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.2019.2919327, IEEE Transactions on Network and Service Management

Automatic and Generic Periodicity Adaptation for KPI Anomaly Detection

Nengwen Zhao[†¶], Jing Zhu*[†¶], Yao Wang[†¶], Minghua Ma[†¶], Wenchi Zhang[‡], Dapeng Liu[‡], Ming Zhang[§], Dan Pei[†¶]
[†]Tsinghua University [‡]BizSeer [§]China Construction Bank
[¶]Beijing National Research Center for Information Science and Technology (BNRist)

***Abstract*—KPI (Key Performance Indicator) anomaly detection is critical to ensure service quality and reliability. Due to the effects of work days, off days, festivals and business activities on user behavior, KPIs may exhibit different patterns within different days, which we call periodicity profiles of KPIs. However, existing KPI anomaly detection approaches have difficulties in adapting to diverse periodicity profiles due to the lack of generality. In this paper, we propose an automatic and generic framework called *Period*, which can accurately detect the periodicity profiles through daily subsequences clustering, and improve the performance of anomaly detection methods by robustly and automatically adapting to different periodicity profiles. In our evaluation using several real-world KPIs with different periodicity profiles from large Internet-based services, the clustering algorithm used to detect periodicity can achieve about 0.95 accuracy on average. More importantly, further evaluation on 56 KPIs shows that *Period* can significantly improve the best F-score of several widely-used anomaly detection approaches by up to 0.66.**

***Index Terms*—Key Performance Indicator, Anomaly Detection, Periodicity Detection, Subsequences Clustering**

I. Introduction

NOWADAYS, there is a growing need to accurately detect anomalies and trigger timely troubleshooting or mitigation in Internet-based services, such as search engines, online shopping and social networks [1], [2]. It is critical to closely monitor a great number of KPIs (**Key Performance Indicators**), such as the number of page views (PV), the number of online users, and CPU utilization, to ensure service quality and reliability [3], [4]. In general, many KPIs are strongly correlated with user behavior, which can be influenced by work days, off days, festivals, promotion and some business activities. For example, based on our observation on an online shopping service, the PV is higher in the day time than that at night in a common diurnal cycle, and it is higher on work days than off days. If there is a promotion on every Friday, the PV will also increase significantly on those days. Therefore, KPIs may exhibit different patterns within different days, which we call **periodicity profiles**, such as daily, weekly and other imperfect or complex periodicity. Fig. 1 shows an intuitive example of imperfect periodicity, where the work days, off days (including New Year's Day and weekends) and Spring Festival exhibit different patterns. Anomalies are labeled in red [1].

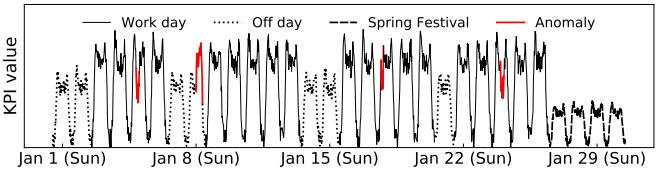
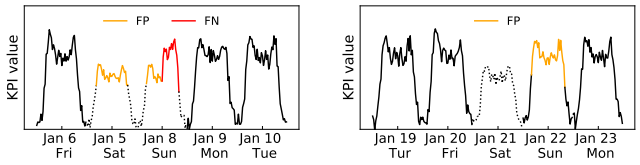


Fig. 1: KPI with imperfect periodicity due to the off days (New Year's Day and weekends) and Spring Festival [1].



(a) TSD-day (b) TSD-week

Fig. 2: With imperfect periodicity, both TSD-day and TSD-week will report some false positives (FP, marked in orange) and false negatives (FN, marked in red).

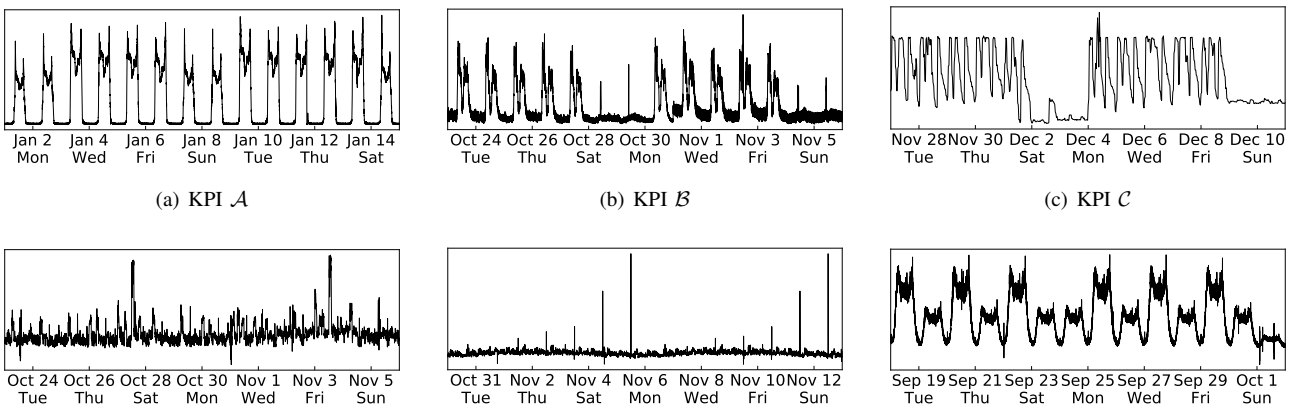
Although a large number of anomaly detection approaches have been proposed in the literature [1]–[11], including traditional anomaly detectors (e.g., Moving Average (MA) [6], Exponential Weighted MA (EWMA) [5], Holt-Winters [1] and Time Series Decomposition (TSD) [11]), supervised ensemble algorithms (e.g., Opprentice [3] and EGADS [7]) and unsupervised algorithms (e.g., Donut [4]), existing methods have no ability to adapt to diverse periodicity profiles due to the lack of **generality**.

In detail, anomaly detectors like Holt-Winters and TSD need the season length of KPI as their input parameters, which is usually manually configured as *either* day *or* week by default. Take the KPI in Fig. 1 as an example, it is unreasonable to simply use a fixed season length as the parameter of TSD or Holt-Winters. As Fig. 2 shows, both TSD-day and TSD-week will generate some false positives (marked in orange) or false negatives (marked in red), since the fixed season length cannot characterize the imperfect periodicity. As for some other anomaly detectors like MA and EWMA, the imperfect periodicity profiles can destroy the stationarity or perfect seasonality of KPI, which will

* Jing Zhu is the corresponding author.

[1]According to the 2017 public holiday calendar in China, Monday January 2nd is an off day, and Sunday January 22th is a work day.

