https://www.cs.ucr.edu/~eamonn/discords/
[6]https://c3.ndc.nasa.gov/dashlink/resources/314/
[7]https://github.com/numenta/NAB
[8]https://bitbucket.org/len_feremans/pbad/
[9]https://itrust.sutd.edu.sg/testbeds/secure-water-treatment-swat/
[10]https://github.com/NetManAIOps/OmniAnomaly/tree/master/ServerMachineDataset

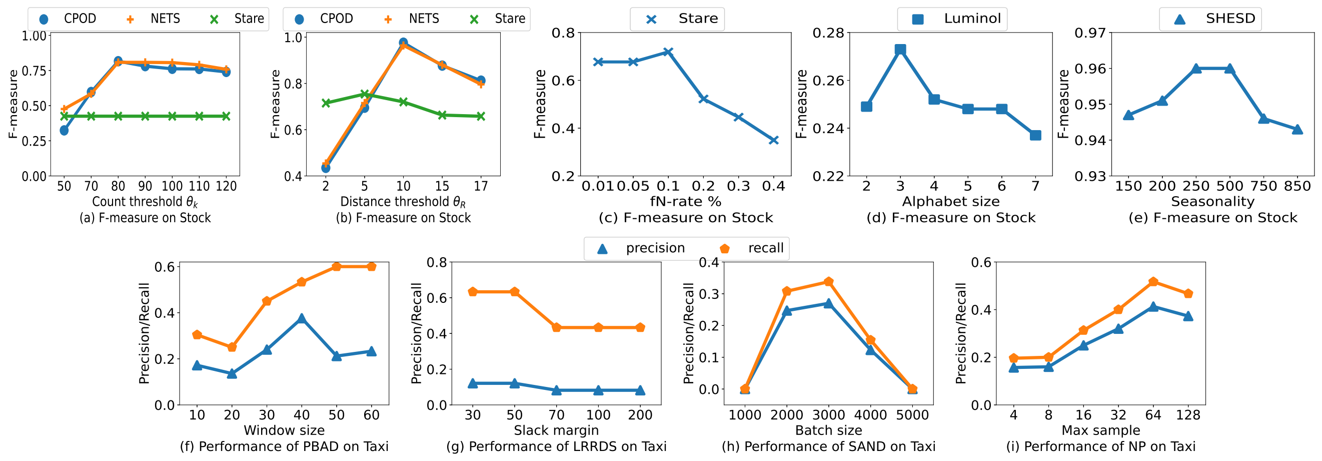


Figure 3: Parameter search of point algorithms (a-e) on Stock and subsequence algorithms (f-i) on taxi

However, for datasets larger than 10k, we only use the first 10k tuples to search for the best parameter. We present the tuning results in Figure 4.2 over the most important parameter, while more detailed results are shown in the technical report.

Point. For the distance-based methods CPOD, NETS and Stare, the most critical parameters are θ_k that denotes the count threshold for neighbors and θ_R that represents the distance threshold for identification of neighbors. We observe in Figure 4.2(a) that, after a particular value of θ_k, the increase of θ_k will not affect the accuracy too much, denoting that normal points in Stock have a large number of neighbors. Figure 4.2(b) shows that all methods are sensitive w.r.t. θ_R, which follows the common sense that the distance should be carefully set for a dataset. As illustrated in Figure 4.2(c), *fN* is the most sensitive parameter for Stare which represents the number of anomalies in the current window. However, it is not fair to feed the exact value for each window. Instead, we set it based on the anomaly rate of the dataset ($fN = Rate * Size/|Window|$) in the following experiments. As for Luminol in Figure 4.2(d), a middle-sized alphabet will lead to a better result, which is the same as suggested by its original paper [19]. Though Figure 4.2(e) illustrates a reverse "U-curve" for SHESD w.r.t. seasonality, the overall f-measure is always high, showing that it is robust on such parameter over Stock.

Sub. The most critical parameter for subsequence method is the length of anomaly, which is evaluated in Section 4.4.3. Thus we test other parameters for every algorithm, and the results are shown in Figure 3(f-i), respectively. For PBAD, more anomalies can be found as the increase of window size, results in an improvement of recall but a reduction of precision. As for LRRDS, a slack margin of 30 is a relatively good value which is consistent with the one proposed in its paper [15]. Small or large batch sizes will affect the accuracy of SAND and hence it is essential to choose an intermediate batch size as a compromise, as illustrated in Figure 4.2(h). Figure 4.2(i) shows an interesting trend on the influence of "Max sample" for NP. A too-large sampling size will affect the effect of bagging and harm the accuracy, which is contrary to the general sense of "the more, the better." Hence we finally choose an intermediate value.

4.3 Intra-class Comparisons

After settling all the parameters, we compare these algorithms in the same class over different scenarios. For simplicity, we use "**Point**" to denote the results from point methods and "**Sub**" for subsequence methods. "**Summary**" stands for the take-away conclusions.

4.3.1 Varying Datasets. In this section, we evaluate the performance of all methods on various datasets with their best parameters. To get the results of univariate algorithms on multivariate cases, we run them on each dimension (a univariate time series) of the multivariate time series and finally union all the reported anomalies per dimension as the final anomalies detected.

Point. We can find that CPOD, NETS always have high accuracy over all datasets in Figure 4(a). In theory, these two methods will have the same results under identical factors. However, we observe that NETS outperforms CPOD a bit on ECG and DLR. It is because NETS applies a sub-dimensionality selection mechanism that replaces the full-dimension with a selected sub-dimension to reduce the computation time, which makes it become the most efficient method (illustrated in Figure 5(a)). Interestingly, such a pruning strategy also promotes precision here. Another multivariate algorithm Stare loses around 0.3 on average over all the datasets compared to them. Its performance heavily relies on the parameter indicating the number of anomaly points in each window. As mentioned in Section 4.2, once we do not provide the exact value, it can hardly do well since the anomaly distributes unevenly in real datasets. Univariate methods SHESD get good accuracy on time series with clear seasonality, such as Yahoo, Twitter, and ECG and poor results on others. Luminol is not so capable when handling complex cases in real datasets. As for efficiency, we observe that CPOD takes much time. NETS and Stare can run 8x-20x faster than it. The possible reason is that CPOD restores more information and costs extra time when processing every slide.

