This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3098784, IEEE Transactions on Network and Service Management

TSAGen: Synthetic Time Series Generation for KPI Anomaly Detection

Chengyu Wang*, Kui Wu†, Tongqing Zhou*‡, Guang Yu*, and Zhiping Cai*‡
*College of Computer, National University of Defense Technology, Changsha, 410073, China
†Department of Computer Science, University of Victoria, Canada
‡Corresponding author: {zhoutongqing, zpcai}@nudt.edu.cn

***Abstract*—A key performance indicator (KPI) consists of critical time series data that reflect the runtime states of network systems (e.g., response time and available bandwidth). Despite the importance of KPI, datasets for KPI anomaly detection available to the public are very limited, due to privacy concerns and the high overhead in manually labelling the data. The insufficiency of public KPI data poses a great barrier for network researchers and practitioners to evaluate and test what-if scenarios in the development of artificial intelligence for IT operations (AIOps) and anomaly detection algorithms. To tackle the difficulty, we develop a univariate time series generation tool called TSAGen, which can generate KPI data with anomalies and controllable characteristics for KPI anomaly detection. Experiment results show that the data generated by TSAGen can be used for comprehensive evaluation of anomaly detection algorithms with diverse user-defined what-if scenarios.**

***Index Terms*—time series generation, time series anomaly detection, fault injection, AIOps.**

I. Introduction

LARGE companies that provide Internet-based services, such as online shopping and cloud computing, need to closely monitor the real-time performance of their systems, since a short interruption in networks or quality degradation in services may result in tremendous business losses. The real-time performance indicators (e.g., response time and available bandwidth) are generally collected and stored as time series, named key performance indicators (KPIs[1]) [1]–[11]. To ensure smooth business operations, it is critical for these companies to develop artificial intelligence for IT operations (AIOps) that can accurately detect KPI anomalies and respond with timely troubleshooting.

Detecting KPI anomalies requires substantial efforts in collecting/labelling KPI data for testing the anomaly detection algorithms before their real-world deployment. Unlike traditional time series data such as weather or climate data, KPI data are much larger and requires domain experts of extensive experience to label the KPI data for anomalies. Labeled KPI data are important and much needed no matter whether the detection algorithms are supervised or unsupervised. Supervised algorithms [11], [12] need a lot of labeled data for training and evaluation. Although unsupervised algorithms [1], [8] do not need labeled data in the training stage, they still need labeled data for the purpose of evaluation.

[1]The term KPI has been broadly used in the domain of AIOps to refer to critical performance metrics of networks and services.

Despite the importance of KPIs, very few KPI anomaly datasets [12]–[14] are available to the public. This phenomenon is due to two main reasons. First, manually labelling KPI data requires domain knowledge and is very labor intensive [2], [11]. Second, for privacy and security concerns, IT companies are reluctant to publish KPI data. For the first problem, without a tool, it may takes hours to label a year-long KPI data. Even with the help of advanced labeling tools, such as *Label-Less* [2], it still takes tens of minutes to label it. While with optimistic prediction the first problem may be eventually solved in the future, it is unlikely that the second problem will be alleviated due to the business value of KPI data.

The deficiency of labeled KPI data leads to the following problems in KPI anomaly detection.

- **Insufficient Evaluation:** The insufficiency of evaluation is reflected in two aspects. **(1)** Unlike the computer vision domain where detection algorithms are subject to the test with large-scale benchmark datasets such as ImageNet [15], KPI anomaly detection is usually evaluated with much smaller datasets. As a result, KPI anomaly detection algorithms may achieve great results on some public datasets, but may perform not so good as claimed in production environment. **(2)** Some detection algorithms may be sensitive to specific types of anomalies, but not good at detecting other types of anomalies. Some algorithms are suitable for seasonal KPI anomaly detection, while showing poor performance on other types of KPIs. However, current public datasets rarely distinguish the type of anomalies, let alone provide a fine tuning knob to adjust the severity of a special type of anomaly.
- **Immutable Scenarios:** It is hard to build *what-if* scenarios to evaluate algorithms' performance under hypothetical situations. What-if scenarios are necessary to stress-test the algorithms with an environment that is rare in real world but may cause serious problems if not dealt with. Clearly, KPI data collected from regular service operations are static (i.e., it is hard to change components and characteristics in the collected KPIs) and may not capture these rare events (e.g., sudden drift from normal patterns or anomalies of extremely high density) to allow such stress-tests, let alone conduct variable control experiments to explore the effect of different factors on the performance of detectors.

The above two problems make it extremely challenging for network operators to select and test detectors. If detectors are not thoroughly tested before deployment, we may still end up with service disruptions such as those described in [16]–[18]. Overall, KPI anomaly detection, as one of the main functions of AIOps, is in dire need of a tool that can automatically generate synthetic KPI data to address the above problems. Developing such a tool is not a simple implementation of existing time series generation algorithms such as those in [19]. Instead, it involves non-trivial technical challenges and needs to meet the following requirements:

- Instead of reproducing existing data, generated time series must be **innovative** and **controllable** to enable what-if scenarios. In this regard, we need to control the statistical behavior of the data and generate time series that may have not appeared in the current production environment, but is likely to happen in the future, e.g., a steep increment due to business growth, a traffic spike due to potential attacks, or a high drift due to the holiday effect;
- Injected anomalies must be **diverse** and **effective** to ensure the quality of ground-truth for evaluation, i.e. how to generate a variety of anomalies, how to define/change the degree of anomalies for the benchmark comparison;
- The tool itself must be **modular**, **expandable**, and **tractable** so that the users can easily customize their needs when testing detection algorithms.

To tackle the above technical challenges, we develop a time series generation tool called TSAGen, which meets all the above requirements and enables users to comprehensively evaluate the performance of anomaly detection algorithms with synthetic KPI data of rich characteristics. To meet the first requirement, we apply Random Midpoint Displacement Fractal (RMDF) for the generation of KPI, and propose a feature set for controlling the characteristics of generated KPI. To meet the second requirement, we propose an innovative method to generate diverse anomalies and use Extreme Value Theory (EVT) for adjusting the severity of anomalies. For the last requirement, we adopt an additive modular design and generate each component independently, so that users can plug-and-play individual modules easily.

To sum up, our major contributions are as follows:

- To the best of our knowledge, this is the first work that systematically addresses the special challenges in generating KPI data with anomalies. We innovatively apply RMDF and EVT in the generation of high-quality KPI data with various controllable characteristics to facilitate what-if tests;
- This is the first attempt to define and tackle the **Insufficient Evaluation** and **Immutable Scenario** problems in the KPI anomaly detection domain. TSAGen is the first step towards comprehensive evaluation and what-if test;
- We propose a feature representation method for KPI anomaly detection. These features capture the most important characteristic of KPI, and hence allow users to easily define the desired statistical properties of the synthetic KPI data;
- We open-source TSAGen at a public repository[2]. Also, we provide a public benchmark dataset generated by TSAGen, which can be used for comprehensive evaluation of KPI anomaly detection algorithms[3].

In this paper, we mainly focus on univariate KPI (univariate time series), multivariate KPI (multivariate time series) is left as our future work.

The rest of this paper is organized as follows. In Section II, we review related works. In Section III, we give an overview of TSAGen. In Section IV, we explain how to generate the time series. In Section V, we explain how to generate the anomalies. We evaluate TSAGen in Section VI and discuss relevant issues in Section VII. The paper is concluded in Section VIII.

II. Related Work

A. Time Series Generation

Time series generation is not new and has been broadly studied in various domains, e.g., energy, climate, medical, and financial domains [20]–[26]. Existing methods can be roughly divided into three categories: traditional statistical methods, reconstruction methods, and deep generative methods.

1) Traditional statistical methods: Predicting future data can also be considered as a way of generating data. As such, many time series models like Markov Models, Auto Regressive Integrated Moving Average (ARIMA) [27], and Holt-winters [28] also have the ability to generate data. In [25], Kang et.al. [25] proposed an approach to time series generation, based on the Gaussian mixture autoregressive (MAR) model and parameter optimization with a Genetic Algorithm (GA). In general, these methods are not purposefully designed for KPI data generation for anomaly detection and cannot effectively address the challenges introduced in Section I.

2) Reconstruction methods: These methods are usually dataset-oriented and take historical dataset as input and output new datasets via decomposition and reconstruction [20], [21], [26], [29]. These methods usually do not need a model and generate new data with existing data by shuffle [20], averaging [29], and modification [21]. For instance, Iftikhar et.al. proposed a method [26] to decompose an existing time series by Seasonal and Trend decomposition using Loess (STL), where Loess is a method for estimating nonlinear relationships [30]. After that, it recombines the components to generate new data. These methods can be used to augment sparse datasets, but have a limited power in controlling the generated data, because the generated data are usually the modification and reproduction of the original data. Therefore, this kind of methods is not suitable for the task of KPI anomaly detection.

3) Deep generative methods: Deep generative models like Generative Adversarial Networks (GANs) [31] and Variational Auto-Encoders (VAEs) [32] are well known for their power in generating realistic pictures. They are recently applied in the generation of time series data [22]–[24]. Yoon et.al. proposed TimeGAN [22], a GAN-based framework for generating time

[2]Code is available at https://github.com/AprilCal/TSAGen.

[3]Benchmark dataset is available at https://github.com/AprilCal/TSAGen.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3098784, IEEE Transactions on Network and Service Management

series data, which jointly optimizes both supervised and adversarial objectives to make the network adhere to the statistical features of the training data during sampling. As discussed in [33], it is extremely hard to control the characteristics of generated data using GAN, let alone generate data in a variable-control manner (which requires modifying one feature without distorting other features). This problem roots from the main design goal of these methods: generating simulated data indistinguishable from the training data. This goal is thus more suitable for the task of prediction and classification, but not for the task of anomaly detection.

TSAGen differs from all the above methods: (1) It is specifically designed for the purpose of evaluation of detection algorithms, with much consideration on anomaly generation; (2) It can provide powerful control on the characteristics of generated data; (3) It has the ability to inject various anomalies with control on their position, type and severity; (4) It uses features directly as input, which can work with little data or no data at all.

B. Anomaly Injection

KPI generation is relatively new in KPI anomaly detection domain. There is only a limited amount of literature related to time series anomaly injection. Ren et al. [34] inject synthetic anomalies into real data and use generated anomalies to train a CNN-based detector. They randomly select several points in KPIs and calculate the injection values to replace the original points. This method has a limited power in generating new KPI and can only injects point anomalies based on selected points. It is not designed for building what-if scenarios with variable control. Laptev et al. [12] implemented a time series generator in about 300 lines of Python code, which can inject synthetic anomalies into real data. Nevertheless, the generated data have simple patterns only and are not suitable for comprehensive evaluation of KPI anomaly detection algorithm. Laptev also proposed AnoGen [35], which can deterministically generate time series through sampling from the latent z space as well as anomaly through sampling from the outlier region of the latent z space of a trained VAE. However, this method cannot control the type of generated anomalies and the severity of anomalies. TSAGen differs from the above methods in that it can control both the type and the severity of anomalies.

