Anomaly detection in multidimensional time series—a graph-based approach

Marcus Erz, Jeremy Floyd Kielman, Bahar Selvi Uzun and Gabriele Stefanie Gühring
Fakultät Informatik und Informationstechnik, Hochschule Esslingen, Germany
*
Author to whom any correspondence should be addressed.
E-mail: maerit02@hs-esslingen.de, jekiit01@hs-esslingen.de, bauzit01@hs-esslingen.de and gabriele.guehring@hs-esslingen.de

Keywords: anomaly detection, graph-based algorithm, time series transformation, dynamic graphs

Abstract
As the digital transformation is taking place, more and more data is being generated and collected. To generate meaningful information and knowledge researchers use various data mining techniques. In addition to classification, clustering, and forecasting, outlier or anomaly detection is one of the most important research areas in time series analysis. In this paper we present a method for detecting anomalies in multidimensional time series using a graph-based algorithm. We transform time series data to graphs prior to calculating the outlier since it offers a wide range of graph-based methods for anomaly detection. Furthermore the dynamic of the data is taken into consideration by implementing a window of a certain size that leads to multiple graphs in different time frames. We use feature extraction and aggregation to finally compare distance measures of two time-dependent graphs. The effectiveness of our algorithm is demonstrated on the numenta anomaly benchmark with various anomaly types as well as the KPI-anomaly-detection data set of 2018 AIOps competition.

1. Introduction
As a result of digitalization, data is increasingly being generated and collected digitally. Obtaining immense amounts of data was once very costly and difficult, but nowadays the cost of storage options has rapidly decreased [1]. Thus, a massive amount of data can be collected over time and used with various data mining techniques to generate meaningful information and knowledge [2].

In addition to classification, clustering, and forecasting, outlier or anomaly detection is one of the most important research area in time series analysis i.e. [3–5]. Outlier detection in time series characteristically involves the labeling of unusual changes, subsequences, and temporal patterns in the data as outliers [6]. Typically, outlier detection is used for credit card fraud detection, attack vectors in computer networks, fault diagnosis in the industry or predictive health management [6].

Representing time series data in graphs can be helpful in displaying outliers or anomalies in the data. The transformation of a times series to a graph enables the comparison of one time series segment to another time series segment, allowing the study of data objects that are now interdependent. The assumption in the research of graph-based algorithms for outlier detection is that these algorithms can detect outliers or anomalies in time series. Furthermore, it is competitive to the use of neural networks [3].

In this paper we explore existing graph-based outlier detection algorithms applicable to static and dynamic graphs. We optimize the NetSimile algorithm [7] that extracts and statistically compares the structural properties of data over time and apply it to time series that are structured as graphs. NetSimile, a scalable approach to size-independent network similarity, is solely designed for network structures and does not work on traditional time series data. Given that we want to take advantage of graph-based algorithms for time series, in this work, we contribute a way to represent time series data as graphs and likewise, detect outliers in multidimensional time series. Our results can be summarized as follows. We have applied a novel dynamic graph-based algorithm to multidimensional time series data and successfully detected different types of outliers. We have
presented a method that transforms time series data into graphs. For the NetSimile-based algorithm, we have
developed additional features to apply it to fully connected graphs. This combination of transforming time
series into graphs and then applying the proposed modification of the NetSimile algorithm has not been eval-
uated in the literature before. We have tested the efficiency of our algorithm on synthetic as well as real data
sets.

2. Related work

The following taxonomy subdivides the algorithms evaluated for this work and gives a general overview on the
research of graph-based methods used to analyse time series.

Akoglu et al [3] provide a survey on outlier detection in static graphs. In static graphs, outlier detection
methods can be structured in the following way.