Sub. The precision and recall of every subsequence approach are shown in Figure 4(b, c), respectively. We observe an interesting phenomenon that the recall is generally higher than precision, which indicates that these methods tend to find more or larger anomaly subsequences(ranges) but may not be accurate enough. The

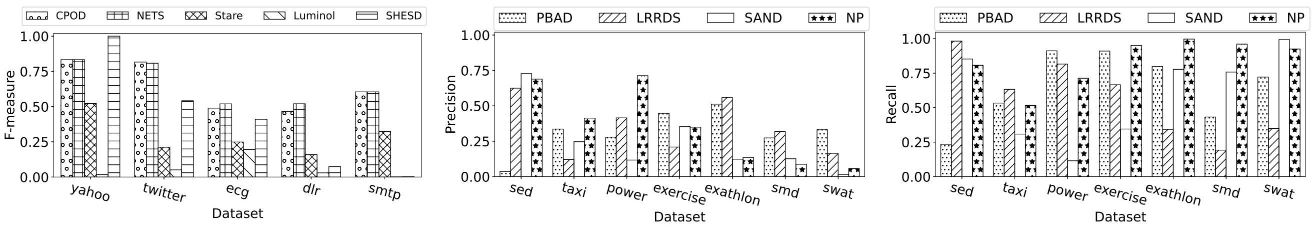

(a) F-measure with point anomaly (b) Precision with subsequence anomaly (c) Recall with subsequence anomaly

Figure 4: Accuracy over various datasets

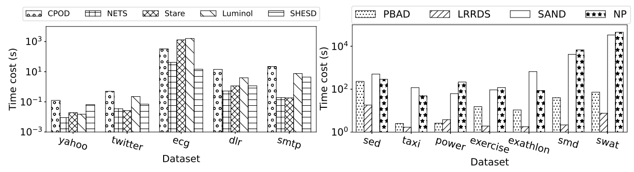

(a) Time cost with point anomaly (b) Time cost with subsequence anomaly

Figure 5: Time cost over various datasets

two multivariate algorithms, PBAD and LRRDS achieve higher precision and lower recall than univariate methods on multivariate datasets. PBAD performs better than LRRDS in general except for Sed, showing the superiority of the embedding techniques over simple mean and length features. In Sed, the abnormal subsequence consists of consecutive data points with small values frequently occurring in the data, which confuses PBAD. On the contrary, the mean value of such a pattern is obvious lower than normal patterns, making LRRDS easy to identify. On the other hand, LRRDS is the most efficient algorithm, as shown in Figure 5(b). PBAD also shows acceptable efficiency over all kinds of datasets. The univariate method NP achieves better results than SAND in general, especially in Power. It's surprising to recognize that SAND cost so much time on every dataset. SAND has to apply matrix eigenvalue decompositions many times when the anomaly has a long length (anomaly length in Power is 750). We observe that once the anomaly length is larger than 100, it will cost far more time.

Summary. (1) There is no super algorithm capable for all cases. (2) Given proper parameters, CPOD and NETS can outperform other point methods in most cases. (3) PBAD and NP have overall better accuracy but cost more time.

4.3.2 *Varying Anomaly Rate a%.*

To check the sensitivity against the percentage of anomalies, we run experiments by varying anomaly rates on synthetic datasets. The range of anomaly rates varies from 5% to 40% since the majority of the data should be normal. The size is 10k for point anomaly and 5k for subsequence.

Point. As shown in Figures 6(a, c), the F-measure of all the methods except Luminol decreases with the increasing of anomaly rate. The decrease in the multivariate dataset is sharper than in the univariate case. It follows the common sense that the more the anomaly points, the harder for algorithms to distinguish the anomaly. For example, for distance-based methods, it is hard to set a proper θ_k and θ_R to identify the anomaly points since they also have a similar number of neighbors with normal points. However, Luminol

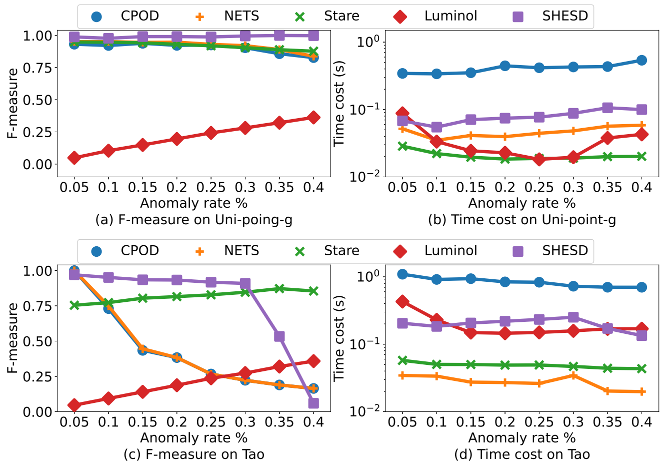

(a) F-measure on Uni-poing-g (b) Time cost on Uni-point-g (c) F-measure on Tao (d) Time cost on Tao

Figure 6: Varying anomaly rate on (a-b) Uni-point-g (c-d) Tao

acts differently from other methods since it tends to report most of the points as anomalies even with the best parameter. Thus, the precision and recall increase with anomaly rate, so as F-measure.

Sub. We inject two anomaly patterns for subsequence cases, i.e., Global and Seasonal in Uni and Mul-sub data, respectively. As shown in Figures 7(b, e), recall of all methods stays still on univariate data while decreasing on multivariate data. As for precision presented in Figures 7(a, d), most methods have a quick drop. Comparing the results of different algorithms, we find out that PBAD and SAND perform better at a lower anomaly rate but gradually become worse when the anomaly rate increases. While NP can maintain relatively stable precision when the anomaly rate is large. It is noted that LRRDS is quite sensitive to the anomaly rate. However, according to the later results, we find out that such a phenomenon holds for other variables.

Summary. (1) The higher the anomaly rate, the worse algorithms perform. (2) Anomaly rate shows little effect on the efficiency.

4.3.3 *Varying Data Size n.*

To check the sensitivity and scalability against the data size, we run experiments by varying data sizes on real and synthetic datasets with 10% anomalies.

Point. We run experiments on Uni-point-g and Tao, where injected anomalies have a uniform distribution. According to the results in Figures 8(b, d), all methods cost more time as the increase of data size. As for accuracy, they show consistent results against the data