III. Overview of TSAGen

In this section, we first introduce the motivating scenarios that operators often encounter in practice as well as our design goals. Then we introduce the modeling of KPI. Finally, we present the workflow of TSAGen.

A. Motivating Scenarios and Use Cases

Motivating scenarios: Fig. 1 shows a typical scenario of network and service management. From the perspective of operators, the whole system is divided into two layers, the network layer and the management layer. In the network layer, system and service level indicators are collected from hardware and applications, respectively, then saved as time series data in the management layer. In the management layer, the anomaly detection algorithm monitors the KPI data from the network layer in real time, and notifies operators when an anomaly is detected.

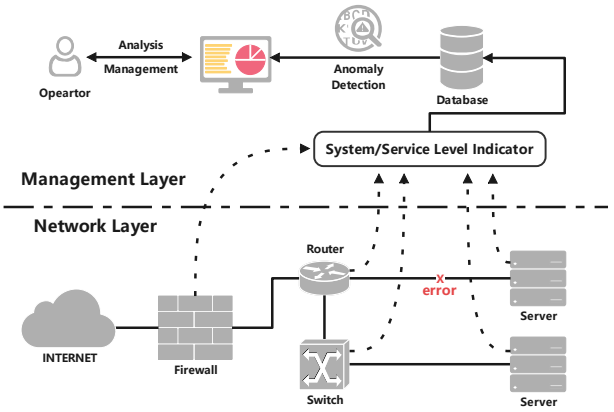


Fig. 1: Motivating Scenarios.

With the rise of AIOps, the pressure of operators has been effectively alleviated: operators do not need to closely monitor these indicators, but instead only need to select appropriate detection algorithms, and timely handle the anomalies reported by the detection algorithms. Nevertheless, due to the **insufficient evaluation** and **immutable scenarios** problems presented in Section I, it is not easy for operators to select an appropriate detector. As reported in [11], it is not uncommon for operators to give up after a few attempts of testing detectors and settle with static, threshold-based detection. If the selected detector is not robust enough or not suitable for the KPI being monitored, we may not avoid disruptions such as those described in [16]–[18]. How to select the proper detector becomes a problem that has plagued operators for a long time. For example, How to select a proper detector for seasonal KPI? How to select a detector to detect abrupt changes? How robust is the selected detector?

We are thus motivated to answer the above call and develop a tractable tool, which benefits network and service management by helping operators to choose appropriate detection algorithms as well as developers to evaluate their algorithms.

Use cases: There are many potential use cases that can benefit from TSAGen. We highlight several typical use cases as follows.

1) Use case 1: Alice wants to select a proper detector for seasonal KPIs, but existing public datasets [12], [13] do not distinguish different types of KPIs. So Alice uses TSAGen to generate a large number of seasonal KPIs and evaluates her algorithm on the generated dataset.

2) Use case 2: Bob wants to detect abrupt changes [8] in his application server, so he wants to evaluate which detector performs better on this kind of anomaly. With TSAGen, he injects a large amount of anomalies of this type into generated data and evaluates the detector on the generated data.

3) Use case 3: Charlie worries about the robustness (e.g. resistance to drift, noise, and missing values) of the detector to be deployed, so he uses TSAGen to generate KPI data with various drift degrees, noise levels, and missing rates without

distorting other components. Quite often, the characteristics in real data are static. To avoid the problem, he needs TSAGen to generate KPI data of dynamic characteristics, with only one feature varying at a time.

B. Modeling of KPI

1) ***KPI dataset****:* KPI is essentially a time series denoted as:

$$\mathbf{x} = (x_1, ..., x_i, ..., x_T), 1 \leq i \leq T, x_i \in \mathbb{R}, T \in \mathbb{N} \quad (1)$$

where x_i is the monitored value at time index i. We use $x_{m:n}$ to represent a continuous subsequence of length $n-m+1$ of time series $\mathbf{x}$ in the rest of this paper.

KPI dataset is a set of time series denoted as:

$$\mathcal{D} = \{\mathbf{x}_j \mid 1 \leq j \leq n, \mathbf{x}_j \in \mathbb{R}^T, n, T \in \mathbb{N}\} \quad (2)$$

where $\mathbf{x}_j$ is time series and j is the index of the time series.

2) ***KPI Components****:* To control the generated KPI and improve the modularity of TSAGen as much as possible, we adopt an additive model, which is a widely used model in time series analysis [36]. By applying the additive model, we do not need to generate time series as a whole; instead, we can generate each component independently and get the ultimate result by combining all the components. To be specific, a KPI time series is represented with several components of the same length in the following form:

$$\mathbf{x}_t = seas_t + tr_t + noise_t \quad (3)$$

where $seas_t,\ tr_t,\ noise_t$ represent the seasonal, trend and noise components, respectively. A time series $\mathbf{x}_t$ is the result of adding the corresponding values of these components. This modular method enables users to control and change each component independently without distorting other components, which is needed for variable control experiments and greatly facilitates what-if tests.

3) ***Time Series Features****:* Time series features are used as the approximate representation of time series. Our goal here is to propose a feature set that can capture important characteristics of KPI, so as to control the behavior of KPI via the feature set. In addition, the features are attributed to different KPI components described above, so that we can control each component through the related feature values.

Given a time series $\mathbf{x}_t$, we have two features associated with the trend component:

$$\theta_1 = level(tr_t),\ \theta_1 \in \mathbb{R} \quad (4)$$

$$\theta_2 = slope(tr_t),\ \theta_2 \in (-\frac{\pi}{2}, \frac{\pi}{2}) \quad (5)$$

where θ_1 denotes the level of trend and θ_2 denotes the slope of trend. We assume that there is a linear trend in $\mathbf{x}_t$, and the trend is essentially a line controlled by level (θ_1) and slope (θ_2). Through tuning θ_1 and θ_2, we can obtain any possible linear trend.

The following three features are associated with the seasonal component of the time series:

$$\theta_3 = amplitude(seas_t),\ \theta_3 \in (0, +\infty) \quad (6)$$

$$\theta_4 = freq(seas_t),\ \theta_4 \in (0, +\infty) \quad (7)$$

$$\theta_5 = cn(seas_t),\ \theta_5 \in (0, +\infty) \quad (8)$$

where $\theta_3, \theta_4, \theta_5$ represent the amplitude, frequency and number of cycles for the $seas_t$ component, respectively. Through tuning θ_3, θ_4 and θ_5, we can control the behavior of seasonal component. In anomaly detection, people care more about the variation in the seasonal components, because if the shapes of the generated cycles are almost identical, then the generated seasonal components may not diverse enough to test anomaly detection algorithms. For this reason, practitioners usually focus on the difference between cycles. In this paper, we use RMDF to generate random cycle shapes. Note that TSAGen allows users to control/adjust the shape of cycles by implementing their own new season generator, e.g., using Fourier series or some other methods to generate desired cycle shapes. More details will be described in Section IV.

The following four features are associated with the noise component of the time series:

$$\theta_6 = \mu = mean(noise_t) = \frac{1}{T}\sum_{t=1}^{T} noise_t \quad (9)$$

$$\theta_7 = std(noise_t) = \sqrt{\frac{\sum_{t=1}^{T}(noise_t - \mu)^2}{T-1}} \quad (10)$$

$$\theta_8 = skew(noise_t) = \frac{T^{-1}\sum_{t=1}^{T}(noise_t - \mu)^3}{(T^{-1}\sum_{t=1}^{T}(noise_t - \mu))^{3/2}} \quad (11)$$

$$\theta_9 = kurt(noise_t) = \frac{T\sum_{T}^{t=1}(noise_t - \mu)^4}{(\sum_{t=1}^{T}(noise_t - \mu)^2)^2} \quad (12)$$

where θ_6, θ_7 represent the mean and the standard deviation of $noise_t$, respectively, θ_8, θ_9 represent the third and fourth standardized moments of $noise_t$, respectively, i.e., skewness and kurtosis, μ denotes the mean of $noise_t$. It is necessary to point out that the type of noise component is not a feature, since we use the Pearson Distribution to model the noise component, the type of noise is determined by it statistical moments. This method is also known as the method of moments [37], [38]. Details will be described in Section IV.

In production environment, KPI may not be strictly seasonal, since the amplitude and frequency of KPI might slowly drift over time. What's more, the seasonal behavior may be affected by the holiday effects [7]. We accommodate this phenomenon in TSAGen by applying two drift factors on amplitude and frequency, respectively:

$$a_i \sim U(1, 1+k_1), 1 \leq i \leq \theta_4, k_1 \in [0, +\infty) \quad (13)$$

$$f_i \sim U(1, 1+k_2), 1 \leq i \leq \theta_4, k_2 \in [0, +\infty) \quad (14)$$

where U is uniform distribution, a_i is the drift factor of amplitude, f_i is the drift factor of frequency. We can apply a_i and f_i on every cycle. The system parameters k_1 and k_2 are used to control the degree of drift, where $k_1 = k_2 = 0$ means that there is no drift in the seasonal component.

All time series features are summarized in Table I. Note that to ease understanding, three more features, recursion depth (d),

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3098784, IEEE Transactions on Network and Service Management

forking depth ($\hat{d}$) and risk (q), will be introduced later after we introduce RMDF and EVT in Sections IV and V, respectively.

TABLE I: Time series features

Feature	Description	value
θ_1	level of trend	$(-\infty, +\infty)$
θ_2	slope of trend	$(-\pi/2, +\pi/2)$
θ_3	amplitude	$(0, +\infty)$
θ_4	frequency	$(0, +\infty)$
θ_5	no. of cycle	$[1, +\infty)$
θ_6	mean	-
θ_7	standard deviation	$[0, +\infty)$
θ_8	skewness	-
θ_9	kurtosis	-
k_1	drift degree of amplitude	$[0, +\infty)$
k_2	drift degree of frequency	$[0, +\infty)$
d	recursion depth	$\mathbb{N}^+$
$\hat{d}$	forking depth	$\mathbb{N}^+, \hat{d} < d$
q	risk	$(0, 1)$

*4) **Meta Features**:* TSAGen is both KPI-oriented and dataset-oriented. When a user want to *quickly* generate a large number of KPI datsets with different feature values, they can use the so-called meta features, i.e., the features that captures the value distributions of the time series features (θ_i and k_j) introduced above. Given a set of θ_i and a set of k_j, we define the meta features:

$$\sigma_i = (max(\theta_i), upper(\theta_i), mid(\theta_i), lower(\theta_i), min(\theta_i)), \quad 1 \le i \le 9 \tag{15}$$

$$\sigma'_j = (max(k_j), upper(k_j), mid(k_j), lower(k_j), min(k_j)), \quad j = 1, 2. \tag{16}$$

where $upper$ and $lower$ represent the upper and lower quartile, respectively, max, min, and mid denote the maximum, minimum, and median, respectively.