Structure-based approach: with the help of the representation of nodes and edges in an ego network [8],
structure-based approaches extract associated properties of each ego network, such as the number of nodes and
edges. Subsequently, this kind of algorithms identifies those nodes and edges that structurally differ massively
from the rest of the ego networks. Kang et al [9] use a structure-based approach to show nonzero patterns in
the adjacency matrix. Their algorithm provides a solution to find structural anomalies visually in plots. In [10]
small pattern changes are declared as structural anomalies.

Clustering-based approach: in [11], two nodes and the intersection of their neighbouring nodes are con-
trasted. Thus, the assumption is made that nodes with very few neighbours compared to the rest of the nodes
in their neighbourhood are outliers.

IsoMap-based approach: by performing dimension reduction using the IsoMap-based algorithm in [12]
information loss about outliers occurs. Reconstructing the whole graph with the reduced dimensions can-
not recover the outlier information. Through a subsequent comparison of the extracted information, outliers
become visible.

Percolation-based approach: the percolation-based algorithm in [12] makes the assumption that outlier
nodes have higher edge weights than their neighbours. In this approach, the edges with the highest weights are
removed from a graph step by step. This removal separates outliers from the rest of the network.

Edge-based approach: in [13], the algorithm iterates randomly over the network. It starts at any node of the
network. The algorithm then switches to a neighbouring node on a random basis. The higher the edge weights
to a neighbouring node are, the higher the probability that this node will be visited. The algorithm records
how often nodes are visited, since nodes with high edge weights to their neighbours are visited more often
than nodes with lower edge weights to their neighbours. Nodes that are visited less frequently are declared as
outliers.

Ranshous et al [4] provide a survey on outlier detection in dynamic graphs.

The approach in [7], which is similar to our approach, considers a snapshot of the graph at temporal
intervals. However, we extend the NetSimile approach so that it can be applied to weighted graphs and use it
to identify outliers or anomalies in multivariate times series. Different structural features of two snapshots of
a graph are compared to evaluate the similarity between them. An outlier is declared in case of a significant
change in the graph.

Bathia et al compare, in [14], each incoming edge with the occurrence of the same edges in previous time
instances. In doing so, it detects microclusters that are anomalous.

To use graph-based algorithms for detecting outliers in time series data, at first the data must be converted
to graphs [13]. Here, distance measures are used to improve the quality of outlier detection. Hence, the use of
suitable distance measures is essential for detecting anomalies and outliers via graph-based algorithms.

A comparison of the different distance measures can be found in [15]. For transforming our time series
data into graphs, we decided to use the Minkowski distance with either $p = 2$ or $p = 3$ for two times series,
which have equal length, but other distance measures are also possible and can be evaluated in future work.

3. Graph-based times series anomaly detection

The following section presents the principles of our graph-based anomaly detection algorithm. Applied to
times series, multivariate and univariate, it is able to identify outliers or anomalies that occur in specific time
periods. Figure 1 presents an overview of our proposed model. First the transformation from the time series
to a graph is illustrated as described in section 3.1. Then the comparison of two or more graphs as described
in section 3.2 and 3.3 is layed out.
3.1. Transformation

To apply the graph-based algorithm to time series data, the sequential data of the multivariate time series \( x = x_1, x_2, \ldots, x_T \) of length \( T \) has to be transformed into graphs. Here and in the following we always use small letters and italic font, when referring to a vector. The only exception we make, is when we deal with the \textit{signature vectors} in subsection 3.3, where we use an arrow on top of the vector in order to indicate, that is a different vector which is not of the same dimension. The transformation of the time series takes place in non-overlapping windows \( w = x_{i-n+1}, x_{i-n+2}, \ldots, x_i \) of size \( n \). In order to avoid conflict of the indexes, we omitted the index \( i \) for the window \( w \), whenever we deal with the elements within a single window. The size \( n \) of the window \( w \) depends on the use case and can be seen as a hyperparameter. When dealing with real-time data, the first step is to wait until \( n \) data points have been recorded. For the transformation to a graph structure, we use the method presented in [13]. For this purpose, a distance measure between the respective elements of a window \( w \) is calculated. Assume \( x_i \) and \( x_j \) are two elements of a window \( w \), which might even be \( d \)-dimensional with \( d > 1 \). The distance between the two elements \( x_i \) and \( x_j \) is calculated with the Minkowski distance as follows, where \( k \) runs over the dimensions of each \( x_i \) and \( x_j \),

\[
D_{ij} = \left( \sum_{k=1}^{d} |x_{ik} - x_{jk}|^p \right)^{1/p}
\]  

with either \( p = 2 \) or \( p = 3 \) see also figure 1.