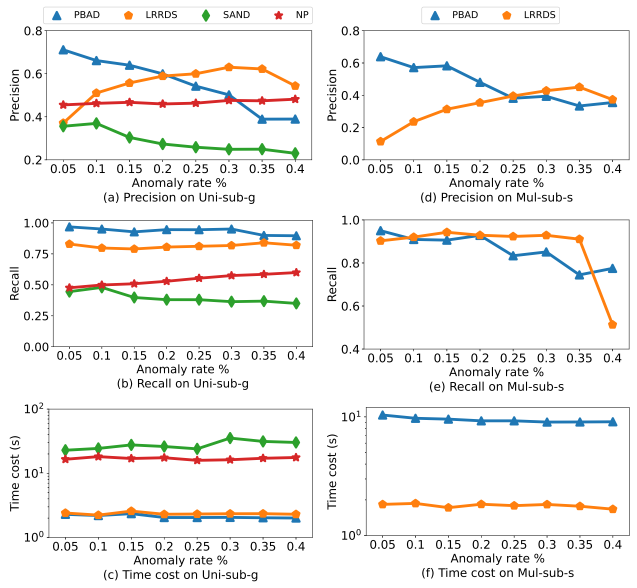


Figure 7: Varying anomaly rate on (a-c) Uni-sub-g (d-f) Mul-sub-s

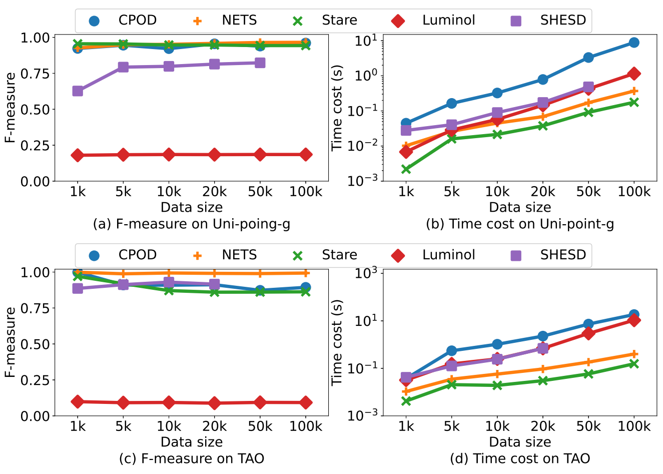


Figure 8: Varying data size on (a-b) Uni-point-g (c-d) Tao

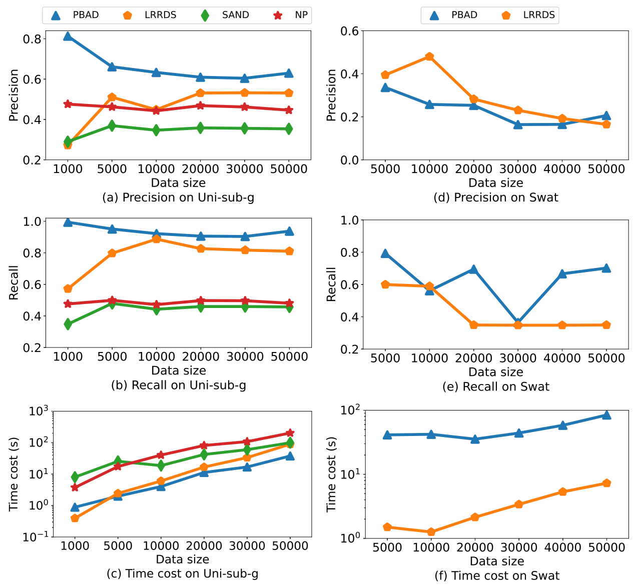


Figure 9: Varying data size on (a-c) Uni-sub-g (d-f) Swat

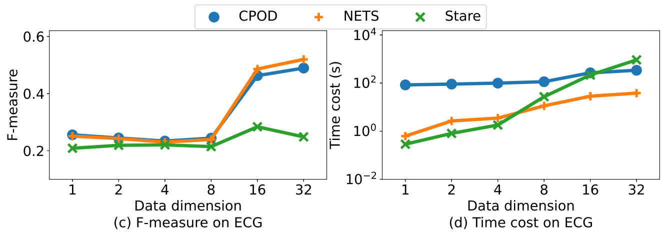


Figure 10: Varying dimension on ECG

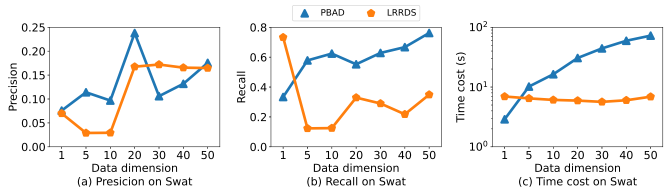
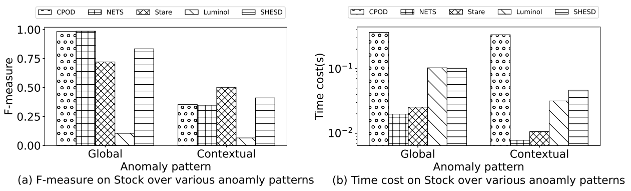


Figure 11: Varying dimension on Swat

size. It is noted that SHESD algorithm (we run the source code from the official website) throws exceptions under some cases, such as Uni-point-g with size 100k and TAO with size 50k and 100k. We do not find the exact reason yet, but it turns out to be one of our future work. Another interesting finding is that the online algorithm CPOD acts as the most time-consuming method while Stare and NETS are the top-two fastest methods. The latter two algorithms take advantage of the "set effect", which skips extra updates when new data arrives. However, CPOD stores more information and has a slower update in slide processing, even slower than the batch methods.

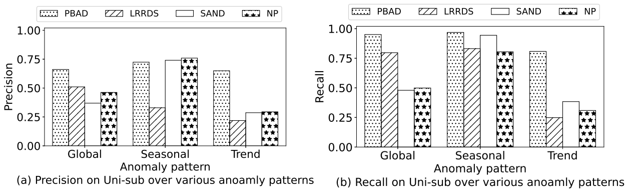
Sub. In the univariate synthetic dataset Uni-sub-g, we observe in Figures 9(a, b) that all methods have stable accuracy results when the size of the data exceeds 5k. When the data size is small (5k here), PBAD can have much better results since the support of anomaly pattern is relatively low, while SAND performs worse due to the difficulty to extract models. In general, PBAD maintains the best accuracy among the four algorithms while NP and SAND take more time. In the multivariate real dataset SwaT, the distribution of anomalies is extremely uneven since they only exist in the first 20k. Thus, the recall and precision tend to decrease with the data size increase.

Summary. (1) Stare and NETS have good efficiency as point methods while LRRDS is the fastest under subsequence case . (2) A small data size may lead to unstable results for subsequence methods.



(a) F-measure on Stock over various anoamly patterns (b) Time cost on Stock over various anoamly patterns

Figure 12: Varying anomaly patterns on point



(a) Precision on Uni-sub over various anoamly patterns (b) Recall on Uni-sub over various anoamly patterns

Figure 13: Varying anomaly patterns on subsequence

4.3.4 *Varying Data Dimension $|D|$.*

Here, we run experiments by varying data dimensions on real datasets.

Point. As illustrated in Figure 10, all methods have better results with larger data dimensions, from a relatively low value to the normal score (in Figure 4(a)). It is noted that, we have to modify the parameter θ_R with the change of dimension to keep a reasonable result, e.g., $\theta_{R_{D=4}} = (4/32)^{0.7} * \theta_{R_{D=32}}$. As for efficiency, all these distance-based methods cost more time with increasing time dimensions due to the higher cost of computing Euclidean distance. However, CPOD presents a relatively stable time cost, which may be because its efficiency is affected mainly by other executions other than distance computing (see Figure 4.4.2(b)).

Sub. The result of the subsequence anomaly can be checked in Figure 11. LRRDS applies dimensional compression and presents great scalability over data dimensions. In contrast, PBAD needs to extract features in each dimension, resulting in a significantly higher time cost as the dimension increase. Since the anomaly locates differently in every dimension of SwaT, it is not surprising that the performance fluctuates wildly in both methods.

Summary. (1) CPOD and LRRDS scales well against the data dimension while NETS, Stare and PBAD are affected significantly. (2) The sparsity problem may be coped with by a good parameter search since methods can get good accuracy with a high dimension.

4.3.5 *Varying Anomaly Patterns.*

Finally we test the selected algorithms on different patterns of anomaly.

Point. As Figure 12 shows, all methods get better accuracy on Global anomaly than Contextual anomaly, which follows the ordinary sense. They also have similar efficiency over such two anomaly patterns.

Sub. The result of the subsequence case is illustrated in Figure 13. Consistent with previous observations, PBAD achieves the best results among all anomaly patterns. SAND and NP are particularly good at seasonal outliers (even comparable to PBAD). LRRDS seems to be relatively more suitable for global outliers since significant mean deviations can be found. Detecting trend outliers is challenging for most algorithms. Only PBAD can get relatively good performance due to the reason that it can identify trend anomalies after extracting frequent patterns.