Given the above meta features, we can quickly generate a large KPI dataset consisting of many KPIs whose features follow the distribution described by the meta features.

C. Overview of TSAGen

We describe the architecture of TSAGen in Fig. 2. There are three main stages in a complete workflow of TSAGen: KPI generation, anomaly injection, and benchmark evaluation.

In the **KPI generation** stage, TSAGen first takes input features (time series features or meta features) from the operator. Then seasonal, trend and noise components are generated by Season Generator, Trend Generator and Noise Generator, respectively. The Season Generator first generates the shape of a single cycle, and then expands the cycle and adds drifts. In addition, multiple seasonal components can be integrated together by using multiple Season Generators. The Trend Generator generates a linear trend according to the level and the trend slope. The noise component is generated by the Noise Generator using Pearson distribution. At the end of this stage, all components are added together. Details of the generation process are described in Section IV.

In the **anomaly injection** stage, Anomaly Generator is responsible for the injection of anomalies. It first generates random anomalies (if not specified) from predefined anomaly patterns. Then it uses EVT to establish low probability regions and make anomalies fall into these regions. Details are described in Section V.

In the **benchmark evaluation** stage, the evaluated algorithm is compared with benchmark algorithms. Note that the performance of an algorithm alone on synthetic data is not very meaningful, since we cannot draw any conclusion if there are no competing algorithms. Even on real-world KPI data, we also need to compare the target algorithm with competing/benchmark algorithms to draw conclusion. Therefore, we integrate some representative algorithms into TSAGen as benchmark algorithms.

Due to the additive model, our system naturally has good modularity. The generators in TSAGen are independent, allowing users to easily define a new type of anomaly, change the strategy of season generation or replace the noise generator.

IV. Generation of Time Series

Meta feature σ_i is used for the generation of θ_i. In our current implementation, θ_i follows a mixture distribution composed of several uniform distributions, but it is easy to replace with other type of distributions. Given a σ_i, the probability density function of θ_i is as follows:

$$p_i(x) = \begin{cases} \frac{1}{4(max(\theta_i) - upper(\theta_i))} & upper(\theta_i) \le x \le max(\theta_i) \\ \frac{1}{4(upper(\theta_i) - mid(\theta_i))} & mid(\theta_i) \le x < upper(\theta_i) \\ \frac{1}{4(mid(\theta_i) - lower(\theta_i))} & lower(\theta_i) \le x < mid(\theta_i) \\ \frac{1}{4(lower(\theta_i) - min(\theta_i))} & min(\theta_i) \le x < lower(\theta_i) \end{cases} \tag{17}$$

Then we generate θ_i for every KPI through the probability density function p_i. With respect to k_1 and k_2, we allow users to input their values directly for each KPI. Note that different KPIs can use different k_1 and k_2 values. We highlight that the distribution of θ_i is a design choice, which is determined by the needs of users. For example, if users want to simulate the distribution of real data, the distribution should be selected to reflect the real data; If users have no prior knowledge of real data, normal distribution may be an appropriate choice; If users want to generate a balanced dataset, uniform distribution may be a good candidate.

A. Generation of Trend

We use the following formula to generate the trend component:

$$tr_t = \theta_1 + \theta_2 * l_t \tag{18}$$

where $l_t = (1, 2, 3, ...)$ is the sequence of time index.

B. Generation of Season

We need a method that can (1) control the characteristics (i.e., $\theta_3 \sim \theta_5, k_1, k_2$) of the generated cycles, (2) effectively model the normal shape difference between cycles, and (3) facilitates anomaly generation (generate anomaly cycles and tune the severity of anomaly). For these reasons, simply using smooth signals (e.g., sin, cos) does not satisfy the above requirements.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3098784, IEEE Transactions on Network and Service Management

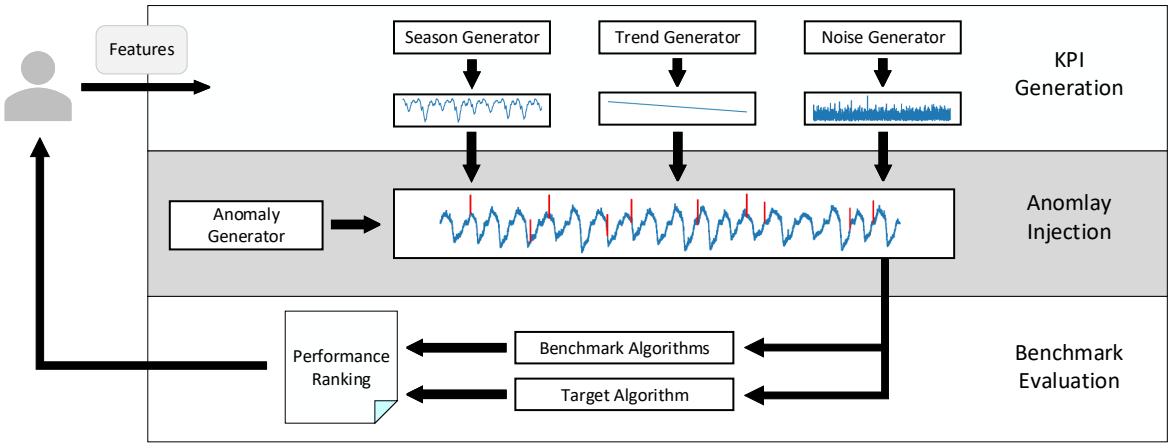


Fig. 2: Overview of TSAGen. There are three main stages in a complete workflow of TSAGen. In the **KPI generation** stage, TSAGen takes as input features from the operator and generates each component independently. In the **anomaly injection** stage, Anomaly Generator is responsible for the injection of anomalies. In the **benchmark evaluation** stage, the evaluated algorithm (i.e., the target algorithm) is compared with benchmark algorithms.

To achieve the above goals, we apply RMDF for the generation of cycle shapes. RMDF is widely used for the generation of topographic map and region boundary. As we will see below, RMDF is a simple but effective tool for the generation of seasonal component.

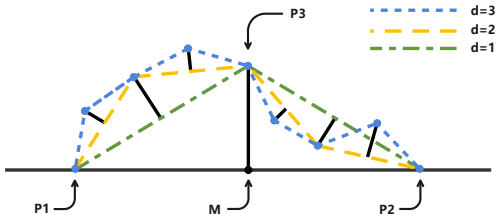


Fig. 3: RMDF process. Green, yellow and blue lines represent the curve generated by RMDF when $d = 1, 2, 3$, respectively.

1) Generating Shape with RMDF: To ease explanation, we use normal font to denote lines and bold font to denote vectors in the following discussion, e.g., A denotes a point, AB denotes a line, $\boldsymbol{AB}$ denotes a vector. The process of RMDF is shown in Fig. 3. Assuming that there are two points P_1 and P_2 and M is the midpoint of line P_1P_2, we can generate a displacement $\boldsymbol{MP_3}$ orthogonal to $\boldsymbol{P_1P_2}$ (i.e., $\boldsymbol{P_1P_2}\cdot\boldsymbol{MP_3} = 0$), where $\|\boldsymbol{MP_3}\|_2$ (i.e., l_2-norm) follows Gaussian distribution. The coordinate of P_3 can be calculated by:

$$(x, y) = \frac{\sqrt{a^2 + b^2}}{b}(\cos\theta, \sin\theta)\boldsymbol{P_1M} \tag{19}$$

where $a = \|\boldsymbol{P_1M}\|_2$, $b = \|\boldsymbol{MP_3}\|_2$, and $\theta = \arctan\frac{b}{a}$. In this way, we obtain a sequence of points P_1, P_3, P_2, which are also called control points. Through connecting these control points from left to right, we can get a continuous function consisting of two lines (Fig. 3, green lines). Recursively repeating the above process on P_1P_3 and P_3P_2, respectively, we can obtain random cycles of diverse shape. The variance of the displacement's l_2-norm, named s, must decrease over recursion depth to ensure a relatively smooth shape. Here we initialize s to $\frac{\|\boldsymbol{P_1P_2}\|_2}{4}$ (i.e., 1/4 when $P_1 = (0, 0), P_2 = (1, 0)$), and make s decrease by 2 times each recursion. Constraints must be taken to ensure the generated curve is a function. For instance, when the random displacement accidentally has an extremely high value, the generated curve may not be function, e.g., a control point P happened to appear at the left of P_1. Complete process of RMDF is described in Algorithm 1.

Algorithm 1 RMDF Procedure

Input: Recursion depth d
Output: Control points

procedure RMDF(P_1, P_2, queue, depth, D)
  if depth $\geq$ D **then**
    queue.push(P_1);
    return;
  $M \leftarrow midpoint(P_1, P_2)$; ▷ get mid point
  sample an $l \sim \mathcal{N}(0, 1/4)$
  $l \leftarrow l/2^{depth}$ ▷ decrease the variance of l
  generate an $\boldsymbol{MP}$ s.t. $\boldsymbol{P_1P_2}\cdot\boldsymbol{MP_3} = 0, \|\boldsymbol{MP}\|_2 = 1$
  $\boldsymbol{MP}_3 \leftarrow l \times \boldsymbol{MP}$
  $a \leftarrow \|\boldsymbol{P_1M}\|_2$
  $b \leftarrow \|\boldsymbol{MP_3}\|_2$
  $\theta \leftarrow \arctan(b/a)$
  $P_3 \leftarrow \frac{\sqrt{a^2+b^2}}{b}(\cos\theta, \sin\theta)\boldsymbol{P_1M}$; ▷ formula (19)
  $depth \leftarrow depth + 1$;
  RMDF(P_1, P_3, queue, depth, D);
  RMDF(P_3, P_2, queue, depth, D);
$start \leftarrow (0, 0)$;
$end \leftarrow (0, 1)$;
init(queue);
$depth \leftarrow 0$;
$D \leftarrow d$;
RMDF(start, end, queue, depth, D);
queue.push(end);
return queue;

Recursion depth d controls the diversity of the generated shapes. As shown in Fig. 4, with the increment of d, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3098784, IEEE Transactions on Network and Service Management

generated curve tends to include more small oscillations. Because of the recursive nature of RMDF, the total number of control points required (denoted as n) will double if d increases by 1 (i.e., $n = 2^0 + 2^1 + \cdots + 2^{d-1} + 2 = 2^d + 1$). Thus the computational complexity of Algorithm 1 is $O(2^d)$. When d is small, the generated curve may be too simple; when d becomes larger, the computational time increases but the generated curve shows a richer shape. To balance the diversity of the curve and the computation efficiency, we use $d = 10$ for simulating KPI data in our later experiments. Besides, from the result of Section VI-E, RMDF forking can generate curves with enough diversity when $d = 10$. Thus, the exponential complexity is not a problem for the practical use of TSAGen.