The \( n \times n \) matrix \( D \) with elements \( D_{ij} \) describes the distance matrix of window \( w \). At the same time, the matrix \( D \) is the adjacency matrix of a graph \( G \). The distances represent the edge weights of \( G \). In our algorithm each window has the same length \( n \), since otherwise the features calculated in each graph are not comparable. Windows of different lengths might be possible, but it remains unclear how to determine the different lengths in one algorithm. Depending on the use case, applying another distance measure can be reasonable and our usage of the Minkowski distance could be seen as exemplary.

3.2. Feature extraction on graphs

Our algorithm is closely related to the NetSimile algorithm in [7] comparing the similarity between two given graphs. The similarity of two graphs in the NetSimile algorithm is based on a set of structural features i.e. the number of neighbors for each node in the \textit{ego network}, the number of outgoing edges for each ego network. We use here the definition of [8], where an \textit{ego network} is defined as a subgraph consisting of a node together with its neighboring nodes and their corresponding edges.
and not calculate these features in an ego network, but directly construct for each graph.

The variable window must deviate from another window to be classified as an anomaly. By only including
graphs in the determination of the threshold $z$.

In order to calculate a threshold for the outlier score $t$, we calculate a mean value $\mu$ and a standard deviation $\sigma$ of the previous $m$ outlier scores and set:

$$ t = \mu \pm z \cdot \sigma. $$

The variable $z$ is a hyperparameter, which is usually taken to be $z \geq 1$. It controls how much a time series window must deviate from another window to be classified as an anomaly. By only including $m$ preceding graphs in the determination of the threshold $t$, the algorithm can forget information. The hyperparameter $m$ is therefore called forgetting factor. This allows our algorithm to handle structural changes in the data.

### Table 1. Content of our optimized feature vector with $v_i$ being the weight of an edge.

| Feature                          | Formula                                      |
|----------------------------------|----------------------------------------------|
| Sum of the edge weights of a node | $\sum_{i=1}^n v_i$                          |
| Arithmetic mean of the edge weights of a node | $\frac{1}{n} \sum_{i=1}^n v_i$              |
| Geometric mean of the edge weights of a node | $\sqrt[\sqrt{\sqrt{n}}]{\prod_{i=1}^n v_i}$ |
| Geometric mean of the edges with the 10% highest edge weights. | $\sqrt[\sqrt{\sqrt{n}}]{\prod_{i=1}^n |v_i|}$ |
| Geometric mean of the edges with the 20% highest edge weights. | $\sqrt[\sqrt{\sqrt{n}}]{\prod_{i=1}^n |v_i|}$ |

However, the original NetSimile algorithm works on unweighted graphs and can therefore not be used on our transformed data set. If we leave out the weights in the graphs $G$ developed in subsection 3.1, the graphs of each window $w$ can be considered identical as can be seen in figure 1. If we then use the original NetSimile features for each ego network, this results in identical features for each respective graph. In order to run the algorithm on weighted graphs as well, we extend the NetSimile algorithm [7] with features, that are calculated with each edge weight.

In a graph, which are presented in table 1. In our case they are identical with the distances $D_{ij}$ calculated in equation (1). Also in contrast to the original NetSimile algorithm, we do not calculate these features in an ego network, but directly construct for each graph $G$ a signature vector $\tilde{s}_G$, that consists of the five aggregated features median, mean, standard deviation, skewness and kurtosis of the features in table 1 for each matrix $D$ or graph $G$. A detailed description of the method of signature vectors is provided in subsection 3.3.