Summary. (1) For point methods, Global anomaly are easier to be identified than Contextual anomaly. (2) For subsequence methods, Seasonal anomaly are the easiest to be found.

4.4 Inter-class Comparisons

In this section, we will compare methods in a specific facet among different classes. In the data dimension facet, we will check the performance of the univariate method on multivariate datasets. In the processing technique facet, we will analyze the potential trade-off between effectiveness and efficiency by running online algorithms over various window and slide sizes on time series with finite observations. We will execute point algorithms on time series with subsequence anomalies in the anomaly type facet.

Table 4: Performance over each single dimension and after union on multivariate time series

	PBAD	LRRDS	SAND	NP
Pattern	Precision/Recall			
Exercise	**0.448**/0.91	0.209/0.666		
Union	0.174/**1**	0.132/**0.977**	0.353/**0.345**	0.349/**0.951**
E_A1	0.368/0.947	0.162/0.582	0.468/0.171	**0.884**/0.891
E_A2	0.368/0.844	0.259/0.874	**0.655**/0.222	0.179/0.169
E_A3	0.285/0.59	**0.308**/0.434	0.015/0.005	0.087/0.097
Mul_cor_g	0.69/0.936	0.152/0.522		
Union	0.481/**1**	0.104/**0.936**	0.49/**0.834**	0.663/**0.9**
M_A1	0.619/0.832	0.163/0.726	**0.985**/0.788	**0.985**/0.788
M_A2	**0.721**/0.858	0.24/0.904	0.67/0.536	0.727/0.582
M_A3	0.534/0.84	**0.45**/0.896	0.205/0.164	0.675/0.54
Mul_ncor_g	**0.663**/0.988	**0.307**/0.808		
Union	0.124/**1**	0.087/**0.875**	0.353/**0.912**	0.349/**0.976**
M_A1	0.165/0.385	0.085/0.345	**0.47**/0.37	**0.52**/0.418
M_A2	0.2/0.469	0.007/0.649	0.35/0.35	0.368/0.368
M_A3	0.174/0.478	0.077/0.623	0.289/0.315	0.328/0.328
Pattern	Time cost (s)			
Exercise	25.844	**1.886**		
Union	14.601	**5.110**	169.832	187.075
Mul_cor_g	**2.788**	8.875		
Union	**3.463**	22.737	83.955	19.651
Mul_ncor_g	**2.720**	9.205		
Union	**3.517**	23.816	70.203	27.057

4.4.1 *Univariate Algorithms in Multivariate Datasets.*

The accuracy of univariate algorithms on multivariate datasets with point anomaly has already been reported in Section 4.3.1. It shows that the performance of Luminol and SHESD fall behind the multivariate algorithms by a big gap. Thus, we focus on the subsequence case.

Settings. As mentioned before, we unite the reported anomalies on each dimension as the final anomalies detected by univariate methods. Specifically, for subsequence algorithms, if the detected anomaly ranges(subsequence) in different dimensions overlap after the union, we will combine these two or more ranges into one large anomaly range to be the final output. Following the same logic, we also run multivariate algorithms (PBAD and LRRDS) on

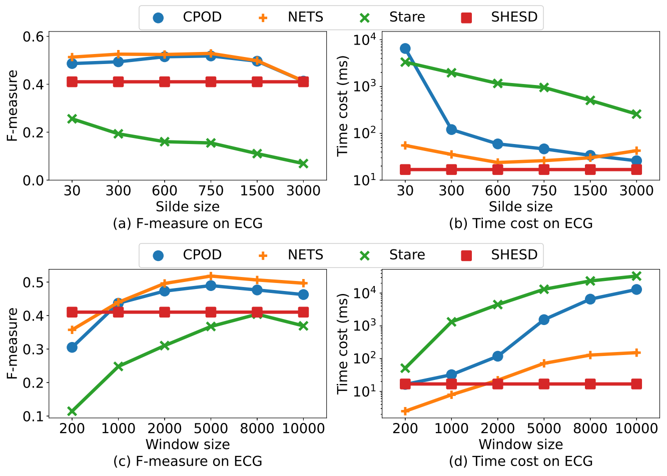


Figure 14: Varying (a-b) slide size (c-d) window size on ECG

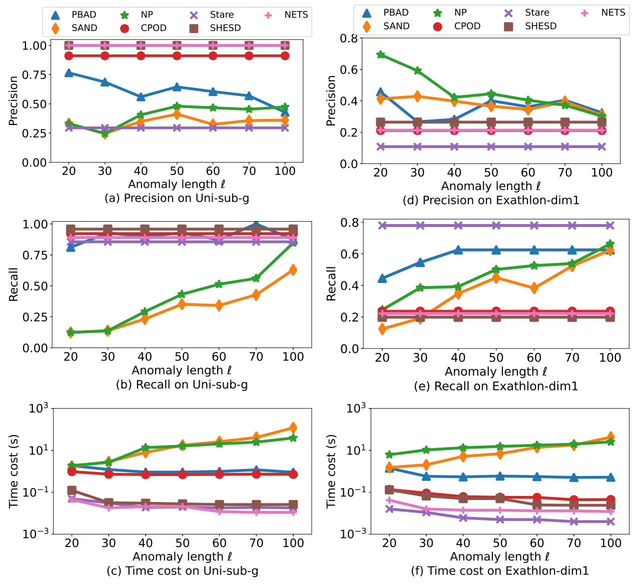


Figure 15: Varying anomaly length on (a-c) Uni-sub-g (d-f) Exathlon-dim1

each dimension and union the detected anomalies to make a direct comparison. These results are shown in lines ”union” in Table 4.

We find out that the anomalies on real dataset Exercise always occur over all dimensions at the same position. We also run experiments on two extra synthetic datasets to further test the performance over different scenarios. The ”ncor” indicates the positions of generated anomalies are different in each dimension, while ”cor” means they occur at the same locations over all dimensions.

Results. We have several interesting findings as follows. (1) Compared with the result on each dimension and after union, all methods have a promotion on recall but a reduction on precision. (2) The result after union can even have a higher recall than the natural output of multivariate methods. (3) Compared with the union results on the two synthetic data, we find out that anomaly positions (co-occur or not) do not clearly impact the effectiveness and efficiency of methods. (4) NP has good overall results but acts poorly on some specific dimensions, while PBAD may perform badly on every dimension but has better overall results, showing the advantage of considering relationships over dimensions. However, NP costs much more time due to the execution on every dimension and its time complexity. (5) PBAD can run faster when there is a limited number of patterns in the time series.

Summary. The first two observations above suggest that we can execute a certain algorithm on every dimension of multivariate time series to reduce the possibility of missing alert in applications.

4.4.2 *Online Algorithms in Batched Time Series.*

As explained in Section 4.3.1, the online algorithm SAND suffers from the iterations of matrix eigenvalue decompositions and its efficiency is significantly affected. Thus, we focus on the point case.

Settings. We run two experiments on ECG varying slide size θ_S and window size θ_W. In the former case, we have $\theta_W = 3k$ while in the latter, $\theta_S = 150$. It is noted that, $\theta_S \leq \theta_W$ always holds. Batch method SHESD is used as a baseline.