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.2019.2919327, IEEE Transactions on Network and Service Management



(a) KPI $\mathcal{A}$ (b) KPI $\mathcal{B}$ (c) KPI $\mathcal{C}$

(d) KPI $\mathcal{D}$ (e) KPI $\mathcal{E}$ (f) KPI $\mathcal{F}$

Fig. 3: Periodicity profiles of the six KPIs used in our experiments.

result in unsatisfactory performance. Besides, the state-of-the-art unsupervised anomaly detection algorithm Donut based on Variational Auto-Encoder (VAE) [4] performs well only on smooth seasonal KPIs. It is also hard for Donut to capture the periodicity profile from training data, since there is no time or date information in Donut input variables (a window of data points). Consequently, the above methods have difficulties in handling various periodicity profiles in real applications.

In this work, we aim to design an automatic and generic framework to *robustly adapt KPI anomaly detection to diverse periodicity profiles*. There are four key challenges as follows.

- **Diverse periodicity profiles.** KPIs often have very different periodicity profiles and Fig. 1 only presents a common and specific phenomenon that off days and festivals are special. Fig. 3 shows six KPIs that are used later in our evaluation (§IV). We can observe that in addition to the off days and festivals, other days may also behave differently, e.g., Friday in KPI $\mathcal{D}$. It is infeasible to predefine the special days based on the calendar since we are not sure which day is special. Therefore, a generic framework to deal with diverse periodicity profiles is needed.
- **Large number of KPIs.** Since the number of KPIs in practice is collected about millions to tens of millions in large Internet-based services [12], it is labor intensive for operators to check the periodicity profile and configure parameters for each KPI manually. Therefore, an automatic periodicity adaptation framework for anomaly detection should be provided.
- **Periodicity drift.** Periodic patterns might not be perfect and can have some "drifts", i.e., the actual execution time of a daily job might vary by tens of minutes, which must be explicitly dealt with.
- **Various anomaly detection approaches.** Given that different anomaly detection methods perform differently for different KPIs [3], our design needs to be generic and robust to accommodate to various anomaly detection approaches.

To address the above challenges, in this paper, we propose an automatic and generic periodicity adaptation framework for KPI anomaly detection named *Period*. The core idea of *Period* is to transform the anomaly detection on a given KPI with unknown periodicity profile into anomaly detection on k sub-KPIs with "clear daily periodicity", and the k sub-KPIs can be acquired by our proposed daily subsequences clustering technique.

Period has two key components. (1) **Offline periodicity detection**. For each KPI, *Period* first cuts the historical KPI data into *daily subsequences*. Then *Period* transforms periodicity detection into an unsupervised *clustering problem* of these daily subsequences. Each of the resulting k clusters indicates a different daily pattern (e.g., $k = 3$ in Fig. 1). By querying the APIs for *official public holiday calendar* (e.g., [13], [14]) and using the dates of daily subsequences, each cluster can be assigned a specific name (e.g., day of week, day of month, off day, festival). Then all the daily subsequences in a cluster are concatenated in chronological order into a new *sub-KPI*, each of which has clear daily periodicity, and *Period* applies an anomaly detection model for each sub-KPI (e.g., each cluster uses a separate TSD model). (2) **Online anomaly detection adaptation**. For online incoming KPI data, the current date is assigned to the corresponding cluster according to the calendar and clusters' name. Afterwards, the cluster-specific model is adopted to detect anomalies. In this way, for a given KPI, similar daily patterns will share a model and different patterns will not interfere with each other.

The contributions of the paper are summarized as follows.

- To the best of our knowledge, there are no related works about complex periodicity adaptation for KPI anomaly detection in the literature. This paper is the first one to identify the problem of automatically adapting to various periodicity profiles for anomaly detection methods. Besides, this paper proposes an automatic and generic framework named *Period* to enable anomaly detection methods to deal with KPIs of diverse periodicity profiles.
- We novelly propose to solve periodicity detection through daily subsequences clustering, and successfully handle the *constrained periodicity drift* problem. In detail, we define an improved time series distance measure named *constrained shape-based distance* (cSBD) and adopt DBSCAN with density estimation to detect the periodicity profile with high accuracy (over 0.95 on average).

- To demonstrate the effectiveness of *Period*, we have conducted extensive evaluation experiments using 56 labeled KPIs with diverse periodicity profiles collected from a large commercial bank. The results show that *Period* can significantly improve the best F-score of several anomaly detection methods by up to 0.66 through automatically adapting them to various periodicity profiles.

The rest of the paper is organized as follows. §II introduces some background and related work. §III shows the design details about *Period*. Evaluation of periodicity detection and anomaly detection adaptation are presented in §IV and §V, respectively. §VI discusses some limitations of our algorithm and future work. Finally, we conclude this paper in §VII.

II. Background and Related Work

In this section, we mainly introduce some key concepts used in this paper and some related work about KPI anomaly detection and traditional periodicity detection.

A. Key Concepts

KPI. KPIs are collected continuously to describe the current state of servers (low-level) or applications (high-level), and ensure service quality and reliability. KPI is a kind of time series data with the format of (timestamp, value) and can be denoted as $x = \{x_1, x_2, \cdots, x_N\}$, where x_i is the value corresponding to time index i for $i \in \{1, 2, \cdots, N\}$, and N is the number of data points of this KPI. KPI values can be collected and calculated from Simple Network Management Protocol (SNMP), syslogs, network traces, web access logs and other data sources.

KPI Anomaly. Anomalies refer to a subset of data points in a KPI that do not conform to the expected behavior and significantly differ from the normal data [2], [3], e.g., jitters, slow ramp-ups, spikes and dips (labeled in red in Fig. 1). Anomalies in KPIs often indicate potential failures on relevant applications, such as server failures, server overload, network failure, external attacks, etc.

Time Series Periodicity. Time series may contain multiple seasonal cycles of different lengths [15], [16]. For example, the KPI $\mathcal{B}$ shown in Fig. 3(b) exhibits both daily and weekly cycles, which we called *basic short-term seasonality* and *whole long-term seasonality*. In this paper, we assume that the basic seasonality of KPI is one day (also discussed in §VI) and focus on detecting the whole long-term seasonality. For the convenience of later discussions, except where noted, season length, period and seasonality refer specifically to the whole long-term seasonality.

B. KPI Anomaly Detection

KPI anomaly detection can be formulated as follows: for a KPI x, given historical observations $x_{t-T+1}, \cdots, x_t$, determine whether an anomaly occurs at time t (denoted by $y_t = 1$). An anomaly detection method typically computes a real-valued score (called anomaly score) indicating the certainty of having $y_t = 1$, i.e., $p(y_t = 1|x_{t-T+1}, \cdots, x_t)$, instead of directly computing y_t. Human operators can affect whether to declare an anomaly by choosing a threshold, where a data point with a score exceeding this threshold indicates an anomaly.