### 3.3. Outlier score

As in [7] we use as a distance measure for outlier detection the Canberra distance of two signature vectors $\tilde{s}_{G_i}$ and $\tilde{s}_{G_j}$ as it is presented in equation (2)

$$ d(\tilde{s}_{G_i}, \tilde{s}_{G_j}) = \frac{1}{25} \sum_{k=1}^{25} \frac{|s_{k,G_i} - s_{k,G_j}|}{|s_{k,G_i}| + |s_{k,G_j}|}. $$

Equation (2) represents at the same time the distance between two graphs $G_i$ and $G_j$ as it is indicated in figure 1. Here $k$ represents the index of each component of the aggregated feature vector of a graph, where all components sum up to 25. The value is 25, as in our implementation we used 5 features as presented in table 1 and for each of these features 5 aggregations as explained in subsection 3.2. Thus a signature vector with 25 components for each graph $G$ is calculated.

In order to calculate an outlier score the Canberra distance of the signature vectors is calculated between the current graph $G_i$ and the previous $m$ graphs $G_{i-1}, G_{i-2}, \ldots, G_{i-m}$ i.e. we obtain the $m$ distances

$$ d(\tilde{s}_{G_{i-m}}, \tilde{s}_{G_i}), d(\tilde{s}_{G_{i-m+1}}, \tilde{s}_{G_i}), \ldots, d(\tilde{s}_{G_{i-1}}, \tilde{s}_{G_i}). $$

The outlier score $o_i$ is then calculated as the average of these $m$ distances

$$ o_i = \frac{1}{m} \sum_{k=0}^{m-1} d(\tilde{s}_{G_{i-m+k}}, \tilde{s}_{G_i}). $$

In order to calculate a threshold $t$ for the outlier score $o_i$ we calculate a mean value $\mu$ and a standard deviation $\sigma$ of the previous $m$ outlier scores and set:

$$ t = \mu \pm z \cdot \sigma. $$

The variable $z$ is a hyperparameter, which is usually taken to be $z \geq 1$. It controls how much a time series window must deviate from another window to be classified as an anomaly. By only including $m$ preceding graphs in the determination of the threshold $t$, the algorithm can forget information. The hyperparameter $m$ is therefore called forgetting factor. This allows our algorithm to handle structural changes in the data.
4. Data

To investigate our algorithm on times series, we use two data sets. One is the numenta anomaly benchmark (NAB)\(^1\), and the other is the KPI-anomaly-detection data set of the 2018 AIOPs competition\(^2\).

4.1. Numenta anomaly benchmark

The NAB consists of real and synthetic time series data. The benchmark was developed to meet real-time streaming applications. Thereby, all time series in the NAB are labeled. The data sets in the NAB are univariate time series [17]. These data are well suited to study the conduct of our algorithm on one-dimensional data. We examine a total of 10 different anomaly types from the synthetic time series part of the NAB data set such as signal drift, individual peaks, rising amplitude, signal jumpsdown, signal jumpsup small noise, increasing noise, sequence change, signal flattmiddle, signal jumpsup, signal nojump, where the anomaly types signal drift, individual peaks, rising amplitude, signal jumpsup small noise, increasing noise, sequence change are data sets similar to the NAB data sets that are constructed for anomaly detection. Since the synthetic times series in the NAB data set are univariate time series, we also expanded them to multivariate times series by adding an additional dimension with the same time series type but without outlier (cf appendix A) in order to evaluate our algorithm also on two-dimensional time series.

4.2. 2018 AIOps’s KPI-anomaly-detection

The 2018 AIOps’s KPI data set comprises time series data collected from various existing internet companies. It consists of key performance indicators (KPIs) and ground truth labels. These KPIs are web service KPIs, more precisely performance metrics that represent the quality and scale of a web service. The metrics are, e.g. the page response time, the number of page views and the occurrence of connection errors.