Results. Figure 14(a) shows that the accuracy of CPOD and NETS will first increase and then decrease as slide size increases. A possible explanation is that when the slide is too small, it is hard to find enough neighbors even for a normal data point, leading to false positives. When the slide is too large, the number of neighbors for a local anomaly point may exceed the pre-defined threshold θ_k, leading to false negatives. As for efficiency presented in Figure 14(b), CPOD and Stare will take less time as slide size increases. Specifically, we observe a significant drop on time cost of CPOD, which is consistent with the aforementioned description in Section 4.3.4 that other executions but not distance computation cost the most time. NETS can not take advantage of the ”net effect” if the slide size equals the window size. Thus it can take the least time with a middle θ_S. Similar results hold when we vary the window size. It is noted that even the fastest online method NETS will take around 10^2 seconds when $\theta_W = 10k$, which is ten times slower than the simple batch method SHESD.

Summary. The trade-off of accuracy and efficiency is not trivial over window/slide size. Larger window size and slide size does not necessarily get better results, though they cost more time. We should find an intermediate window/slide size for online methods to get good results within an acceptable time.

4.4.3 *Point Algorithms in Subsequence Anomalies.*

The state-of-the-art subsequence method always considers the anomaly length ℓ as an input parameter. However, it will somehow affect the accuracy since the subsequence anomalies in real-world time series data seldom have fixed lengths. In contrast, point anomaly methods do not require such input (the anomaly length is 1 for point anomaly). Thus, we want to know the impact of anomaly length and whether it can execute point methods in subsequence anomalies.

Settings. We use univariate time series in order to avoid the effect of dimensions. Apart from the synthetic Uni-sub-g dataset, we extract the first dimension of Exathlon as Exathlon-dim1 since it has several anomaly subsequences with a medium length.

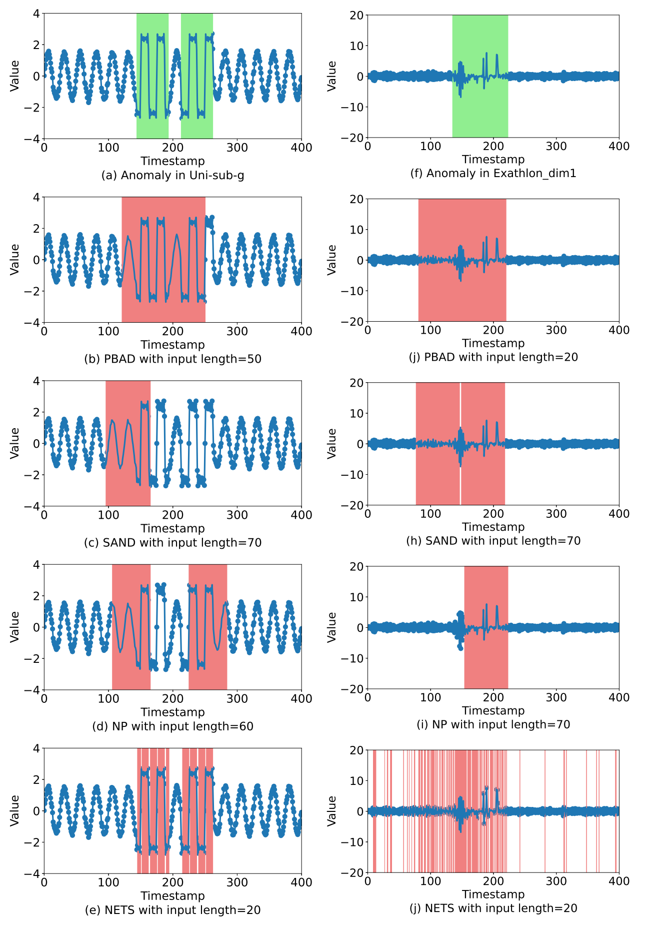


Figure 16: Case study on (a-e) Uni-sub-g (f-j) Exathlon-dim1

Results. As we can see in Figures 15(a, b), PBAD still has the best precision and recall compared to other subsequence methods. It is noted that given length ℓ, PBAD does not identify the subsequence whose length is exactly ℓ, making it different from NP and SAND. The latter two methods will reach the best accuracy when ℓ is similar to the actual length of anomalies. It is surprising to find that point methods such as NETS, CPOD and SHESD achieve incredible precision and recall under such cases. Let alone they cost much less time (as shown in Figure 15(c)). We turn to a case study to better understand what happened in the experiments. The real anomaly is illustrated in Figure 16(a), while the identified anomaly of different algorithms is shown in Figures 16(b-e), respectively. We can find out that PBAD can cover the anomaly subsequence, but SAND and NP fail to do so. We also observe that the detected anomaly points from NETS are almost the same as real anomaly subsequences. A possible explanation is that, with proper parameters (say, θ_k and θ_R), NETS can identify the subsequence anomaly with the Global pattern. We execute another experiment on Exathlon-dim1 and find similar results for subsequence method w.r.t. the impact of length ℓ. However, point methods are not able to get accurate results under such complex case. As shown in Figures 16(f-j), NETS identify much more points as anomalies.

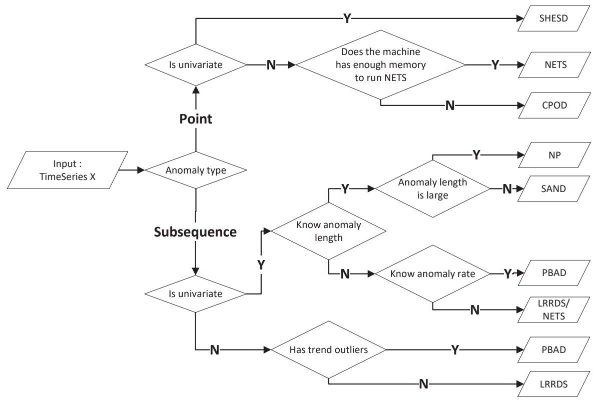


Figure 17: Guide for selecting anomaly detection methods

Summary. (1) Point anomaly methods can also perform well under subsequence anomaly cases, which may help eliminate the requirement of anomaly length or adaptively set it. (2) The length parameter ℓ close to the actual anomaly length leads to good results.

4.5 Discussions

We have described our findings in the above sections. The non-trivial highlights are summarized in Section 1. Based on these findings, we propose a practical guide for anomaly detection in Figure 17. In the following, we suggest some research opportunities:

Adaptive Window Size. Currently, the online algorithms take window size and slide size as input parameters. However, it is non-trivial to pre-define an appropriate size. Some sampling techniques [26] can be used in this context.

Key Dimensions Discovery. Data points in a high-dimensional data set are usually sparsely distributed. Existing methods can have good anomaly detection performance over datasets with high dimensions, but they require a great effort in parameter tuning. If we can find the key dimensions w.r.t. anomalies, detection methods' accuracy and efficiency can be further improved.

Input Length Relaxation. It is common to input the length of the anomaly for subsequence anomaly detection methods in the literature [3]. However, it contradicts the real datasets since the latter have different lengths (even different patterns) of anomalies. There is a chance for proposing a bottom-up method that forms anomaly subsequence from identified anomaly points to relax the fixed value of input length.