As discussed in §I, a rich body of literature exists on KPI anomaly detection [1]–[11], including traditional anomaly detectors, ensemble supervised learning algorithms and unsupervised algorithms. The basic schema of traditional anomaly detector works in the following way. When an anomaly detector receives an incoming KPI value, it internally produces a forecast value. The absolute difference between the incoming value and the forecast one is regarded as the anomaly score. Ensemble supervised learning, such as Opprentice [3] and EGADS [7], adopt the forecast values from various traditional anomaly detectors as features to train a binary classifier (like Random Forest in [3]), then for online KPI data, the classifier will give a probability for each point indicating the certainty of anomaly, i.e., $p(y_t = 1|x_{t-T+1}, \cdots, x_t)$. Unsupervised algorithms directly model the raw KPI data and try to learn the normal pattern of the KPI. The anomaly score can be computed based on how well the current window of a given data point follows the normal expected patterns. However, all the above cannot deal with the KPIs with diverse periodicity profiles like Fig. 1 and Fig. 3 show, and can lead to many false negatives and false positives (see Fig. 2) in real applications.

In addition to anomaly detection algorithms in academia, there are also many anomaly detection products in industry, including Anodot [17], Splunk [18], Datadog [19], Dynatrace [20], Prometheus [21], Grok [22], Kibana [23], etc. To the best of our knowledge, only Anodot proposes to adopt Auto-correlation Function (ACF) to handle time series data without prior knowledge of seasonality [24], and the weakness of ACF will be discussed in §II-C2. Other products ignore the influence of complex periodicity on anomaly detection.

Therefore, it is in urgent need to design a generic and automatic framework so as to adapt anomaly detection methods to various periodicity profiles.

C. Traditional Periodicity Detection

To the best of our knowledge, this paper is the first one to focus on the periodicity of KPIs for *anomaly detection*. However, considering that KPI is a kind of time series, in the field of traditional time series data mining, many efforts have been devoted to the research of periodicity detection with very different goals [25]–[28]. Generally, traditional periodicity detection adopts the techniques in time series analysis to model the raw data and *output KPI's season length* (i.e., an explicit number, such as the length of day or week). Two widely-used periodicity detection methods, *Periodogram* and *Auto-correlation Function*, are introduced below [27].

1) Periodogram: Given a KPI x with length N, suppose that X is the Discrete Fourier Transform (DFT) of x, then the periodogram $\mathcal{P}$ is provided by the squared length of each Fourier coefficient:

$$\mathcal{P}(f_{k/N}) = ||X(f_{k/N})||^2, \quad k = 0, 1, \cdots, \lceil \tfrac{N-1}{2} \rceil \tag{1}$$

where

$$X(f_{k/N}) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi kn}{N}} \tag{2}$$

We pick the frequency with the largest power of the periodogram to discover the season length. For example, Fig. 4(a) shows a two-week-long KPI with a half-hour monitoring interval. Its season length is one day, i.e., $T = 24 \times 2 = 48$ points. Fig. 4(b) shows the periodogram of this KPI, and the red point denotes the largest power with frequency equals to 0.0208, thus the output season length is $1/0.0208 \approx 48$.

2) Auto-correlation Function (ACF): Given a KPI x, for different l lags, we have:

$$ACF(l) = \frac{\sum_{i=0}^{N-1} x(i)x(i+l)}{N}, l = 1, 2, \cdots, \frac{N-1}{2} \quad (3)$$

It is clear that the auto-correlation becomes high at certain lags, i.e., $T, 2T, 3T, \cdots$. Typically, as Fig. 4(c) shows, the lag corresponding to the first peak in the ACF is regarded as the KPI's period (labeled in the red point).

However, the above two approaches perform well only on time series with a short period (e.g., one day) and become a little inaccurate for a longer period. Specifically, each element of the periodogram provides the power at frequency k/N (at period N/k). It is obvious that DFT bins are fine-grained for the short period and coarse-grained for the long period, for example, for a time series of length $N = 600$, the DFT bins will be $(N/1, N/2, N/3, \cdots = 600, 300, 200, \cdots)$. Therefore, the accuracy of detected period deteriorates for longer period due to the increasing width of DFT bins (N/k). Another problem is spectral leakage, which causes frequencies that are not multiples of the DFT bin width, to disperse over the entire spectrum [27]. Besides, auto-correlation is also affected by longer lags, since the last l points of time series are padded with zero when computing auto-correlation, which will lead to lower correlation. More importantly, both approaches are significantly impacted by the imperfect or ambiguous periodicity and noises or anomalies.

Fig. 5 provides a case where the three-week-long KPI has an one-week long-term seasonality with 5-minute monitoring interval ($T = 7 \times 24 \times 12 = 2016$ points). However, both periodogram and auto-correlation only detect the basic short-term seasonality (one day) and fail to detect the whole long-term seasonality correctly (one week). This is because the patterns within weekdays/weekends are very similar and they cannot distinguish the difference between weekdays and weekends accurately.

3) Summary: The above two methods can only provide an explicit (and often inaccurate) season length, while **in practice the complex or imperfect periodicity profile cannot be characterized by a fixed number**, e.g., the KPI in Fig. 1 has complex periodicity rather than a day or a week. In summary, traditional methods are not able to handle diverse periodicity profiles for KPI anomaly detection.

III. Design of *Period*

This section first presents the core idea and overview of *Period*, then introduces *Period*'s two key components *Periodicity Detection* and *Anomaly Detection Adaptation* in detail.

A. Core Idea and Overview

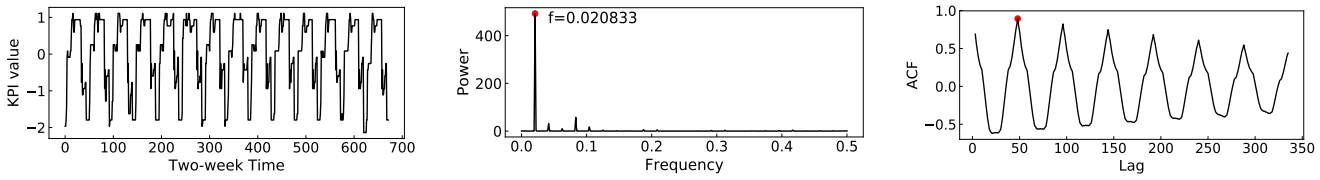
The core idea of *Period* is *transforming the anomaly detection on a given KPI with unknown periodicity profile into anomaly detection on k sub-KPIs with "clear daily periodicity"*, and the k sub-KPIs can be acquired by our proposed daily subsequences clustering technique. Fig. 6 presents an intuitive example to illustrate the core idea. Daily subsequences clustering will discover the number of different daily patterns in the given KPI ($k = 3$ in this example). We can assign *a specific name* to represent each cluster (e.g., work day, off day and Spring Festival in this example). Then the daily subsequences in one cluster are concatenated in chronological order into a new sub-KPI with clear daily periodicity as shown in Fig. 6(b). Next, we train a separate anomaly detection (AD) model (e.g., Holt-Winters (HW)) for each cluster. Afterwards, for the current date, *Period* first queries the official holiday calendar to see whether it is a work day, off day, or Spring Festival, and assigns the data into the correct cluster. For example, if the current day is January 31th 2017, during the Spring Festival, AD Model 3 will be adopted for anomaly detection.