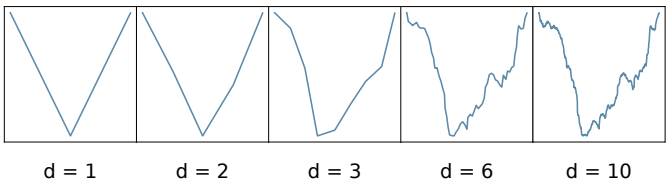


Fig. 4: Effect of recursion depth d. With the increment of d, the generated curve tends to include more small oscillations.

Through the RMDF process, we can eventually obtain a sequence of control points, and through connecting these control points, we can obtain a continuous function $F(x)$ composed of several lines. The cycles are generated by sampling on $F(x)$, and the seasonal component is generated by connecting these cycles. We standardize the frequency of cycle through making $P_1 = (0, 0), P_2 = (1, 0)$ and the amplitude of cycle with scaling. Thus we can control the frequency of season through tuning the sampling number on $F(x)$ as well as the amplitude of season via multiplying $F(x)$ by a scalar.

2) Generating Seasonal Component: After we generate cycles using the above methods, we then build the seasonal component with the generated cycles. To explain how the seasonal component is produced, we define $cycle_{t,s,l}$ as a single cycle of $seas_t$, where l denotes the length of the cycle, s indicates its order. Define $\bigoplus$ as connection operation which concatenates two time series, $\sum\bigoplus$ means multiple connections, which concatenate multiple cycles in order. Thus the seasonal component can be denoted as:

$$seas_t = \sum_{s=1}^{\theta_5} \bigoplus (\theta_3 * a_s) cycle_{t,s,f_s * 1/\theta_4} \tag{20}$$

where f_s and a_s are the drift factors defined in Section III-B1. θ_4 and θ_5 are amplitude and frequency, respectively. The above formula explains how the season is produced, how the amplitude and frequency is controlled, and how the drift is added.

3) Modeling Shape Differences and Anomalies: To model the small shape difference between cycles, all shapes can be generated from a certain depth during the RMDF process. For example, during the cycle generation process, initially all cycles can share the same shape when $d < 8$, and their ultimate shape (when $d = 10$) is generated with two more recursions respectively, based on the shared shape. In this case, $\hat{d} = 2$ is called the forking depth, $d^* = d - \hat{d} = 8$ is called shared depth. We call this method **RMDF forking**.

To explain how the difference is generated by RMDF forking, let $\mathbf{c}^i = \{c_i\}_{i=1}^{2^i+1}$ be the set of control points generated by a RMDF process in the i-th recursion, where $i \in \{1, 2, \ldots, d\}$. It is obvious that $|\mathbf{c}^j| = 2^j + 1$, $\mathbf{c}^j \subset \mathbf{c}^{j+k}$ for $j, k \in \mathbb{N}^+, j+k \leq d$, thus we have $|\mathbf{c}^{j+k} - \mathbf{c}^j| = |\mathbf{c}^{j+k}| - |\mathbf{c}^j| = (2^{j+k} + 1) - (2^j + 1) = 2^j(2^k - 1)$.

For simplicity and without loss of generality, let $\mathbf{c}_1^d, \mathbf{c}_2^d$ be two sets of control points generated by a RMDF process in the last recursion with forking depth $\hat{d}$, the set of their shared control points can be denoted as $\mathbf{c}_1^d \cap \mathbf{c}_2^d = \mathbf{c}^{d^*}$, thus the number of different control points (denoted as m) between the two forking curves can be calculated by:

$$m = |\mathbf{c}_1^d - \mathbf{c}_2^d| = |\mathbf{c}_2^d - \mathbf{c}_1^d| = |\mathbf{c}^d - \mathbf{c}^{d^*}| = 2^d - 2^{d-\hat{d}} \tag{21}$$

Based on Equation (21), with the increment of $\hat{d}$, the number of different control points between two forking curves increases as well, indicating that the difference between the two curves becomes larger.

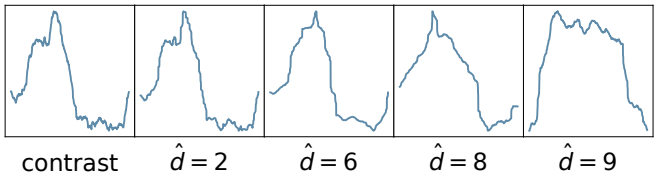


Fig. 5: RMDF forking. With the increment of forking depth $\hat{d}$, the generated cycles tend to be more and more different from the contrast.

As shown in Fig. 5, the greater the forking depth, the larger the difference between the generated curve and the contrast curve. Thus RMDF forking is not only a good method to generate random shapes, but also can effectively model the shape difference between cycles. Besides the benefits mentioned above, RMDF forking also benefits anomaly generation, which will be further discussed in Section V and evaluated in Section VI.

C. Generation of Noise

For evaluation purpose, it is desired that a method can (1) control the statistical properties of noise component easily, (2) benefit anomaly generation and injection (e.g., injecting an anomaly by varying the noise type or moment of a segment), and (3) generate various noise types. As reported in [3], using a single distribution like Gaussian noise cannot achieve the third goal.

Given the above consideration, we use the Pearson distribution [37], [38] for the generation of noise component, due to its capability of controlling the statistical proprieties of noise component. We can generate noise component that meets user-specified features (i.e., $\theta_6 \sim \theta_9$), which are much needed for what-if tests.

The Pearson distribution is a *family* of continuous probability distributions (e.g., normal, gamma, beta and so on). For a given combination of mean, standard deviation, skewness, and kurtosis, we can calculate a probability density function (PDF)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3098784, IEEE Transactions on Network and Service Management

with given statistical moments through the method described in [37], [38]. Then we can generate noise component through the PDF.

By applying the Pearson Distribution in TSAGen, users do not need to determine the concrete type of the noise component. Instead, they only need to give the combination of mean, standard deviation, skewness, and kurtosis to describe what the distribution of the noise looks like, based on which TSAGen can automatically determine the type of distribution. The generation process of noise component can be denoted as follows:

$$noise_t = pearson(mean, sigma, skew, kurt)$$

V. Generation of Anomaly

A. Definition of Anomaly

Anomaly is defined in contrast to normal KPI: An anomaly is a point or segment that deviates significantly from the rest of the data. Clearly, anomalies depend on specific application context and user-defined thresholds, and a large variety of anomalies and criteria for classifying the anomalies have been studied [12], [39], [40]. From the perspective of evaluation of anomaly detection methods, practitioners mainly focus on how well a detector can capture the changes in KPIs. Generally speaking, a detector could be considered as a classifier for data exhibiting certain patterns, i.e., data following the underline patterns are classified as normal and otherwise anomaly.

Overall, we need a tool that (**R1:**) generates different kinds of anomalies, and (**R2:**) includes tuning knobs to vary the severity/degree of these anomalies. TSAGen meets all the above two requirements. With the data generated with TSAGen, we can easily test whether or not a detector can capture the correlation between observations (i.e., $p(x_t|x_{1:t-1})$) or the long term patterns of observations (i.e., $p(x_{t-w:t}|x_{1:t-w-1})$). In addition, since we know the underline model parameters impacting the generated data, we can easily test the performance of a detector and find the root causes of the performance change. Aligning with **R1** and **R2**, we innovatively introduce anomalies into two categories: **non-structural anomaly** and **structural anomaly**. The former are generated by superposition of patterns pre-defined by users, and the latter are special types of anomalies that cannot be generated by adding pre-defined patterns. The latter category can be further classified into shape-based anomalies and statistical moment-based anomalies.

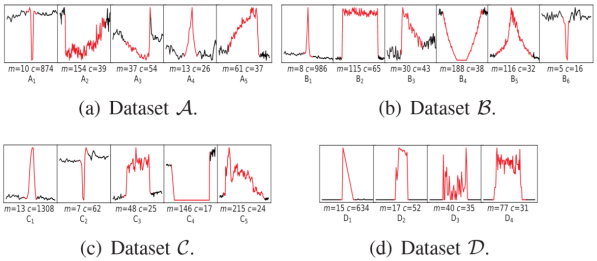
(1) Non-structural anomaly is the most common kind of anomalies and is characterized by at least one abnormally high or low value inside it. Non-structural anomalies are usually related to external events, e.g., an attack, and persist until the external event ends. Such events usually can be modeled with pre-defined patterns. As shown in Fig. 6, the anomaly can appear in the form of a point or segment. Sometimes it can also be context-relevant, for instance, an attack does not result in a significant increase in the KPI value, but the value is much higher than values at the same time of past days. To detect a non-structural anomaly, an algorithm at least needs to model $p(x_t|x_{1:t-1})$, where $x_{m:n}$ represents a continuous subsequence (of length $n-m+1$) in KPI x.

(2) Structural anomaly is characterized by the change in the inherent structure of KPI, e.g., cycle shapes, cycle length or statistical moments. This kind of anomalies is more difficult to detect since they can appear in various forms. Depending on the component they occur, this kind of anomalies can be generally classified into shape-based anomalies (which occur in seasonal component) and statistical moment-based anomalies (which occur in noise component). In this case, modeling $p(x_t|x_{t-1})$ usually does not work well. To detect this kind of anomalies, an algorithm at least needs to efficiently model $p(x_{t-w:t}|x_{1:t-w-1})$.

Rationale for the above classification: There are no industrial standards when comparing the performance of different anomaly detection algorithms. The commonly-used metrics, such as accuracy, recall, precision, only disclose the performance of a detection algorithm *w.r.t. a certain KPI data*. As the statistical features of KPI vary, the relative performance of different detection algorithms may change [34]. In this sense, we believe the best way to objectively evaluate a detection algorithm is to evaluate whether or not it can capture the underline statistical feature changes. In other words, a detection algorithm inherently has the capability of data modeling, and as such we should evaluate their performance by testing their modeling capability. Our classification of anomalies facilitates such test. For instance, an algorithm that only models $p(x_t|x_{t-1})$ is unlikely to detect a structural anomaly, and an algorithm that models $p(x_{t-w:t}|x_{1:t-w-1})$ might not work well on non-structural anomaly. As we will see in Section VI-H, using this classification we can effectively distinguish detection algorithms by their modeling capability.

Note that it is impossible to pre-define all anomaly patterns in production environment, since they might be related to unknown attacks or faults. We do not aim at pre-defining all possible forms of anomalies, but try to present a set of commonly-known anomaly patterns and enable users to define new types of anomalies at the same time. In addition, with TSAGen's modular design, users can use it to easily generate new patterns that are unknown to current production environment (i.e., what-if test).

B. Implementation of Anomaly



(a) Dataset $\mathcal{A}$. (b) Dataset $\mathcal{B}$.