5 CONCLUSION

In this paper, we re-implement nine anomaly detection methods and make a systematic comparison using our testbed. Based on our classification of anomaly detection in time series, intra-class and inter-class comparisons are proposed. We provide a detailed analysis of the accuracy and efficiency of each method under different cases and summarize non-trivial findings. We also propose a practical guide for algorithm selection and suggest interesting future research opportunities.

REFERENCES

[1] Charu C. Aggarwal. 2013. *Outlier Analysis.* Springer.

[2] Ane Blázquez-García, Angel Conde, Usue Mori, and José Antonio Lozano. 2021. A Review on Outlier/Anomaly Detection in Time Series Data. *ACM Comput. Surv.* 54, 3 (2021), 56:1–56:33.

[3] Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J. Franklin. 2021. SAND: Streaming Subsequence Anomaly Detection. *Proc. VLDB Endow.* 14, 10 (2021), 1717–1729.

[4] Mohammad Braei and Sebastian Wagner. 2020. Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art. *CoRR* abs/2004.00433 (2020).

[5] Bernardo Branco, Pedro Abreu, Ana Sofia Gomes, Mariana S. C. Almeida, João Tiago Ascensão, and Pedro Bizarro. 2020. Interleaved Sequence RNNs for Fraud Detection. In *KDD.* ACM, 3101–3109.

[6] Mikel Canizo, Isaac Triguero, Angel Conde, and Enrique Onieva. 2019. Multi-head CNN-RNN for multi-time series anomaly detection: An industrial case study. *Neurocomputing* 363 (2019), 246–260.

[7] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep Learning for Anomaly Detection: A Survey. *CoRR* abs/1901.03407 (2019).

[8] Andrew A. Cook, Goksel Misirli, and Zhong Fan. 2020. Anomaly Detection for IoT Time-Series Data: A Survey. *IEEE Internet Things J.* 7, 7 (2020), 6481–6494.

[9] Ethan W. Dereszynski and Thomas G. Dietterich. 2011. Spatiotemporal Models for Data-Anomaly Detection in Dynamic Environmental Monitoring Campaigns. *ACM Trans. Sens. Networks* 8, 1 (2011), 3:1–3:36.

[10] Len Feremans, Vincent Vercruyssen, Boris Cule, Wannes Meert, and Bart Goethals. 2019. Pattern-Based Anomaly Detection in Mixed-Type Time Series. In *ECML/PKDD (1) (Lecture Notes in Computer Science)*, Vol. 11906. Springer, 240–256.

[11] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. 2014. Outlier Detection for Temporal Data: A Survey. *IEEE Trans. Knowl. Data Eng.* 26, 9 (2014), 2250–2267.

[12] D. M. Hawkins. 1980. *Identification of Outliers.* Springer.

[13] Yuanduo He, Xu Chu, and Yasha Wang. 2020. Neighbor Profile: Bagging Nearest Neighbors for Unsupervised Time Series Mining. In *ICDE.* IEEE, 373–384.

[14] Jordan Hochenbaum, Owen S. Vallis, and Arun Kejariwal. 2017. Automatic Anomaly Detection in the Cloud Via Statistical Learning. *CoRR* abs/1704.07706 (2017).

[15] Min Hu, Xiaowei Feng, Zhiwei Ji, Ke Yan, and Shengchen Zhou. 2019. A novel computational approach for discord search with local recurrence rates in multivariate time series. *Inf. Sci.* 477 (2019), 220–233.

[16] Vincent Jacob, Fei Song, Arnaud Stiegler, Bijan Rad, Yanlei Diao, and Nesime Tatbul. 2021. Exathlon: A Benchmark for Explainable Anomaly Detection over Time Series. *Proc. VLDB Endow.* 14, 11 (2021), 2613–2626.

[17] Eamonn J. Keogh, Jessica Lin, Sang-Hee Lee, and Helga Van Herle. 2007. Finding the most unusual time series subsequence: algorithms and applications. *Knowl. Inf. Syst.* 11, 1 (2007), 1–27. https://doi.org/10.1007/s10115-006-0034-6

[18] Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and Xia Hu. 2021. Revisiting Time Series Outlier Detection: Definitions and Benchmarks. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1).* https://openreview.net/forum?id=r8IvOsnHchr

[19] Alexander Lavin and Subutai Ahmad. 2015. Evaluating Real-Time Anomaly Detection Algorithms - The Numenta Anomaly Benchmark. In *ICMLA.* IEEE, 38–44.

[20] Kim-Hung Le and Paolo Papotti. 2020. User-driven Error Detection for Time Series with Events. In *ICDE.* IEEE, 745–757.

[21] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. 2019. MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks. In *ICANN (4) (Lecture Notes in Computer Science)*, Vol. 11730. Springer, 703–716.

[22] Jessica Lin, Eamonn J. Keogh, Li Wei, and Stefano Lonardi. 2007. Experiencing SAX: a novel symbolic representation of time series. *Data Min. Knowl. Discov.* 15, 2 (2007), 107–144.

[23] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *ICDM.* IEEE Computer Society, 413–422.

[24] Aditya P. Mathur and Nils Ole Tippenhauer. 2016. SWaT: a water treatment testbed for research and training on ICS security. In *2016 International Workshop on Cyber-physical Systems for Smart Water Networks, CySWater@CPSWeek 2016, Vienna, Austria, April 11, 2016.* IEEE Computer Society, 31–36. https://doi.org/10.1109/CySWater.2016.7469060

[25] Pavel Senin, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, Arnold P. Boedihardjo, Crystal Chen, and Susan Frankenstein. 2018. GrammarViz 3.0: Interactive Discovery of Variable-Length Time Series Patterns. *ACM Trans. Knowl. Discov. Data* 12, 1, Article 10 (Feb. 2018), 28 pages. https://doi.org/10.1145/3051126

[26] Shaoxu Song, Aoqian Zhang, Jianmin Wang, and Philip S. Yu. 2015. SCREEN: Stream Data Cleaning under Speed Constraints. In *SIGMOD Conference.* ACM, 827–841.

[27] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust Anomaly Detection for Multivariate Time Series through Stochastic Recurrent Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019.* ACM, 2828–2837.

[28] Nesime Tatbul, Tae Jun Lee, Stan Zdonik, Mejbah Alam, and Justin Gottschlich. 2018. Precision and Recall for Time Series. In *NeurIPS.* 1924–1934.

[29] Markus Thill, Wolfgang Konen, and Thomas Bäck. 2017. Online anomaly detection on the webscope S5 dataset: A comparative study. In *EAIS.* IEEE, 1–8.

[30] Luan Tran, Liyue Fan, and Cyrus Shahabi. 2016. Distance-based Outlier Detection in Data Streams. *Proc. VLDB Endow.* 9, 12 (2016), 1089–1100.

[31] Luan Tran, Minyoung Mun, and Cyrus Shahabi. 2020. Real-Time Distance-Based Outlier Detection in Data Streams. *Proc. VLDB Endow.* 14, 2 (2020), 141–153.

[32] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn J. Keogh. 2016. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. In *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain.* IEEE Computer Society, 1317–1322.