5. Experimental evaluation

To investigate the performance of our novel algorithm we applied it on both data sets described in section 4.

5.1. Numenta anomaly benchmark

We performed tests on one-dimensional and two-dimensional time series. In both cases, a window size \(n = |\omega| = 288\) was chosen for the respective non-overlapping windows \(\omega\). The value for \(n\) corresponds to the periodicity of the different time series. For the threshold in equation (5) \(z = 3\) is selected.

Thus, values that deviate from the mean by three times the standard deviation are categorized as anomalies. The calculation of an outlier score is based on the previous graphs with \(m = 5\). For this reason, it is not possible to calculate an outlier score for the first five windows of a time series, since not enough values are available to calculate the distances in equation (4). Likewise, the calculation for the next five graphs is not reliable, as the value 0 is included in the computation of \(\mu\) and \(\sigma\). In the original time series of the NAB data set, individual time points are labeled as anomalies. However, our algorithm classifies windows as outliers. Because of this, the outlier indicator in figure 2 spans a whole window from beginning to start although in an online application this indicator would be calculated at the end of each window.

Figure 2 illustrates the results for the one-dimensional data. It can be seen, that the starting points of the anomalous behavior are detected in each of the 10 examples. Whereas, whenever the anomalies span multiple windows the end of the anomalies is often not detected accurately, because the algorithm already adapts to the new behavior.

For all of the respective times series in the NAB data set we also calculated in table 2 an F1-score for outlier detection, in order to compare our results with existing models in the literature i.e. [17, 18]. However, only the NAB time series signal nojump, signal jumpsup, signal jumpsdown and signal flattmiddle are contained in the original NAB dataset and evaluated in [17, 18].

Thus, our graph based model performs on the original NAB dataset, when looking on the F1-score, most of the time better than 0.8 which is the F1-score obtained in [17] or [18] for the best deep learning model. ARIMA models perform in both papers only with an F1-score less than 0.36. Additionally, we tested our algorithm on the two-dimensional NAB data as explained in subsection 4.1. As can be seen in figure 3(a), the same outliers as in the one-dimensional case are detected. Compared to the one-dimensional case, only the outlier score is lower, e.g. the outlier score for the ‘signal jumpsup’ anomaly is 80. In contrast, in the two-dimensional case,

---

\(^1\) This data set is available on https://github.com/numenta/NAB.

\(^2\) More information about the data set can be found on http://iops.ai/competition_detail?competition_id=5&flag=1.
Figure 2. Outlier detection on NAB data sets.

Table 2. $F_1$-scores for the NAB data set.

| Time series                        | Prec | Recall | $F_1$-score |
|------------------------------------|------|--------|-------------|
| Signal drift                       | 1.00 | 0.23   | 0.38        |
| Individual peaks                   | 0.01 | 0.50   | 0.01        |
| Rising amplitude                   | 0.92 | 0.32   | 0.47        |
| Signal jumpsdown                   | 1.00 | 0.72   | 0.84        |
| Signal jumpsup small noise         | 1.00 | 0.72   | 0.84        |
| Increasing noise                   | 0.16 | 0.17   | 0.16        |
| Sequence change                    | 1.00 | 0.23   | 0.38        |
| Signal flatmiddle                  | 0.70 | 0.50   | 0.58        |
| Signal jumpsup                     | 1.00 | 0.72   | 0.84        |
| Signal nojump                      | 1.00 | 0.72   | 0.84        |

Figure 3. Outlier detection on 2D NAB data set.