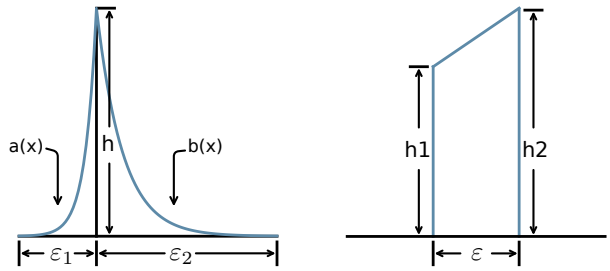
(c) Dataset $\mathcal{C}$. (d) Dataset $\mathcal{D}$.

Fig. 6: Non-structural anomaly templates found by Label-Less [2], where m is the length of the template and c is the number of similar anomalies.

(1) Non-structural anomaly. Inspired by Zhao et al. [2], we find that despite various forms of non-structural anomalies, the shapes of the anomalies themselves are relatively stable. Fig. 6 shows all anomaly templates retrieved from four datasets

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3098784, IEEE Transactions on Network and Service Management

by Label-Less [2] with a high recall, where m is the length of anomalies and c is the number of similar anomalies. It is easy to find that the diversity of non-structural anomalies is due to the various length, degree, and the complexity of the context. In addition, most non-structural anomalies can be characterized as the superposition of several known patterns.



(a) Anomaly pattern of Type I (b) Anomaly pattern of Type II

Fig. 7: Pre-defined patterns for non-structural anomaly.

According to this observation, we try to define and implement non-structural anomalies based on their shape, so that more anomalies can be covered with little effort. As we will see below, a variety of non-structural anomalies can be derived from our predefined patterns.

- As shown in Fig. 7 (a), anomaly pattern of type I is defined as:

$$\Gamma_1(\varepsilon_1, \varepsilon_2, h, a, b, x) = \begin{cases} a(x) & 0 \le x < \varepsilon_1 \\ b(x) & \varepsilon_1 \le x \le \varepsilon_1 + \varepsilon_2 \end{cases}$$

  where $\varepsilon_1, \varepsilon_2 \in \mathbb{N}$, $\varepsilon_1 + \varepsilon_2 \neq 0$, $h \in (-\infty, 0) \bigcup (0, +\infty)$, $x \in \{0, 1, 2, \ldots, \varepsilon_1 + \varepsilon_2\}$. $a(x)$ is the ascent curve, which indicates how fast an anomaly comes; $b(x)$ is the attenuation curve, which indicates how fast an anomaly goes away. Here we make $a(x)$ an exponential function and $b(x)$ a negative exponential function, which are expressed as follows:

$$a(x) = h \times e^{-\frac{\ln \beta}{\varepsilon_1} \times (x - \varepsilon_1)}$$

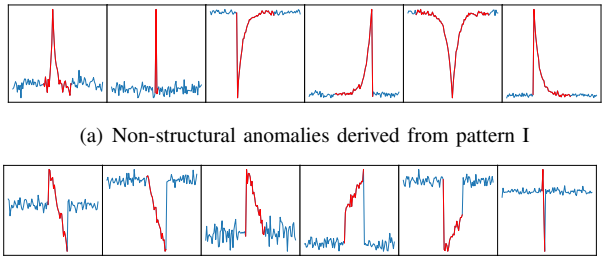
$$b(x) = h \times e^{\frac{\ln \beta}{\varepsilon_2} \times (x - \varepsilon_1)}.$$

  Since $\forall x \in \mathbb{R}, a(x), b(x) \neq 0$, we use $\beta = b(\varepsilon_2)/h$ to represent the error. Usually, $\beta = 1/10000$ is acceptable. $a(x)$ and $b(x)$ can also be linear or polynomial according to the need. The degree of this kind of anomaly is represented by h.
- As shown in Fig. 7 (b), anomaly pattern of type II is defined as:

$$\Gamma_2(\varepsilon, h_1, h_2, x) = \frac{h_2 - h_1}{\varepsilon - 1} \times x$$

  where $\varepsilon \in \mathbb{N}^+$, $\varepsilon \geq 2$, $h_1, h_2 \in [-\infty, +\infty]$, $|h_1| + |h_2| \neq 0$, $x = [0, 1, 2, \ldots, \varepsilon]$. This kind of anomalies are considered to occur suddenly. The degree of this kind of anomaly is represented by $max(|h1|, |h2|)$.

Users can generate a specific anomaly by giving parameters of a pattern, e.g., anomaly pattern of Type I with $h = 1, \varepsilon_1 = 1$ and $\varepsilon_2 = 1$ is essentially a point anomaly. They can also obtain a variety of anomalies by randomly varying the parameters of the patterns. Comparing Fig. 6 and Fig. 8, we can see that anomalies generated by our approach can cover most of the templates showed in Fig. 6.



(a) Non-structural anomalies derived from pattern I

(b) Non-structural anomalies derived from pattern II

Fig. 8: Non-structural anomalies generated by our approach.

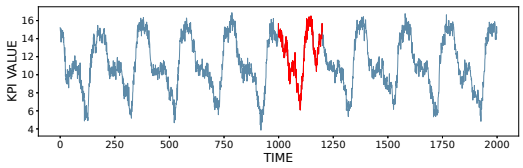
(2) Structural anomaly. This type of anomaly can occur in any component, such as deformation of a cycle in multi-seasonal component, and change of statistical moment of noise component. Therefore, it is very difficult to pre-define all patterns for this kind of anomalies in advance. But we can still use TSAGen to cover some representative structural anomalies. Here we mainly consider two typical structural anomalies, which are shape-based and statistical moment-based, respectively. Shape-based structural anomaly is characterized by the deformation of cycle shapes; statistical moment-based structural anomaly is characterized by the change of statistical moment. Thanks to RMDF and Pearson distribution applied in TSAGen, we can easily implement the above two types of anomalies. For shape-based anomaly, we specify a smaller forking depth for all normal cycles, and generate an anomaly cycle by specifying a much larger forking depth. For statistical moment-based anomalies, we randomly change the statistical moment of a segment in the noise component and label this segment as an anomaly.

Fig. 9 (a) shows an example of shape-based anomaly. We can see that the shape of the anomaly cycle (colored in red) is significantly different from the normal part (blue). Fig.9 (b) shows an example of statistical moment-based anomaly. It is easy to observe that the anomaly segment significantly differs from the normal part.

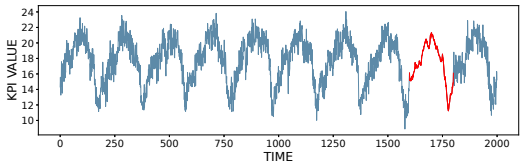
C. Degree of Anomaly

To provide a tuning knob to vary the severity of anomalies, another challenge we confront with is how to determine the degree of anomaly. This problem concerns the quality of ground truth. Anomalies of high degree can be detected easily, while anomalies of low degree (e.g., small fluctuations) may be hard to detect. Clearly, the degree of anomalies depends on specific context. For instance, in a fluctuating environment, an anomaly usually has a large fluctuation, while in a stationary environment a small fluctuation may also be an anomaly. Therefore, to control the "quality" of anomalies, it is necessary to establish the relationship between anomaly degree and context.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3098784, IEEE Transactions on Network and Service Management



(a) An example of shape-based anomaly



(b) An example of statistical moment-based anomaly

Fig. 9: Examples of structural anomaly.

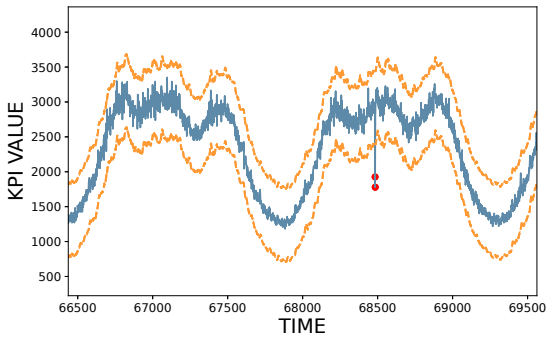


Fig. 10: Low probability boundaries. Regions above the upper dashed line or below the lower dashed lines are called low probability regions.

(1) Degree of non-structural anomaly. Anomalies are usually rare events. Empirically, the higher the degree of an anomaly, the rarer it is. Thus, with respect to non-structural anomaly, our problem can be expressed as follows. Given a context $\mathbf{x}$ (essentially a KPI segment), x_i is an instance inside $\mathbf{x}$, u_q, l_q denote the upper-bound and lower-bound, respectively, $u_{q,i}$, $l_{q,i}$ represent the corresponding bit in upper-bound and lower-bound for x_i, respectively. Can we find a u_q s.t. $\forall x_i,\ p(x_i > u_{q,i}|\mathbf{x}) < q$, as well as a l_q s.t. $\forall x_i,\ p(x_i < l_{q,i}|\mathbf{x}) < q$, where q is a desired small threshold?

Fortunately, this problem has been solved by Siffer et al. [41]. They successfully applied Extreme Value Theory (EVT) [42] and showed the power of EVT. As a widely-used tool, EVT can estimate the probability of rare events by fitting an Extreme Value Distribution (EVD) for the distribution tail of the context. That is to say, given a probability q, we can estimate a low probability boundary for the context, meaning that it can figure out to what degree a non-structural anomaly is rare enough to be a true anomaly. Here we applied the approach described in [41] to establish low probability boundaries and make non-structural anomalies fall into the low probability regions to ensure the quality of ground truth. As shown in Fig. 10, the upper dashed line is u_q, the lower dashed line is l_q. Regions above u_q or below l_q are called low probability regions.

Here we also call the probability q as the risk. It represents the risk that an anomaly is not a true anomaly, since as q decreases, injected anomalies are more and more likely to be true anomalies. By tuning the q value, we can control the quality of ground truth. In practice, we should carefully tune the q value. For instance, the anomalies can be detected easily if q is an extremely low value. In this case, most detection algorithms will perform well, and the generated data are not very useful for distinguishing the performance of different algorithms. According to our evaluation (refer to Section VI-F), when q = 1e-3, the performance of benchmark algorithms are most closest to their performance on real data.

(2) Degree of structural anomaly. For this type of anomaly, it is hard to explicitly define its degree due to its diverse forms. Instead, we implicitly define the degree of structure anomaly by adjusting parameters in different cases. For shape-based anomaly, we make sure that the forking depth of the anomaly cycle is much larger than the normal part (i.e., $d - 3 \leq \hat{d} \leq d - 1$, as presented in Section VI-E); For statistical moment-based anomaly, we make sure that the statistical moments of anomaly segments change significantly enough.

Point-adjust [1] is a widely-used strategy in the evaluation of detection algorithms. Its core content is as follows: if any point in an anomaly segment in the ground truth can be detected by a chosen threshold, we say this segment is detected correctly, and all points in this segment are treated as if they can be detected by this threshold. In this case, if a structural anomaly happens to contain an extreme high or low value, it will be easily detected by an algorithm that only models $p(x_t|x_{1:t-1})$. This does not help to distinguish different detection algorithms. To avoid this problem, we use EVT to ensure that the value of structural anomaly does not fall in the low probability region, resulting in structural anomalies that are easily detectable. In this way, we can control the quality of structural anomaly.