[33] Susik Yoon, Jae-Gil Lee, and Byung Suk Lee. 2019. NETS: Extremely Fast Outlier Detection from a Data Stream via Set-Based Processing. *Proc. VLDB Endow.* 12, 11 (2019), 1303–1315.

[34] Susik Yoon, Jae-Gil Lee, and Byung Suk Lee. 2020. Ultrafast Local Outlier Detection from a Data Stream with Stationary Region Skipping. In *KDD.* ACM, 1181–1191.

Table 5: Set of parameters for each method over which we perform grid search.

Method	Parameter	Description	Search space
NETS	θ_R	Distance threshold	$\{4, 6, 8, 10, 12\}$
	θ_k	Neighborhood threshold	$\{3, 4, 5, 6, 7, 8\}$
	θ_S	Sliding size	$\{2, 3, 4, 5, 6, 7\}$
	θ_W	Window size	$\{12, 14, 16, 18, 20\}$
CPOD	θ_R	Distance threshold	$\{4, 6, 8, 10, 12\}$
	θ_k	Neighborhood threshold	$\{3, 4, 5, 6, 7, 8\}$
	θ_S	Sliding size	$\{2, 3, 4, 5, 6, 7\}$
	M	The multiples of window and slide	$\{2, 4, 6, 8\}$
Stare	θ_R	Distance threshold	$\{4, 6, 8, 10, 12, 14\}$
	θ_k	Neighborhood threshold	$\{3, 4, 5, 6\}$
	θ_S	Sliding size	$\{3, 4, 5, 6, 7\}$
	θ_W	Window size	$\{15, 16, 18, 20\}$
Luminol	A	Alphabet size	$\{2, 3, 4, 5, 6\}$
	C	Chunk size	$\{7, 8, 9, 10\}$
	L	Lag window size	$\{8, 10, 12\}$
	F	Future window size	$\{8, 10, 12\}$
SHESD	θ_S	Seasonality	$\{500, 1000, 1500, 2000\}$
	M	Max anomaly rate	$\{0.2, 0.25, 0.3, 0.35\}$
	A	Alpha	$\{0.25, 0.3, 0.35, 0.4, 0.5\}$
	aT	Anoms threshold	$\{0.2, 0.5, 0.8\}$
PBAD	Window size	The subsequence length	$\{12, 20, 40, 100\}$
	Window incr	Windows overlapping rate	$\{6, 10, 20\}$
	Bin size	The moving average length	$\{1, 2, 4, 10\}$
	Threshold	Threshold for evaluation	$\{0.1, 0.2\}$
LRRDS	Compressed rate	The rate of compression using PAA	$\{0.1, 0.2, 0.5\}$
	Slack	Slack margin	$\{30, 50, 70, 100\}$
	Sub minlength	The min length of anomaly	$\{5, 10\}$
SAND	Batch size	The size of input for each batch	$\{2000, 5000\}$
	Pattern length	The subsequence length	$\{50, 75\}$
	Top k	Threshold for evaluation	$\{5, 9\}$
NP	Max sample	The subsample size	$\{8, 16, 32, 64\}$
	Sub length	The subsequence length	$\{50, 75\}$
	Scale	The normalization method	$\{demean, zscore\}$
	Top k	Threshold for evaluation	$\{5, 9\}$

A PARAMETER TUNING

Given the many parameters in each of the compared methods and the numerous data sets we wish to benchmark against, it is intractable to perform a thorough parameter search for every method. We begin with their default parameters as described in their corresponding papers or specified in their open-source implementations for all the methods. We choose the most influential parameter sets for each method, as shown in Table 5, to perform a grid search to select the parameters with the best performance. For large datasets larger than 10k, only the first 10k univariate data and 5k of multivariate data are used for tuning. For the point anomaly part, we use the stock dataset to represent all datasets because the parameter is much different in each dataset.

B VARYING ANOMALY RATE

We also test algorithm performance on Stock and Mul-point-g with point anomaly, and the results are in Figure 18. All the algorithms' F-measure slides down when the anomaly rate increases and time cost remains steady with some slight fluctuation. As for subsequence anomaly, as shown in Figures 19(a, f), we observe similar features to those in Figures 7(a, f).

C VARYING DATA SIZE

The result of the point anomaly algorithm is shown in Figures 20 and when the data size increases, the time cost increases as well. As shown in Figures 21(a, f), we observe similar features to those in Figures 9(a, f), except for PBAD, which performs very badly in Sed. The reason has been explained in Section 4.3.1.

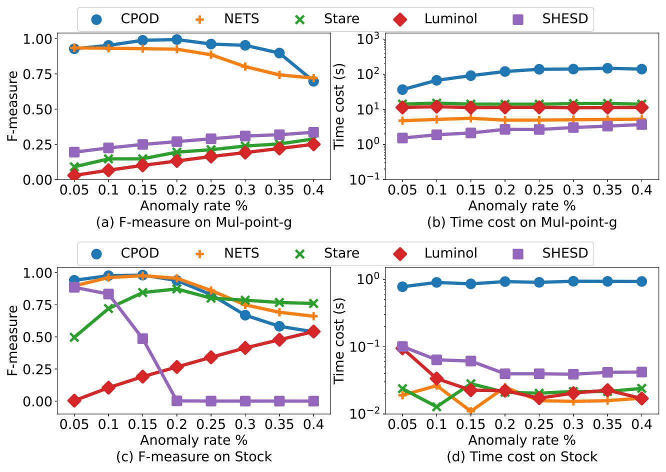


Figure 18: Varying anomaly rates on (a-b) Mul-point-g (c-d) Stock with point anomaly

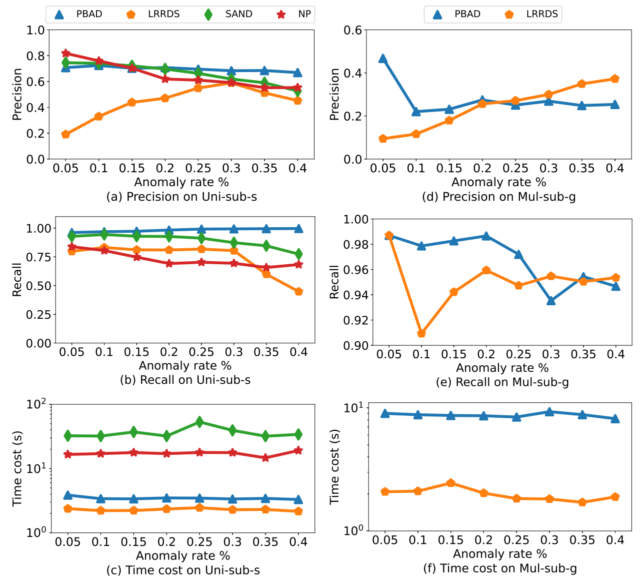


Figure 19: Varying anomaly rates on (a-c) Uni-sub-s (d-f) Mul-sub-g with subsequence anomaly

D VARYING DATA DIMENSION

We have interesting findings on Mul-pointg-g, shown in Figure 22. We can see that CPOD and NETS reach a slight peak when the data dimension is 1. When the data dimension rises from 1, the anomaly data on other dimensions may shrink the distance between outliers and inliers. This may cause an anomaly point may like a normal point. As shown in Figures 23(a, b), PBAD is not affected by data dimension in the Mul-sub-g, while LRRDS is as unstable as ever.