The overall framework of *Period* shown in Fig. 7 has two major components. First, historical KPI data is used for offline periodicity detection. In detail, the historical KPI is cut into daily subsequences, then the model adopts constrained shape-based distance (cSBD) as distance measure to deal with constrained periodicity drift and DBSCAN with density estimation to cluster these subsequences. For each cluster, we can assign a name for it based on the official holiday calendar as shown in Algorithm 1. Then we concatenate all daily subsequences in a cluster into a sub-KPI and train an anomaly detection model independently for each sub-KPI. The second step is online anomaly detection adaptation, which aims to choose the correct AD model to detect anomalies by assigning into the corresponding cluster according to the clusters' name and current date. In this way, similar daily patterns in a KPI will share a model and different patterns will not interfere with each other, so that *Period* can make full use of historical data and latest data to detect anomalies more accurately. The historical data for offline periodicity detection are updated periodically, e.g., monthly, so that the newly appeared daily pattern can be periodically incorporated into the newly learned offline model.

B. Offline Periodicity Detection

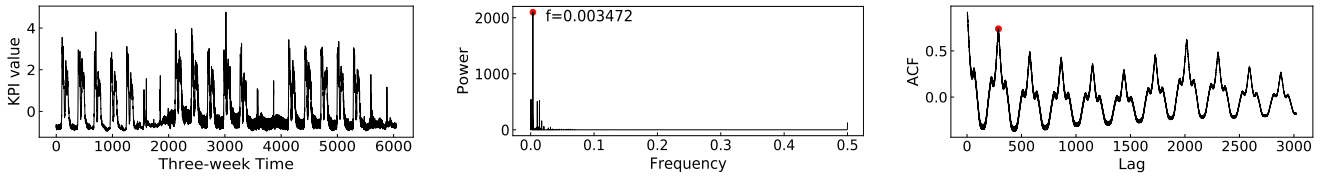
1) Data Preprocessing: Before daily subsequences clustering, we first need to preprocess the KPI data. KPIs are monitored with certain interval (e.g., every minute), but occasionally, a monitoring system does not receive data, leading to missing values. Fortunately, the percentage of missing values usually is very small according to our observation. We simply use linear interpolation to fill them based on their adjacent data points. Besides, considering that noises and anomalies in historical data may mislead the clustering result, we apply Moving Average [6] to smooth the KPI, so as to remove noises and weaken the negative impact of anomalies. Notice that data preprocessing is conducted on offline historical data, not on

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.2019.2919327, IEEE Transactions on Network and Service Management



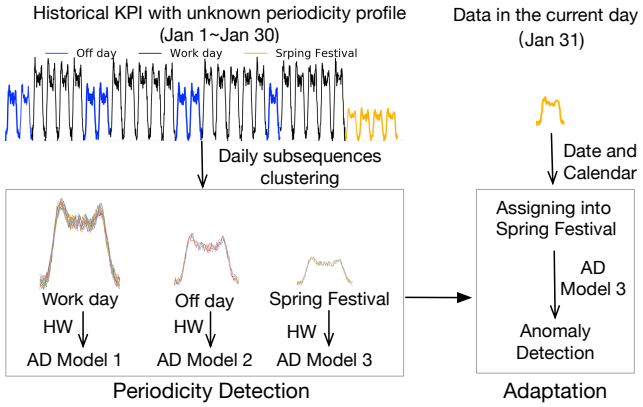
(a) KPI Data (T=one day, 48 points) (b) Periodogram (one day, 48 points) (c) Auto-correlation (one day, 48 points)

Fig. 4: Illustration of periodogram and auto-correlation.

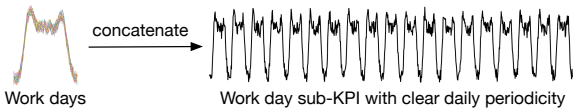


(a) KPI Data (T=one week, 2016 points) (b) Periodogram (one day, 288 points) (c) Auto-correlation (one day, 288 points)

Fig. 5: Failure case of periodogram and auto-correlation.



(a) Core idea



(b) Concatenating daily subsequences into sub-KPI

Fig. 6: A toy example to illustrate *Period*'s core idea.

online data on which we need to detect anomalies. Here we choose the window size of MA as half an hour based on our experience and experiments. We found that if the window size is too small, noise and anomalies cannot be eliminated completely. And a too large window will destroy the original shape and information of KPI. After that, the KPI is cut into daily subsequences (i.e., from 0:00 to 24:00), which will be used in the following steps.

2) Distance Measure: The first step of daily subsequences clustering is selecting an appropriate distance measure. Before reviewing the existing time series similarity measures, we first introduce a common phenomenon in practice, i.e., constrained periodicity drift. In detail, for some real applications, the task is triggered manually, instead of triggered by the script automatically, which leads to time shift among daily subsequences. According to our domain knowledge, manual triggering time is generally within a tolerable range of time, e.g., one daily task is triggered two hours after the morning office hour starts every work day. Thus a threshold needs to be added to restrict the range of time shift. Fig. 8 depicts an intuitive example. It is obvious that the three peaks all happen at different times in three days, but they are all between 10:00 and 14:00. Consequently, the distance measure in our algorithm needs to tackle the constrained periodicity drift problem.

Many time series distance measures have been proposed in the literature. L_p-norms [29] are a group of widely-used distance measures thanks to their simplicity and efficiency, but they are sensitive to noises and distortions along the time dimension. Dynamic Time Warping (DTW) [30] is well-known for its robustness against time shift and scaling distortion, but has high computational complexity. Generally, it takes $O(m^2)$ to compute the distance between two sequences with length m, which makes it impractical to deal with a large number of long sequences (m is the length of daily subsequence in our case). Cross-correlation [31], e.g., shape-based distance (SBD) [32] can natively handle time shift and its computational complexity can be reduced to $O(m \log(m))$ through Convolution Theorem and Fast Fourier Transform. Thus cross-correlation-based metric can be a suitable distance measure in our algorithm. In order to cope with constrained periodicity drift problem, we define an improved distance measure named constrained SBD (cSBD) based on SBD.