VI. Evaluation

In this section, we start by introducing the datasets, detection algorithms, and metrics used in our experiment. To estimate the effectiveness of TSAGen, we display KPI generated by TSAGen and KPIs from public datasets and show their visual similarity. Then we show the effectiveness of RMDF forking through experiments and also evaluate the computational overhead of TSAGen. In the last, we demonstrate the validity of generated data as well as the usefulness of TSAGen.

A. Datasets

There are several open access datasets for anomaly detection, such as Yahoo, NAB, AIOps challenges (for simplicity we call it AIOps in the rest of this section). Yahoo and NAB both contain synthetic data and real data while AIOps only contains real data collected from different IT companies including Tencent, eBay, etc. The details of these datasets can be found in the related papers [12]–[14].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3098784, IEEE Transactions on Network and Service Management

B. Benchmark Algorithms

We use 4 representative detection algorithms in our experiments: MA [43], AKDE [44], Donut [1], and DDCOL [8]. MA is a traditional statistical method. AKDE and DDCOL are unsupervised learning methods. Donut is unsupervised learning method and has the ability to leverage anecdotal labels. Detailed descriptions of the above algorithms are in related papers [1], [8], [43], [44]. In order to better measure the performance of a target algorithm (i.e., any detection algorithm under investigation), the above four algorithms are also integrated into TSAGen as benchmark algorithms. Users can always optionally get the performance of these algorithms on their customized data for comparison.

C. Metrics

Most of the evaluation methods for time series generation are to measure the similarity between generated data and real data based on a distance measure. This may be misleading in KPI anomaly detection domain, since our ultimate goal is to evaluate the detection algorithms rather than simulate the data. Therefore, we use the performance ranking of benchmark detection algorithms on the generated data to measure the effectiveness of generated data. If the *relative* performance rankings of benchmark algorithms on the generated KPIs and similar real KPIs are close, we can conclude that the generated data is valid, and thus it is reasonable to expect that if the performance of an algorithm is improved on the generated data, its performance will also be improved on similar real data.

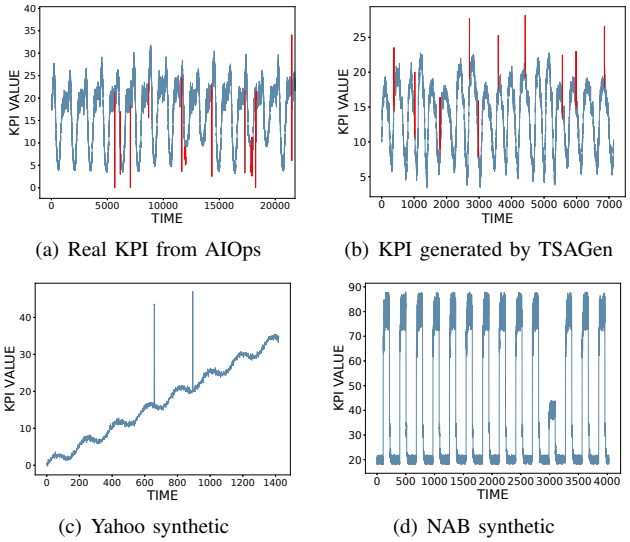
Next we introduce the performance metrics. F-score, precision and recall, whose definition can be found in [1], are widely used for evaluating the performance of detection algorithms. For the judgement of true positive (TP) number, false positive (FP) number and false negative (FN) number, we adopt the strategy used in AIOps challenges [14]. We will rank benchmark algorithms by their F-score.

D. Intuitive Illustration and Synthetic Benchmark

To visualize the effectiveness of generated data, we display a KPI generated by TSAGen. Synthetic KPI from Yahoo and NAB datasets as well as real KPI from AIOps are also displayed for visual comparison. As shown in Fig. 11, compared with the synthetic data from Yahoo and NAB, data generated by TSAGen contains drift, normal shape differences, and context-relevant anomalies and is more similar to the real KPI from AIOps. More examples can be found in the generated Benchmark introduced below.

To help users quickly grasp TSAGen, we use TSAGen to generate a synthetic benchmark, namely TSABen, which includes typical use cases for algorithm comparison and evaluation. The details are as follows.

The first group can be used for evaluating the algorithm's resistance to interference (e.g., noise level, drift degree). We generate this group in a variable control way. Taking the noise level case as an example, we generate 100 KPIs. In these KPIs, except for the noise level, all other parts of the data are exactly the same, including the location and degree (controlled by r)



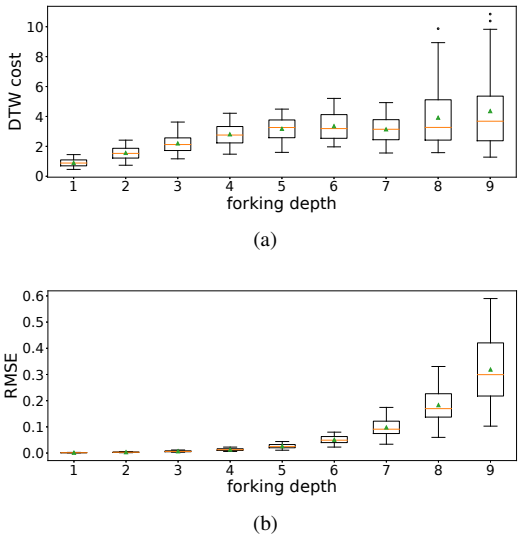
(a) Real KPI from AIOps (b) KPI generated by TSAGen

(c) Yahoo synthetic (d) NAB synthetic

Fig. 11: Real KPI and synthetic KPIs generated by TSAGen, Yahoo, and NAB. Anomalies are labelled in red color.

of anomalies, while the variance of noise increases over the order of KPIs.

The second group is used for evaluating the ability of the algorithm to detect different anomalies. In this group, we use TSAGen to inject only one anomaly to each KPI. According to our classification, there are two subgroups in this group, corresponding to, $p(x_t|x_{1:t-1})$, $p(x_{t-w:t}|x_{1:t-w-1})$, respectively, each subgroup contains several concrete types of anomalies. By running detection algorithms on this group, we can test their modeling capability (refer to Section V-A for the explanation of modeling capability of a detection algorithm).



(a)

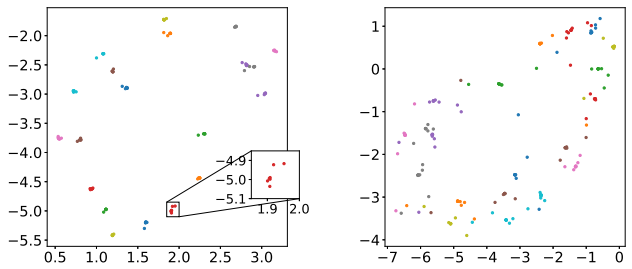
(b)

Fig. 12: With the increment of forking depth, the DTW cost (a) and RMSE (b) between contrast and forking curves get larger and larger.

E. Effectiveness of RMDF forking

To show the effectiveness of RMDF forking, we randomly generate 1000 contrast curves (of length 1000) with $d = 10$,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3098784, IEEE Transactions on Network and Service Management



(a) $\hat{d}$ range from 1 to d-4 (b) $\hat{d}$ range from 1 to d-1

Fig. 13: Two-dimensional t-SNE visualization with perplexity= 100 of the generated curves. The embeddings of cognate curves are clustered together, while non-cognate curves are distinguishable. After adding curves with larger forking depth, the embeddings are not distinguishable.

and for each contrast curve, we generate 9 forking curves for contrasting, with forking depth $\hat{d}$ range from 1 to $d-1$. Then we plot the Dynamic Time Warping (DTW) cost [45] and Root Mean Squared Error (RMSE) between each contrast curve and their corresponding forking curves, as shown in Fig. 12. Note that we remove the top 5% outliers because the RMDF process has randomness. It is possible that some generated curves might have a quite different shape. In practice, users can reject these "bad" curves, i.e., users can configure an accept interval, and only those forking curves whose DTW cost (or any other distance metric) fall in the interval will be accepted.

To further demonstrate the good characteristics of RMDF forking in modeling seasonal components, we randomly generate 20 curves (of length 1000, with $d = 10$) as well as their corresponding forking curves, with $\hat{d}$ range from 1 to $d-4$. The cognate curves (i.e., curves whose mutual shared depth $\hat{d} \geq 1$) are labeled as the same group, and non-cognate curves (i.e., curves whose mutual forking depth $\hat{d} = 0$) are labeled as different groups (corresponding to different colors in Fig. 13). Then we use t-SNE [46] to visualize the generated curves. As shown in Fig. 13(a), we can see that the curves of the same group are clustered together, although they have different shapes, and the curves of different groups are pushed away from each other. On this basis, we add the forking curves whose forking depth range from $d-3$ to $d-1$, and again, visualize these curves by t-SNE. As shown in Fig 13(b), all the curves can not be clustered well after adding the curves with larger forking depth. This is because the curves with a large $\hat{d}$ are so different that by t-SNE it is hard to tell which group they should belong to, indicating that these curves are different enough for the purpose of anomaly detection. The experimental result implies that RMDF forking can effectively model the small shape difference between cycles when $1 \leq \hat{d} \leq d-4$, and the generated curves can be deemed as anomalies when $d-3 \leq \hat{d} \leq d-1$.

F. Validity of Generated Data

In this experiment, we select seven similar seasonal KPIs from AIOps as a mini dataset, named $Real$. Then we calculate the proportion of all kinds of anomalies (shown in Table II). Also, we retrieve time series features $\theta_1 - \theta_9$ for every KPI in $Real$. Here we do not use meta features because the number of KPIs in $Real$ is relatively small. Since the amplitude drift of the season in $Real$ is small, and the frequency has almost no drift, we set $k_1 = 0.5, k_2 = 0.1$ and set the recursion depth $d = 10$. Then we generate a new dataset (named $Synthetic$) similar to $Real$ using these features. The proportions of each type of anomalies are listed in Table II.

In this experiment, benchmark algorithms will be executed on real and synthetic data respectively, we also adjust risk q during the experiment to show the the performance of benchmark algorithms under different quality of anomalies, so as to show the effect of q. Note that there is no randomness in the processes of MA and AKDE. Given the parameters, the result of each execution will be the same. However, there is randomness in the process of Donut and DDCOL, so we run them ten times on every KPI and get their average performance for ranking. In addition, the parameters of benchmark algorithms are optimized to achieve their best performance.

The experimental results are shown in Table III. The results show that the relative ranking of selected algorithms is roughly the same on the generated dataset and real dataset: Donut is always of the best performance on all experimental data and AKDE is always of the baddest performance, DDCOL slightly outperforms MA except in $Synthetic$ with $q = 1e\text{-}3$.