5.2. 2018 AIOps’s KPI-anomaly-detection

The 2018 AIOps’s KPI-anomaly-detection data set is a web service KPI data set that consists of performance metrics of web services (cf subsection 4.2). This data set comprises 15 different time series. Each of the 15 time series is one-dimensional. As described in subsection 5.1, our algorithm always detects a whole window as an anomaly in contrast to detecting only a point anomaly. The algorithm thus detects if an abnormal data...
Table 3. Hyperparameters for the web service KPI data set.

| Window size $n$ | Forgetting factor $m$ | Threshold $z$ |
|-----------------|-----------------------|---------------|
| 200             | 2                     | 100           | 1.5          |

Table 4. Accuracy scores for the web service KPI.

| Web service KPI Index | Prec | Recall | $F_1$ | # Outliers |
|-----------------------|------|--------|-------|------------|
| 1                     | 85.71| 69.76  | 77.74 | 43         |
| 2                     | 94.28| 78.57  | 86.42 | 42         |
| 3                     | 83.33| 53.84  | 68.58 | 65         |
| 4                     | 62.50| 30.0   | 46.25 | 100        |
| 5                     | ---  | ---    | ---   | 0          |
| 6                     | 3.84 | 100.00 | 51.92 | 2          |
| 7                     | 26.08| 42.85  | 34.47 | 28         |
| 8                     | 33.33| 47.36  | 40.35 | 19         |
| 9                     | 24.00| 36.36  | 30.18 | 33         |
| 10                    | 100.00| 58.13 | 79.06 | 43         |
| 11                    | ---  | ---    | ---   | 0          |
| 12                    | 60.00| 42.85  | 51.42 | 63         |
| 13                    | 88.57| 79.48  | 84.02 | 39         |
| 14                    | ---  | ---    | ---   | 0          |
| 15                    | 51.35| 43.18  | 47.26 | 44         |
| Ø                     | 59.41| 56.86  | 58.14 | 34.73      |

Figure 4. Example outlier detection on the AIOps’s data set.

(a) Outlier detection of Web Service KPI Index 1  
(b) Outlier detection of Web Service KPI Index 2

point occurs within a certain window. Furthermore, our algorithm needs a certain time to adapt to each time series, it therefore cannot detect anomalies occurring at the beginning of a time series. The length of this starting part of the time series is twice the forgetting factor $m$. It is not taken into account when calculating the accuracies in table 4. The list in table 3 consist of the parameters for the experiments performed with the 2018 AIOps’s KPI-anomaly-detection data set.

The accuracies achieved for the respective time series can be found in table 4. An outlier is declared as detected if the corresponding window is classified as an outlier. In figure 4 two results for outlier detection in the web service KPI data set are shown as examples.

The results for the web service KPI are at an acceptable level. The requirement of the competition was an $F_1$ score of 0.55. This requirement is satisfied by our algorithm. The time series with the index 5, 11 and 14 do not contain any labeled outliers and our algorithms does not detect any outliers. Therefore our algorithm can be assumed to work correctly on these time series, too. We use the same hyperparameters for each of the 15 times series leading to acceptable results for all of them. Better results might be possible by using different hyper parameters for each of the 15 time series.
6. Conclusion

In this paper, we present a novel algorithm for outlier detection in data streams. This algorithm is characterized by dynamically transforming the time series into graphs before analyzing the respective signature vectors. The transformation leads to a new application area of the original NetSimile algorithm [7]. In addition, we developed an extension of NetSimile to apply it to weighted fully-connected graphs. Our algorithm uses only a few hyperparameters, therefore making it easy to calibrate. In addition our algorithm is able to learn structural changes in the time series after a certain amount of window sizes as explained in subsection 3.3. Furthermore, the algorithm can be extended or reduced by any number of features, as described in table 1, depending on the use case. This makes the algorithm very flexible.

When applying the algorithm to the numenta benchmark data set, excellent results were achieved. On the one hand, all different outlier types are recognized. On the other hand, they are recognized not only in the one-dimensional but also in the two-dimensional data sets. Our algorithm also achieves acceptable results with the AIOps’s KPI-anomaly-detection data set. In a future work we plan to extend the algorithm to provide an outlier score for individual data points in a time series.