Given two sequences $\vec{x} = (x_1, x_2, \cdots, x_m)$ and $\vec{y} = (y_1, y_2, \cdots, y_m)$, in order to achieve shift-robustness, cross-correlation keeps $\vec{y}$ static and slides $\vec{x}$ over $\vec{y}$ to compute the inner-product for each shift s. The sequence $\vec{x}$ with a shift s is denoted as:

$$\vec{x}_{(s)} = \begin{cases} (\overbrace{0, \cdots, 0}^{|s|}, x_1, x_2, \cdots, x_{m-s}), & s \geqslant 0 \\ (x_{1-s}, x_{1-s+1} \cdots, x_{m-1}, x_m, \underbrace{0, \cdots, 0}_{|s|}), & s < 0 \end{cases} \tag{4}$$

There is no restriction on shift s in original SBD [32]. In our problem, in order to deal with constrained periodicity drift, we

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.2019.2919327, IEEE Transactions on Network and Service Management

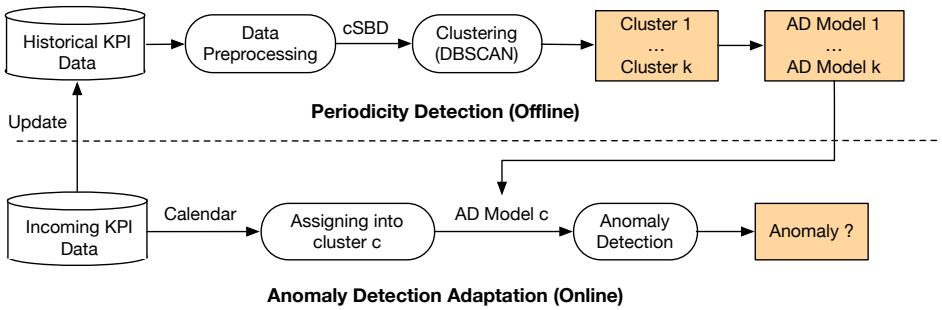


Fig. 7: Overall framework of *Period*.

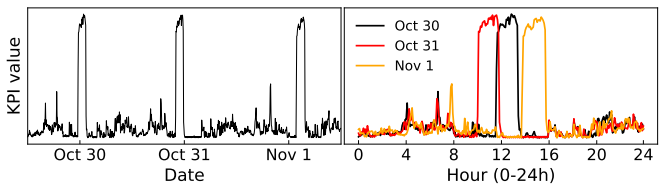


Fig. 8: An intuitive example about constrained periodicity drift. The three-day-long fragment of a KPI is plotted on the left, and the three daily subsequences are plotted separately on the right.

define a shift threshold w (e.g., 2 hours) and set $s \in [-w, w]$. When all possible $\vec{x}_{(s)}$ are considered, we get $CC_s(\vec{x}, \vec{y})$ as the similarity between sequences $\vec{x}$ and $\vec{y}$ with a time shift s. It is defined as:

$$CC_s(\vec{x}, \vec{y}) = \begin{cases} \sum_{i=1}^{m-s} x_i \cdot y_{s+i}, & s \geqslant 0 \\ \sum_{i=1}^{m+s} x_{i-s} \cdot y_i, & s < 0 \end{cases}, s \in [-w, w] \quad (5)$$

The cross-correlation is the maximum value of $CC_s(\vec{x}, \vec{y})$, which means the distance between $\vec{x}$ and $\vec{y}$ at the optimal shift s. In practice, a normalized version of cross-correlation (NCC) is often used to limit the values to be within [-1, 1], where 1 indicates a perfect similarity and -1 indicates the two sequences are completely different. NCC is defined as follows:

$$NCC(\vec{x}, \vec{y}) = \max_s \frac{CC_s(\vec{x}, \vec{y})}{\|\vec{x}\|_2 \cdot \|\vec{y}\|_2}, \quad s \in [-w, w] \quad (6)$$

Then we can define cSBD based on NCC:

$$cSBD(\vec{x}, \vec{y}) = 1 - NCC(\vec{x}, \vec{y}) \quad (7)$$

cSBD takes values between 0 and 2, where 0 indicates perfect similarity. A smaller cSBD value means higher similarity. The effectiveness of cSBD compared with other distance measures will be demonstrated in §IV-C.

3) Clustering: Many algorithms have been proposed for time series clustering [33]. In general, all approaches modify existing algorithms, either by replacing the default distance measures with a version that is more suitable for time series (raw-data-based methods) or by extracting several features from raw time series so that they can be directly used in classical algorithms (feature-based and model-based methods). We argue that feature-based and model-based methods often make strong assumptions (e.g., assuming that time series can be modeled using Gaussian mixture [34]) and lose much information in feature extraction. Therefore, we follow a raw-data-based approach in our algorithm.

Two popular raw-data-based methods are partitional and density clustering. K-means [35] and k-medoids [36] are widely-used partitional methods due to their simplicity and effectiveness. However, the number of clusters k and the initial centers of each cluster need to be predetermined, which will result in unstable results with different initial centers. Besides, k-means requires the computation of artificial sequences as centroids, which hinders the easy adaptation of distance measures other than Euclidean distance. Density methods like DBSCAN [37] find dense regions separated by low-density areas to form clusters. A cluster is expanded if its neighbors are dense, i.e., others that are similar to its core will be absorbed into it. We opt to use DBSCAN for the following two reasons. First, since KPIs are collected from various applications and systems, operators have no prior knowledge about the number of clusters, while DBSCAN can infer k based on the data. Second, it can discover clusters of arbitrary shape and can work with most distance measures. As a comparison, k-means usually discovers spherical clusters and is applicable only to Euclidean distance. The effectiveness of DBSCAN will be demonstrated in §IV-D.

DBSCAN has two input parameters: the minimum size of a cluster $minPts$ and density radius ϵ, which is the maximum distance between samples (daily subsequences in our case) in a cluster. The key idea of DBSCAN is to find the cores in dense regions, and then expand cores by transitivity of distance to form clusters. A core p is defined as a sample which has at least $minPts$ samples within a distance of ϵ from it (excluding p). Only cores can be used to expand clusters. In other words, only samples with the distance smaller than ϵ from a core can be absorbed into a cluster. Any sample that has fewer neighbors than $minPts$ is declared to be an outlier that is not associated with any cluster.