We also calculate the Pearson correlation coefficient between the relative rankings of selected algorithms on the four datasets we used, for F-score, Precision, and recall, respectively. As shown in Fig. 14(a), there is a strong positive correlation between relative rankings of F-score, and the same is true for relative rankings of Precision (Fig. 14(b)). Fig. 14(c) shows that there is also a strong positive correlation between relative rankings of Recall, except when $q = 1e\text{-}5$. This is because most of the anomalies in the experimental data are point anomalies (beat, spike, dip). When $q = 1e\text{-}5$, the degree of anomaly is relatively high, resulting in a high 2D-difference (i.e.,$(x_t - x_{t-1}, x_t - x_{t-2})$) in anomaly points. DDCOL is more sensitive to this change, so its recall increases faster with the increment of r. Note that the color bar of Fig. 14 (a) and (b) start from 0.9, and (c) from -1.0.

The experimental results indicate that when the generated data meet some conditions (i.e., the proportion of all kinds of anomalies is the same and their degree is similar), the generated data are effective and can be used for evaluating the performance of detectors. We can also infer that if the performance of an algorithm is improved on the generated dataset (satisfying the conditions), the performance of the algorithm will also be improved on real dataset.

G. Computational Overhead

TSAGen has only two time consuming processes in its workflow: the RMDF process and the EVT fitting process. Other parts of computation can be neglected compared to the above two processes (as we will see below). The computation time of EVT fitting depends on the length (i.e., θ_5/θ_4) of generated time series, while the computation time of the RMDF process mainly depends on the cycle number (i.e., θ_4). Note that $\theta_1 \sim \theta_3, \theta_6 \sim \theta_9, k_1$ do not impact the total

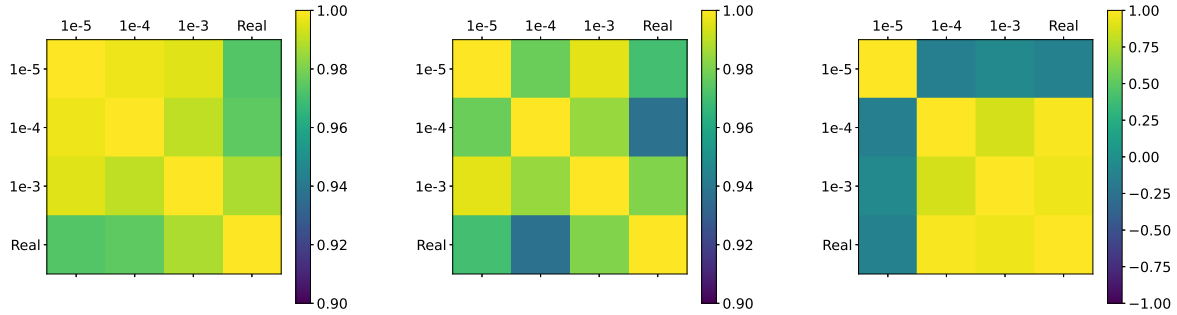
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3098784, IEEE Transactions on Network and Service Management

TABLE II: Proportion of all kinds of anomalies in the selected KPIs

ID	beat	spike	dip	fluctuate	level shift	total	Anomaly points	Total points	anomaly portion
1	11	2	38	7	0	58	654	100254	0.65%
2	13	0	49	10	0	71	1087	147629	0.74%
3	3	2	49	12	0	66	1113	147668	0.75%
4	9	1	51	3	0	64	426	147689	0.29%
5	22	2	50	1	0	75	381	137925	0.28%
6	6	1	39	4	1	51	527	129453	0.41%
7	13	2	45	4	0	64	584	147680	0.40%

TABLE III: Performance of benchmark algorithms on real data and synthetic data with different risk q.

Method	Synthetic with q=1e-5			Synthetic with q=1e-4			Synthetic with q=1e-3			Real		
	F_1-Score	Precision	Recall	F_1-Score	Precision	Recall	F_1-Score	Precision	Recall	F_1-Score	Precision	Recall
simple MA	0.7936	0.8239	0.7654	0.6386	0.5634	0.7370	0.5916	0.6812	0.5229	0.5765	0.9556	0.4127
Donut	0.9234	0.9314	0.9110	0.7915	0.8131	0.7692	0.6225	0.7847	0.5102	0.5944	0.8968	0.4442
DDCOL	0.8334	0.7538	0.9304	0.6624	0.6596	0.6602	0.5542	0.6954	0.4613	0.4520	0.9761	0.2918
AKDE	0.0128	0.0064	0.8827	0.0122	0.0061	0.8348	0.0096	0.0048	0.7577	0.0168	0.0085	0.6170



(a) F_1-*score* (b) Precision (c) Recall

Fig. 14: Pearson correlation between relative rankings on F-score, Precision, Recall.

computation time because these parameters do not affect the length and cycle number of generated time series. k_2 has a small impact on the generated KPI, on average, incerasing k_2 is essentially equivalent to increasing the expected cycle length (i.e., $1/\theta_4$) of generated KPIs.

Thus we control $\theta_1 \sim \theta_3, \theta_6 \sim \theta_9, k_1, k_2$ as constants, make $d = 10$, $\hat{d} = 2$, $\theta_4 = 1/200$ and vary θ_5 to generate KPIs of different length and cycle numbers. In this way, we can observe the influence of length and cycle number on the computation time. To ease illustration, we divide the overall computation time into three parts: time of the RMDF process, time for the EVT fitting process, and time for other processes (e.g., trend/noise generation, and anomaly injection). The hardware is a 1.8GHz, 16G RAM PC running Windows 10.

As shown in Fig. 16, with the increment of cycle number (θ_5) and length (θ_5/θ_4), the overall computation time increases linearly. Generating a long KPI of length $100,000$ only take about 55 seconds. This is acceptable because TSAGen works in an offline fashion. We can also find that EVT fitting and RMDF are the two most time-consuming processes, and time consumed by other processes can negligible.

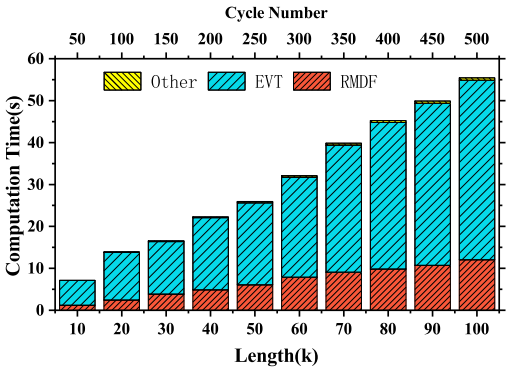


Fig. 15: Computation time of TSAGen. With the increment of cycle number (θ_5) and length (θ_5/θ_4), the overall computation time increases linearly.

It is worth noting that the computation time of benchmark evaluation depends on the execution time of benchmark algorithms. We only evaluate the computation time of the generation stage. The computation time of benchmark algorithms can be found in related papers [1], [8], [43], [44].

H. Results on TSABen

We also test the benchmark algorithms on TSABen. Here we selectively report some interesting results. In this experi-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3098784, IEEE Transactions on Network and Service Management

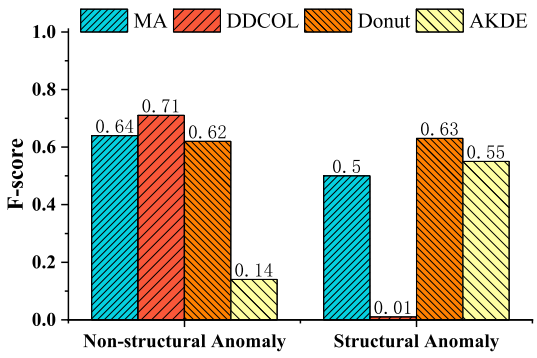


Fig. 16: Results on TSABen.

ment, we re-use the settings of benchmark detectors used in Section VI-F, and run them on TSABen's second group.

As shown in Fig. 16, MA and Donut have good performance on non-structural anomaly and structural anomaly. DDCOL only performs well on non-structural anomaly, but its performance on structural anomaly is extremely poor. AKDE contrasts DDCOL in that AKDE has bad performance on non-structural anomaly but good performance on structural anomaly. Note that because the proportions of anomalies in TSABen, $Real$, and $Synthetic$ are different, the detector's performance will be slightly different from what we show in Table. III. The above result implies that MA and Donut can both model $p(x_t|x_{1:t-1})$ and $p(x_{t-w:t}|x_{1:t-w-1})$; DDCOL can only model $p(x_t|x_{1:t-1})$, but failed to model $p(x_{t-w:t}|x_{1:t-w-1})$; AKDE can effectively model $p(x_{t-w:t}|x_{1:t-w-1})$, but has a limited power in modeling $p(x_t|x_{1:t-1})$. It is worth noting that long-term anomalies contribute the most to the F-score of AKDE on the non-structural anomaly, and AKDE is very weak in detecting point anomalies. This experiment is also cross-validated with the previous experiment. The reason that AKDE performs poorly in Table III is because most of the anomalies in the four datasets we used in Section VI-F are point anomalies.

Inspecting related papers [1], [8], [43], [44], we find that DDCOL is specifically designed for detecting abrupt changes and it does not consider the context (i.e., the previous behavior) at all. It only models $p(x_t)$, thus it only works well on point anomalies. MA can model $p(x_t|x_{t-w:t-1})$ by using a sliding window of size w, thus it can detect structural anomalies, but its performance is not that good. AKDE can only model $p(x_{t-w:t})$, and is not sensitive to point changes. Only Donut models $p(x_{t-w:t}|x_{1:t-w-1})$ in its design and thus it works well on both structural and non-structural anomalies. While there is no framework that unifies the design of these algorithm, the above analysis offers an intuitive explanation for the phenomenon shown in Fig. 16. The lesson we have learned from this experiment is that it is necessary to model the context for better detection results.

VII. Discussion

Usefulness and positioning: Synthetic data from TSAGen should not be treated as a substitute or replacement of real-world datasets. Instead, TSAGen generates complementary datasets for what-if tests in the following ways. First, since it is hard to control the characteristics of real data, developers and operators can conduct variable control evaluations with the help of TSAGen. This enables practitioners to easily discover the influences of a specific factor on the performance of detectors. Second, running the detectors on the generated data can yield a lot of meaningful results, just as we did in Section VI-H. With the help of TSAGen, operators and developers can evaluate the modeling capability of the detectors and discover if the detectors have fatal defects before deploying them in production.

Tractability: Although TSAGen has many parameters, TSAGen is easy to learn and use due to its modular design. By varying a parameter and fixing other parameters at a time, users can easily find the influence of a parameter on the generated data. TSAGen is fast, e.g., it takes less than 8 seconds (refer to Fig. 15) to generate a KPI of length $10,000$. Hence, users can quickly learn how to use TSAGen in an interactive way, i.e., tuning a parameter and getting feedback on the generated data immediately. Moreover, the synthetic benchmark TSABen contains many examples, each provided with a picture to illustrate its shape. Users can also learn the effect of parameters with TSABen.