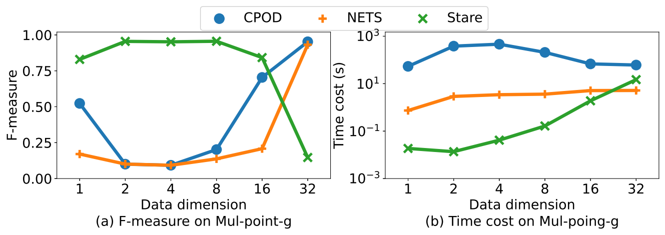


Figure 22: Various data dims on Mul-point-g with point anomaly

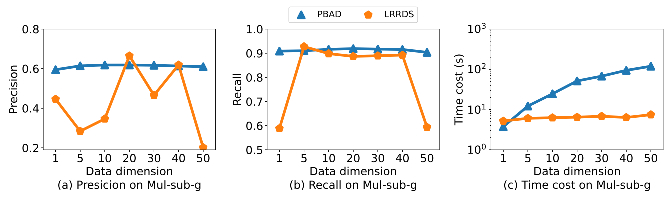


Figure 23: Varying dimension on Mul-sub-g with subsequence anomaly

Table 6: Precision/Recall over each single dimension of univariate algorithms

Pattern	PBAD	LRRDS	SAND	NP
Exathlon	**0.512**/0.798	**0.558**/0.342		
Union	0.118/**1**	0.152/**0.813**	0.123/**0.778**	0.135/**1**
T_A1	0.443/0.672	0.117/0.324	**0.606**/0.435	**0.576**/0.548
T_A2	0.355/0.429	0.09/0.547	0.01/0.014	0.248/0.384
T_A3	0.167/0.313	0.142/0.406	0.332/0.22	0.391/0.276
Mul_ncor_mix	**0.308**/0.714	**0.191**/0.601		
Union	0.149/**1**	0.092/**0.814**	0.377/**0.723**	0.317/**0.905**
M_A1	0.21/0.586	0.077/0.42	0.368/0.454	0.313/0.429
M_A2	0.213/0.446	0.117/0.377	**0.4**/0.297	0.413/0.313
M_A3	0.289/0.396	0.06/0.603	0.063/0.143	**0.425**/0.322
Mul_cor_mix	0.542/0.78	**0.204**/0.818		
Union	0.47/**1**	0.073/**0.866**	0.644/**0.836**	0.64/**0.894**
M_A1	**0.619**/0.832	0.163/0.726	**0.985**/0.788	**0.985**/0.788
M_A2	0.48/0.752	0.136/0.81	0.985/0.788	0.91/0.728
M_A3	0.567/0.86	0.142/0.214	0.415/0.35	0.288/0.23

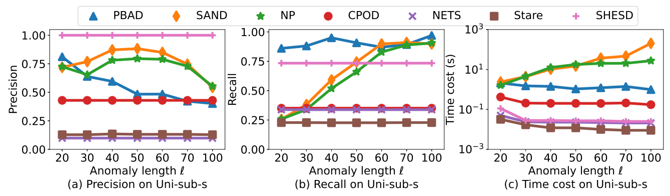


Figure 24: Varying anomaly length on (a-c) Uni-sub-s

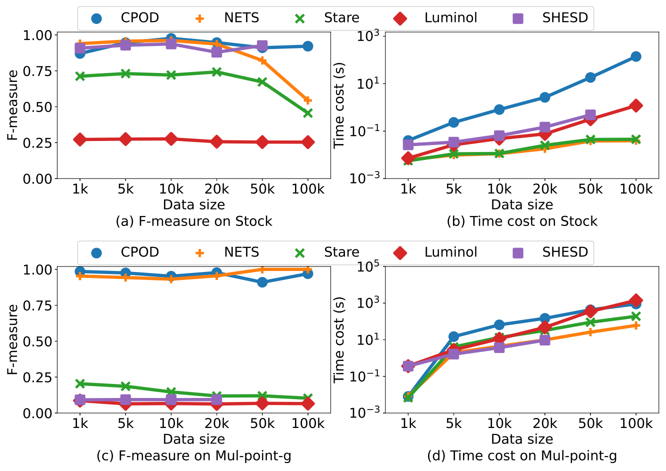


Figure 20: Various data size on (a-b) Stock (c-d) Mul-point-g with point anomaly

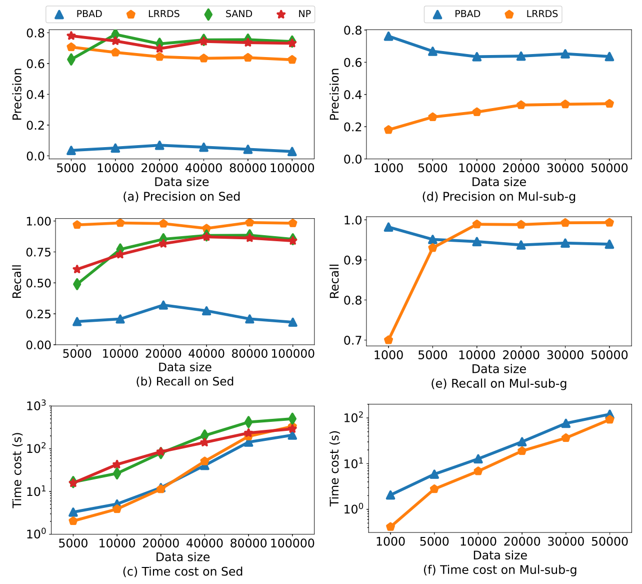


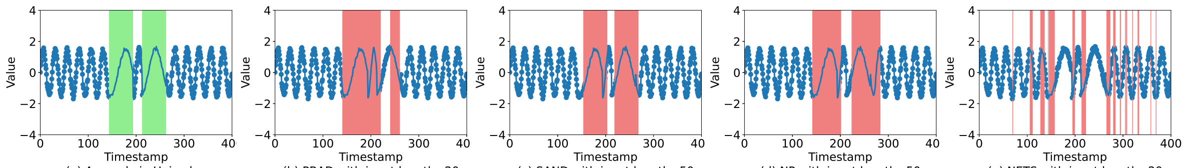
Figure 21: Various data size on (a-c) Sed (d-f) Mul-sub-g with subsequence anomaly

E DETAILED RESULTS OF UNIVARIATE ALGORITHMS IN MULTIVARIATE DATASETS

We demonstrate the result on Exathlon in Table 6, with only the intermediate results of the three dimensions captured. We can observe the same characteristics as in Table 4.

F DETAILED RESULTS OF POINT ALGORITHMS IN SUBSEQUENCE ANOMALIES

It can be observed that except for SHESD, the performance of other point algorithms is very low. We do the same experiment in uni-sub-s, and the results are shown in Figure 24. Likewise, we also perform a case study in Figure 25. As can be seen from the figure, subsequence algorithms can find the seasonal outlier very well, while NETS cannot find the real anomaly point at all.



(a) Anomaly in Uni-sub-s (b) PBAD with input length=20 (c) SAND with input length=50 (d) NP with input length=50 (e) NETS with input length=20

Figure 25: Case study on (a-e) Uni-sub-s