We adopt the elbow method to determine density radius ϵ [38]. If ϵ is too small, a large part of data cannot be clustered; whereas for a too high ϵ, clusters will be merged and the majority of samples will be in the same cluster. For $minPts = c$, to determine the parameter ϵ, we need to look at the behavior of the distance from the sample to its c-th nearest neighbor, which is called c-dis. The c-dis is computed for all samples, then all c-dis values are plotted in descending order as shown in Fig. 9 ($c = 4$). A flat area indicates the density

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.2019.2919327, IEEE Transactions on Network and Service Management

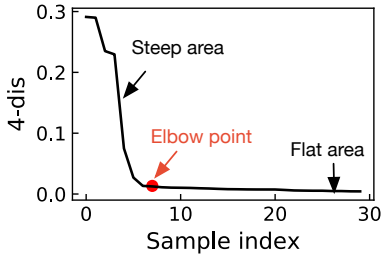


Fig. 9: 4-dis graph with 30 samples. The steep area indicates sharp density changes and the red point denotes elbow point.

around a number of samples is consistent, while a steep area means significant density changes. The elbow point (denoted by the red point) is the junction between steep area and flat area, and the corresponding c-dis value is a suitable density radius ϵ [39]. However, [37] argued that it is very difficult to discover the elbow point automatically and proposed to follow an interactive approach for determining the threshold point by users. In our algorithm, in order to find the elbow point automatically, considering the c-dis values in the flat area are similar and all the distances are normalized between 0 and 2, we set a slope threshold (0.001 is used) conservatively. That is to say, we check the difference between two adjacent c-dis values from left to right in c-dis graph. Once the difference is smaller than 0.001, the point is regarded as elbow point and the c-dis value is the optimal density radius. We set $minPts = 4$ as suggested in [37], since experiments in [37] indicate that c-dis graphs for $c > 4$ do not significantly differ from the 4-dis graph and they need considerably more computation.

Furthermore, the computational complexity about offline periodicity detection needs to be considered. The time complexity of computing cSBD distance is $O(n^2 m \log m)$ [32], where m is the length of daily subsequences and n is the number of daily subsequences. The clustering with DBSCAN requires $O(n \log n)$, and the time complexity of density estimation is $O(n)$. Therefore, the total running time largely depends on the complexity of computing cSBD distance.

C. Adaptation for Anomaly Detection

Through the above daily subsequences clustering, each cluster denotes a kind of daily pattern. Afterwards, we need to assign a specific name to represent each cluster so that the incoming data can be assigned into the correct cluster. The detailed assigning strategy is displayed in Algorithm 1. First, the clusters are sorted in ascending order according to the number of samples in this cluster. This is because the clusters with fewer samples generally represent special daily patterns, and are more easily identified by dates. Then the date features for each daily subsequence are extracted in first $k-1$ clusters. For each value v of each feature f, we compute $ratio(f, v) = \frac{\#f=v \text{ in cluster } i}{\#f=v \text{ in all clusters}}$ for each pair of (f, c). The $f = v$ with the maximum ratio is the name of cluster i. The key idea here is using the frequent feature in cluster i while it is infrequent in other clusters. For the last cluster k, we do not need to assign a name for it specifically, since the last cluster with most samples represents the most common daily pattern, thus we can name it as the remaining days that do not belong to first $k-1$ clusters. For example, the names of two clusters of KPI $\mathcal{D}$ are the day of week is "Friday" and "remaining days (not Friday)". Notice that we only list some common date features here, and new date features can be added into the model according to the real applications, e.g., business promotion days.

Afterwards, subsequences in a cluster can be concatenated into a new sub-KPI with perfect daily periodicity, which can be solved with original anomaly detection methods, thus we can apply anomaly detection methods (e.g., Holt-Winters, Donut) on each cluster, respectively. For online incoming KPI data, it can be assigned into the correct cluster according to the clusters' name and public holiday calendar, then the corresponding model is adopted to detect anomalies. In summary, in order to tackle diverse periodicity adaptation for KPI anomaly detection problem, we novelly transform the anomaly detection on a KPI with unknown periodicity profile into anomaly detection on k sub-KPIs with clear daily periodicity, and the k sub-KPIs are obtained through daily subsequences clustering.

Algorithm 1: Assigning specific names for clusters.

Input: Clustering results and public holiday calendar
Output: Clusters with specific names

Sort the clusters in ascending order according to the number of samples in the cluster;
k = the number of clusters;
$i = 1$;
`/* Assigning names for first` $k-1$ `clusters */`
while $i < k$ **do**
  for *each daily subsequence in cluster i* **do**
    Get the date of this daily subsequence;
    Extract date features (day of week, day of month, off day or not, festival or not, etc.);
  end
  for *each feature f* **do**
    for *each value v of feature f* **do**
      compute $ratio(f, v) = \frac{\#f=v \text{ in cluster } i}{\#f=v \text{ in all clusters}}$
    end
  end
  The $f = v$ with the maximum ratio is the name of cluster i;
  $i+ = 1$;
end
The last cluster k is named as remaining days that do not belong to first $k-1$ clusters.

IV. Evaluation of Periodicity Detection

This section mainly focuses on evaluating the performance of periodicity detection, i.e., daily subsequences clustering. We first introduce the datasets and the metric used in the experiments. Then we show the effectiveness of our clustering algorithm by comparing with other distance measures and clustering algorithms.

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.2019.2919327, IEEE Transactions on Network and Service Management

TABLE I: Six KPIs in our experiments, where n is the number of days and k is the true number of different daily patterns.

KPIs	Name	Interval (min)	n	k
$\mathcal{A}$	Transaction per second (TPS)	1	186	3
$\mathcal{B}$	CPU utilization	5	112	2
$\mathcal{C}$	Memory utilization	30	336	2
$\mathcal{D}$	CPU utilization	5	112	2
$\mathcal{E}$	CPU utilization	5	184	4
$\mathcal{F}$	Transaction per second (TPS)	1	105	3

A. Datasets

We collected six KPIs named $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}, \mathcal{F}$ from a large commerical bank through their real-time monitoring systems. The details about these KPIs are displayed in Table I, where n is the number of days and k is the true number of different daily patterns. For the purpose of evaluating the performance of periodicity detection, the experienced operators carefully check the number of different daily patterns of each KPI and provide the value of k. To present the periodicity profiles of these six KPIs intuitively, we plot a 2-week-long fragment of each KPI, which has been presented in Fig. 3. In detail, KPI $\mathcal{A}$ ($k = 3$) has three different daily patterns, i.e., work days, off days (including New Year's Day) and Spring Festival. KPI $\mathcal{B}$ and KPI $\mathcal{C}$ ($k = 2$) behave differently between work days and off days. KPI $\mathcal{D}$ ($k = 2$) has a peak on every Friday and other days behave similarly. KPI $\mathcal{E}$ ($k = 4$) has an increasing peak from Thursday to Sunday. For KPI $\mathcal{F}$ ($k = 3$), Monday, Wednesday and Friday are different from other days, and Chinese National Day (from Oct 1 to Oct 7) also behaves differently. Besides, KPI $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{E}$ have the periodicity drift.