Data availability statement

The data that support the findings of this study are openly available at the 2018 AIOps Challenge at Tsinghua University under http://iops.ai as well as from the Numenta Anomaly Benchmark datasets under https://github.com/numenta/NAB.

Appendix A. 2D results numenta
ORCID iDs

Gabriele Stefanie Gühring  https://orcid.org/0000-0003-3926-7872

References

[1] Alliance B T S 2015 What’s the big deal with data? https://data.bsa.org/
[2] Blázquez-García A, Conde A, Mori U and Lozano J A 2021 A review on outlier/anomaly detection in time series data ACM Comput. Surv. 54 56
[3] Akoglu L, Tong H and Koutra D 2015 Graph based anomaly detection and description: a survey Data Min. Knowl. Discov. 29 626–88
[4] Ranshous S, Shen S, Koutra D, Harenberg S, Faloutsos C and Samatova N 2015 Anomaly detection in dynamic networks: a survey Wiley Interdiscip. Rev.: Comput. Stat. 7 223–47
[5] Xing L, Demertzis K and Yang J 2019 Identifying data streams anomalies by evolving spiking restricted Boltzmann machines Neural Comput. Appl. 32 6699–713
[6] Gupta M, Gao J, Aggarwal C C and Han J 2014 Outlier detection for temporal data: a survey IEEE Trans. Knowl. Data Eng. 26 2230–67
[7] Berlingerio M, Koutra D, Eliassi-Rad T and Faloutsos C 2012 NetSimile: a scalable approach to size-independent network similarity (arXiv:1209.2684)
[8] Akoglu L, McGlohon M and Faloutsos C 2010 Oddball: spotting anomalies in weighted graphs Advances in Knowledge Discovery and Data Mining (Berlin: Springer) pp 410–21
[9] Kang U, Lee J-Y, Koutra D and Faloutsos C 2014 Net-ray: visualizing and mining billion-scale graphs Advances in Knowledge Discovery and Data Mining (Berlin: Springer) pp 348–61
[10] Eberle W and Holder L 2007 Discovering structural anomalies in graph-based data 7th IEEE Int. Conf. Data Mining Workshops (ICDMW 2007) (Piscataway, NJ: IEEE)
[11] Xu X, Yuruk N, Feng Z and Schweiger T A J 2007 SCAN: a structural clustering algorithm for networks Proc. 13th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining KDD ‘07 (New York: Association for Computing Machinery) pp 824–33
[12] Amil P, Almeira N and Masoller C 2019 Outlier mining methods based on graph structure analysis Front. Phys. 7 194
[13] Cheng H, Tan P, Potter C and Klooster S 2008 A robust graph-based algorithm for detection and characterization of anomalies in noisy multivariate time series 2008 IEEE Int. Conf. Data Mining Workshops pp 349–58
[14] Bhatia S, Hooi B, Yoon M, Shin K and Faloutsos C 2020 Midas: microcluster-based detector of anomalies in edge streams Proc. AAAI Conf. Artificial Intelligence vol 34 pp. 3242–9
[15] Cha S H 2007 Comprehensive survey on distance/similarity measures between probability density functions Int. J. Math. Models Methods Appl. Sci. 1 300–7
[16] Ahmad S, Lavin A, Purdy S and Agha Z 2017 Unsupervised real-time anomaly detection for streaming data Neurocomputing 262 134–47
[17] Däubener S, Schmitt S, Wang H, Bäck T and Krause P 2019 Anomaly detection in univariate time series: an empirical comparison of machine learning algorithms 19th Industrial Conf. Advances in Data Mining (ICDM 2019) pp 161–75
[18] Geiger A, Liu D, Alnegheimish S, Cuesta-Infante A and Veeramachaneni K 2020 Tadgan: time series anomaly detection using generative adversarial networks 2020 IEEE Int. Conf. Big Data (Big.Data) pp 33–43