Model selection: We used heuristics rather than some advanced co-training deep generative models (e.g. GANs and VAEs) [31], [32]. This decision is based on much deliberation over cons and pros. First, it is unlikely to conduct variable control generation with GANs, since it is extremely hard to vary a characteristic without distorting other characteristics of the KPI when using GANs. Second, GANs and VAEs are more suitable for simulating existing data, but not for conditional generation. Furthermore, these models can easily suffer from mode collapse [47]. The main goal of GANs is to reproduce data indistinguishable from the training data. This mismatches the purpose of what-if tests where we need the flexibility of controlling the statistical patterns in an interpretable manner.

VIII. Conclusion and Future Work

AIOps uses big data analytics and machine learning technologies to automatically identify and resolve IT operational issues. In its core, KPI anomaly detection plays a critical role. Nevertheless, due to security and privacy concerns, new KPI anomaly detection algorithms lack enough KPI datasets that consist of various anomalies for testing their performance. We fill this urgent need by developing TSAGen, a tool that can easily generate synthetic KPI data with various anomalies with simple control parameters. With TSAGen, researchers and practitioners can perform comprehensive evaluation of KPI anomaly detection algorithms as well as build what-if scenarios.

Since the data generated with TSAGen are not real, the data may not be suitable for directly training machine learning models, but they can be used for transfer learning. This will be investigated as our future work. Also, since single method is not possible to detect all types of anomalies, we plan to evaluate various detection algorithms with KPI data

generated with TSAGen and selectively integrate them for better detection results. In addition, due to the good properties of RMDF forking in generating shapes, the KPI data generated by RMDF forking may also be used to evaluate shape-based time series clustering. This remains to be further researched.

IX. Acknowledgment

This work is supported by the National Natural Science Foundation of China (62072465), the National Key Research and Development Program of China (2018YFB1800202, 2018YFB0204301, 2020YFC2003400) and the NUDT Research Grants (No. ZK19-38).

References

[1] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng, J. Chen, Z. Wang, and H. Qiao, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proc. of the 2018 International World Wide Web Conferences (WWW)*, 2018, p. 187–196.

[2] N. Zhao, J. Zhu, R. Liu, D. Liu, M. Zhang, and D. Pei, "Label-less: A semi-automatic labelling tool for kpi anomalies," in *Proc. of the 2019 IEEE Conference on Computer Communications (INFOCOM)*, 2019, pp. 1882–1890.

[3] W. Chen, H. Xu, Z. Li, D. Pei, J. Chen, H. Qiao, Y. Feng, and Z. Wang, "Unsupervised anomaly detection for intricate kpis via adversarial training of vae," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 1891–1899.

[4] Y. Su, Y. Zhao, W. Xia, R. Liu, J. Bu, J. Zhu, Y. Cao, H. Li, C. Niu, Y. Zhang, Z. Wang, and D. Pei, "Coflux: Robustly correlating kpis by fluctuations for service troubleshooting," in *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*, 2019, pp. 1–10.

[5] J. Bu, Y. Liu, S. Zhang, W. Meng, Q. Liu, X. Zhu, and D. Pei, "Rapid deployment of anomaly detection models for large number of emerging kpi streams," in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, 2018, pp. 1–8.

[6] Y. Wang, Z. Wang, Z. Xie, N. Zhao, J. Chen, W. Zhang, K. Sui, and D. Pei, "Practical and white-box anomaly detection through unsupervised and active learning," in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, 2020, pp. 1–9.

[7] N. Zhao, J. Zhu, Y. Wang, M. Ma, W. Zhang, D. Liu, M. Zhang, and D. Pei, "Automatic and generic periodicity adaptation for kpi anomaly detection," *IEEE Transactions on Network & Service Management*, vol. PP, no. 3, pp. 1–1, 2019.

[8] G. Yu, Z. Cai, S. Wang, H. Chen, F. Liu, and A. Liu, "Unsupervised online anomaly detection with parameter adaptation for kpi abrupt changes," *IEEE Transactions on Network and Service Management*, 2019.

[9] A. Castro, V. A. Villagra, B. Fuentes, and B. Costales, "A flexible architecture for service management in the cloud," *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, pp. 116–125, 2014.

[10] S. Aoki, K. Shiomoto, and C. L. Eng, "Few-shot learning and self-training for enodeb log analysis for service-level assurance in lte networks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2077–2089, 2020.

[11] D. Liu, Y. Zhao, H. Xu, Y. Sun, D. Pei, J. Luo, X. Jing, and M. Feng, "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *Proc. of the 2015 Internet Measurement Conference (IMC)*. Association for Computing Machinery, p. 211–224.

[12] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proc. of the 2015 ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2015, p. 1939–1947.

[13] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms – the numenta anomaly benchmark," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 38–44.

[14] "Aiops challenge," http://iops.ai/dataset_list/.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[16] "Summary of the amazon simpledb service disruption," https://aws.amazon.com/cn/message/65649/.

[17] "Summary of the amazon s3 service disruption in the northern virginia (us-east-1) region," https://aws.amazon.com/cn/message/41926/.

[18] "Summary of the amazon dynamodb service disruption and related impacts in the us-east region," https://aws.amazon.com/cn/message/5467D2/.

[19] C. Chatfield and H. Xing, *The analysis of time series: an introduction with R*. CRC press, 2019.

[20] L. Kegel, M. Hahmann, and W. Lehner, "Feature-based comparison and generation of time series," in *Proceedings of the 2018 International Conference on Scientific and Statistical Database Management (SSDBM)*.

[21] L. Kegel, M. Hahmann, and W. Lehner, "Generating what-if scenarios for time series data," in *Proc. of the 2017 International Conference on Scientific and Statistical Database Management (SSDBM)*.

[22] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.

[23] Z. Lin, A. Jain, C. Wang, G. C. Fanti, and V. Sekar, "Generating high-fidelity, synthetic time series datasets with doppelganger," *CoRR*, vol. abs/1909.13403, 2019.

[24] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional gans," 2017.

[25] Y. Kang, R. J. Hyndman, and F. Li, "Gratis: Generating time series with diverse and controllable characteristics," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 13, no. 4, p. 354–376, May 2020.

[26] N. Iftikhar, X. Liu, S. Danalachi, and F. E. Nordbjerg, "A scalable smart meter data generator using spark," *Springer, Cham*, 2017.

[27] A. H. Yaacob, I. K. T. Tan, S. F. Chien, and H. K. Tan, "Arima based network anomaly detection," in *2010 Second International Conference on Communication Software and Networks*, 2010, pp. 205–209.

[28] Q. Luo, S. J. Hu, and Z. L. Liu, "Research on forecasting the lifetime value of major goods owners in railway freigjt traffic," *Journal of The China Railway Society*, 2005.

[29] G. Forestier, F. Petitjean, H. A. Dau, G. I. Webb, and E. Keogh, "Generating synthetic time series to augment sparse datasets," in *IEEE International Conference on Data Mining*, 2017.

[30] R. B. Cleveland *et al.*, "Stl: A seasonal-trend decomposition procedure based on loess," *DOI: citeulike-article-id*, vol. 1435502, 1990.

[31] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[32] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2014.

[33] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014. [Online]. Available: http://arxiv.org/abs/1411.1784

[34] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang, "Time-series anomaly detection service at microsoft," in *Proc. of the 2019 ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2019, p. 3009–3017.

[35] N. Laptev, "Anogen : Deep anomaly generator," https://research.fb.com/wp-content/uploads/2018/11/AnoGenDeep-AnomalyGenerator.pdf, accessed June, 2020.

[36] R. Schlittgen, "Robert h. shumway and david s. stoffer: Time series analysis and its applications with r examples, 2ndedn." *Asta Advances in Statistical Analysis*, vol. 92, no. 2, pp. 233–234, 2008.

[37] K. Pearson, "Contributions to the mathematical theory of evolution," *Philosophical Transactions of the Royal Society of London. A*, vol. 185, pp. 71–110, 1894.

[38] K. Pearson, "Contributions to the mathematical theory of evolution.—ii. skew variation in homogeneous material," *Philosophical Transactions of the Royal Society of London. A.*, no. 186, pp. 343–414, 1895.

[39] L. I. Kuncheva, "Classifier ensembles for detecting concept change in streaming data: Overview and perspectives," in *2nd Workshop SUEMA*, vol. 2008, 2008, pp. 5–10.

[40] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[41] A. Siffer, P.-A. Fouque, A. Termier, and C. Largouet, "Anomaly detection in streams with extreme value theory," in *Proc. of the 2017 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 1067–1075.

[42] J. Beirlant, Y. Goegebeur, J. Segers, and J. L. Teugels, *Statistics of extremes: theory and applications*. John Wiley & Sons, 2006.

[43] D. R. Choffnes, F. E. Bustamante, and Z. Ge, "Crowdsourcing service-level network event monitoring," in *Proc. of the 2010 ACM SIGCOMM conference*, 2010, pp. 387–398.
[44] O. Ibidunmoye, A. Rezaie, and E. Elmroth, "Adaptive anomaly detection in performance metric streams," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 217–231, 2018.
[45] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, ser. AAAIWS'94. AAAI Press, 1994, p. 359–370.
[46] V. Laurens, M. Der, and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 2605, pp. 2579–2605, 2008.
[47] Z. Lin, A. Khetan, G. Fanti, and S. Oh, "Pacgan: The power of two samples in generative adversarial networks," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018, pp. 1498–1507.

Chengyu Wang received the bachelor's degree in computer science and technology from Wuhan University of Technology, China, in 2018. He is currently a master student in the College of Computer, National University of Defense Technology, China. His main research interests include network security, network measurement, AIOps, and time series analysis.

Kui Wu received the BSc and the MSc degrees in Computer Science from the Wuhan University, China, in 1990 and 1993, respectively, and the PhD degree in Computing Science from the University of Alberta, Canada, in 2002. He joined the Department of Computer Science, University of Victoria, Canada, in 2002, where he is currently a Full Professor. His research interests include network science, edge computing, AIOps, and network security.

Tongqing Zhou received the bachelor's, master's, and Ph.D degrees in computer science and technology from the National University of Defense Technology (NUDT), China, in 2012, 2014, and 2018, respectively. He is currently a postdoc in College of Computer, NUDT. His main research interests include ubiquitous computing, crowdsensing, and data privacy.

Guang Yu received the bachelor's degree in computer science and technology from Sichuan University, China, in 2018. He is currently working toward the Ph.D. degree at the College of Computer, National University of Defense Technology, China. His main research interests include anomaly/outlier detection and self-supervised/unsupervised learning. His works have been published on leading conference and journals, such as ACM MM and TNSM

Zhiping Cai received the B.Eng., M.A.Sc., and Ph.D. degrees in computer science and technology from the National University of Defense Technology (NUDT), China, in 1996, 2002, and 2005, respectively. He is a full professor in the College of Computer, NUDT. His current research interests include network measurement, network security and big data. He is a senior member of the CCF and a member of the IEEE.