From Table I, we can observe that the same monitoring indicator of different services have different periodicity profiles, e.g., CPU utilization (KPI $\mathcal{B}, \mathcal{D}, \mathcal{E}$) and TPS (KPI $\mathcal{A}, \mathcal{F}$). Therefore, it is infeasible for operators to manually check or predefine the periodicity profile for every monitoring indicator due to diverse services. Here we only take these six representative KPIs as examples to prove the robustness and effectiveness of our clustering algorithm. We are confident that the six KPIs are general enough to represent other kinds of KPIs, and we use 50 more KPIs to evaluate the periodicity adaptation for anomaly detection in §V.

B. Metric

In order to quantitatively evaluate *Period*'s clustering performance, we adopt the following two popular metrics.

Clustering Accuracy (ACC) discovers the one-to-one relationship between resulting clusters and true classes, and measures the extent to which each cluster contains subsequences from the corresponding class. It is defined as follows:

$$ACC = \frac{\sum_{i=1}^{n} \delta(map(r_i), l_i)}{n}$$

where r_i denotes the predicted cluster label of daily subsequence x_i and l_i denotes the true class label of x_i, n is the total number of daily subsequences, and $\delta(x, y)$ is the delta function that equals one if $x = y$ and equals zero otherwise,

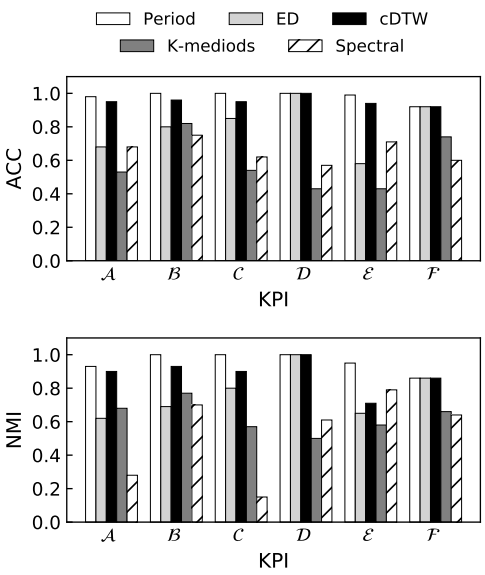


Fig. 10: ACC and NMI comparisons of different distance measures and clustering algorithms.

and $map(r_i)$ is the permutation mapping function that maps each cluster label r_i to the equivalent true class label.

Normalized Mutual Information (NMI) is used for determining the quality of clusters. Given a clustering result, the NMI is estimated by:

$$NMI = \frac{\sum_{i=1}^{k_d} \sum_{j=1}^{k} n_{i,j} \log \frac{n_{i,j}}{n_i \hat{n_j}}}{\sqrt{(\sum_{i=1}^{k_d} n_i \log \frac{n_i}{n})(\sum_{j=1}^{k} \hat{n_j} \log \frac{\hat{n_j}}{n})}}$$

where k_d is the number of resulting clusters, k is the number of true classes, n_i denotes the number of subsequences contained in the cluster i ($1 \leqslant i \leqslant k_d$), $\hat{n_j}$ is the number of subsequences belonging to true class j ($1 \leqslant j \leqslant k$), and $n_{i,j}$ denotes the number of subsequences that are in the intersection between cluster i and the class j [40].

Besides, we record k_d denoting the number of clusters found by clustering algorithm and compare with the true k in Table I. The running time is also recorded to evaluate the efficiency of the clustering algorithm.

C. Comparison with Other Distance Measures

We compare cSBD with another two popular distance measures, i.e., Euclidean distance (ED) and constrained Dynamic Time Warping (cDTW) [41], to illustrate the effectiveness of cSBD. The comparison results of ACC, NMI, k_d and running time are displayed in Fig.10. and Table II (rows 3-5).

Clearly, cSBD with DBSCAN finds the number of clusters accurately ($k = k_d$) and outperforms ED on KPI $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{E}$. This is because ED cannot handle the time shift problem, as showcased in Fig. 8, and these KPIs exhibit periodicity drift. When calculating distance, ED assumes the i-th point in one sequence is aligned with the i-th point in the other, thus they are sensitive to small distortions along the time dimension. For KPI $\mathcal{D}$ and $\mathcal{F}$ without periodicity drift, ED performs as well as cSBD. Besides, we can observe that cSBD also behaves better than cDTW on some KPIs. cDTW calculates

TABLE II: Running time (seconds) and k_d comparisons of different distance measures and clustering algorithms.

KPIs	$\mathcal{A}$ (True k=3)		$\mathcal{B}$ (True k=2)		$\mathcal{C}$ (True k=2)		$\mathcal{D}$ (True k=2)		$\mathcal{E}$ (True k=4)		$\mathcal{F}$ (True k=3)	
Metrics	k_d	Time	k_d	Time	k_d	Time	k_d	Time	k_d	Time	k_d	Time
***Period* (cSBD,DBSCAN)**	**3**	**35.2**	**2**	**1.2**	**2**	**2.9**	**2**	**0.9**	**4**	**1.0**	**3**	**10.3**
(ED, DBSCAN)	4	2.5	3	0.8	3	1.1	2	0.5	5	0.7	3	2.2
(cDTW, DBSCAN)	3	1785.9	2	182.4	2	219.4	2	179.9	3	114.5	3	543.9
(cSBD, k-mediods)	2	55.1	3	2.4	3	3.3	3	2.2	2	1.2	2	7.8
(cSBD, spectral)	2	47.2	2	2.9	4	3.0	3	3.5	2	2.1	2	8.6

the minimized accumulated aligning cost as their distance, thus it has the ability to tackle the constrained periodicity drift. However, cDTW has the risk of distorted alignment in the process of dynamic programming (just like overfitting). Moreover, the running time is mainly affected by the number of clustering samples (n in Table I) and the length of daily subsequence (m). Since both KPI $\mathcal{A}$ and $\mathcal{F}$ have an one-minute monitoring interval, their running time is much longer than others. Although compared with original DTW which has $O(m^2)$ computational complexity, cDTW only needs $O(wm)$ time where w is the threshold of constrained periodicity drift range just like w in cSBD. However, as shown in Table II, cDTW still suffers from high computational cost, about 2 orders of magnitudes higher than cSBD.

In summary, the results reveal that taking into account both performance and efficiency, cSBD is the best distance measure in our problem.

D. Comparison with Other Clustering Algorithms

In order to demonstrate the effectiveness of DBSCAN, we compare it with another two popular clustering algorithms, i.e., k-mediods [42] and spectral clustering [43]. K-medoids divides the data into groups and attempts to minimize the distance between samples labeled to be in a cluster and a sample designated as the center of that cluster. The reason why k-means is not compared here is that k-means adopts the artificial sequences as new centroids, which hinders the easy adaptation of distance measures other than Euclidean distance. Considering that k-mediods is an unstable algorithm, we run the experiment for ten times, then compute the average ACC and NMI. Spectral clustering derived from graph theory makes use of the spectrum of the similarity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions. Besides, since both k-mediods and spectral clustering need k as their input variables, we adopt Silhouette Analysis [44] to determine the best k.

Fig.10 and Table II (rows 6-7) present the comparison results. Clearly, DBSCAN consistently outperforms other clustering algorithms on all KPIs and discovers the number of clusters accurately ($k = k_d$). The performance of k-mediods greatly depends on the selection of initial cluster center, and spectral clustering is not suitable for high dimensional data. Moreover, k-mediods and spectral clustering need to run several experiments with different k to determine the best k by Silhouette Analysis, but DBSCAN can infer k based on the data by itself.

In summary, the results demonstrate that DBSCAN is the most suitable clustering algorithm in our problem.

V. Evaluation of Anomaly Detection Adaptation

This section mainly focuses on evaluating the performance and robustness of anomaly detection adaptation to diverse periodicity profiles. We first introduce the datasets and metric. Then anomaly detection methods adopted in our evaluation are presented. Finally, we present the comparison results and provide a detailed empirical analysis.

A. Datasets

In terms of datasets, in addition to the 6 KPIs introduced in §IV-A, another 50 KPIs are collected from a large commercial bank in order to further show the performance of *Period*. We collected these representative KPIs from the bank's IT environment which contains more than 200 application systems and more than 5000 virtual machines. These KPIs are chosen based on the following principles. 1) These KPIs fall into two categories: one is the transaction indicators (high-level) such as transactions per second and order volume, which are obtained by Application Performance Management (APM) or aggregating transaction logs from each application entry; the other is monitoring indicators of virtual machines (low-level) such as CPU utilization and memory utilization. 2) These KPIs contain various kinds of periodicity profiles, which are generic enough for evaluation. 3) All of KPIs have a time span of about 3-6 months. The first two months of each KPI are used as historical data for offline periodicity detection, and the rest (1-3 months) is used to evaluate online anomaly detection with periodicity adaptation. In practice, it would be better if the historical data is long-term enough (e.g., one year) to include all kinds of daily patterns. 4) In order to ensure the high quality of data, these KPIs do not have too many missing points. Therefore, we are confident that these 56 KPIs are representative enough to demonstrate the performance of *Period*. Besides, experienced operators have labeled these KPIs carefully as the ground truth. Notice that we only display *detailed* results about the 6 KPIs in the interest of space, and results of other 50 KPIs are displayed briefly.

B. Metric

In general, anomaly detection methods compute an anomaly score for each data point. If the anomaly score for the point is larger than the threshold, an alert should be triggered. In this way, we can compute the precision and recall corresponding to each threshold. F-score as the harmonic mean of precision and recall can be computed as $\frac{2\times\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$. However, threshold selection in anomaly detection is a very tricky problem, since the best thresholds are different among

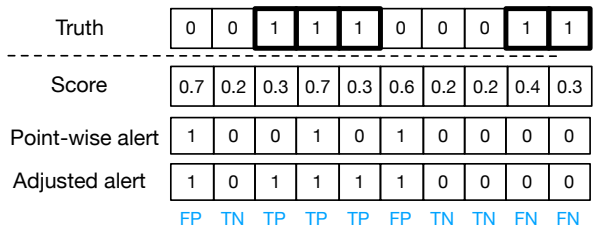


Fig. 11: Illustration of the strategy for modified metric. The first row is the truth with 10 contiguous points and two anomalous segments highlighted in the shaded squares. The detector scores are shown in the second row. The third row shows the point-wise detector results with a threshold of 0.5. The fourth row shows the detector results after adjustment. The last row indicates each point is true positive (TP), false positive (FP), true negative (TN) or false negative (FN). We shall get precision= $\frac{\text{TP}}{\text{TP+FP}}$ =0.6, recall= $\frac{\text{TP}}{\text{TP+FN}}$ =0.6 and F-score 0.6.

different KPIs. In general, there are several popular threshold selection methods, for example, choosing the threshold that performs best in training set. In this paper, in order to avoid the threshold selection problem, we may enumerate all thresholds and use the best F-score as the metric, which indicates the best possible performance given an optimal threshold. In practice, the threshold can be decided by operators based on their knowledge and experience. Moreover, since operators do not care about the point-wise alerts in real applications, instead they prefer to trigger an alert for any time in a continuous anomalous segment. Thus following [4], we adopt the modified anomaly detection metric. In detail, if any point in an anomalous segment in the ground truth can be detected (over a given threshold), we say all points in this segment are treated as they have been detected. Meanwhile, the points outside the anomalous segments are treated as usual. The precision, recall, and F-score are then computed accordingly. This modified metric is illustrated in Fig. 11. Therefore, the modified best F-score is adopted as evaluation metric in the following experiments.

C. Anomaly Detection Methods in Experiments

Given different anomaly detection methods perform differently on different KPIs [3], in order to show *Period*'s robustness and generality, we choose five widely-used anomaly detectors, i.e., Holt-Winters (HW) [1], Time Series Decomposition (TSD) [11], Difference (Diff) [3], Moving Average (MA) [6], Exponential Weighted MA (EWMA) [5], and a state-of-the-art unsupervised anomaly detection algorithm Donut [4] to demonstrate the effectiveness of *Period*. Notice that we only take these six methods as examples, but *Period* is not limited to these anomaly detection methods and instead it can be applied to many other approaches.

Table III describes anomaly detection methods used in experiments. Since HW, TSD and Diff need the season length of KPI as their input parameters, *-day and *-week mean that the seasonality parameters are configured as the length of day and week, respectively. MA, EWMA and Donut does not need

TABLE III: Anomaly detection methods used in our experiments. The first column is the basic anomaly detection methods. The second column is the variants of the basic methods compared in experiments.

Basic methods	Methods used in experiments
HW	HW-day; HW-week; HW-*Period*
TSD	TSD-day; TSD-week; TSD-*Period*
Diff	Diff-day; Diff-week; Diff-*Period*
MA	MA; MA-*Period*
EWMA	EWMA; EWMA-*Period*
Donut	Donut; Donut-*Period*

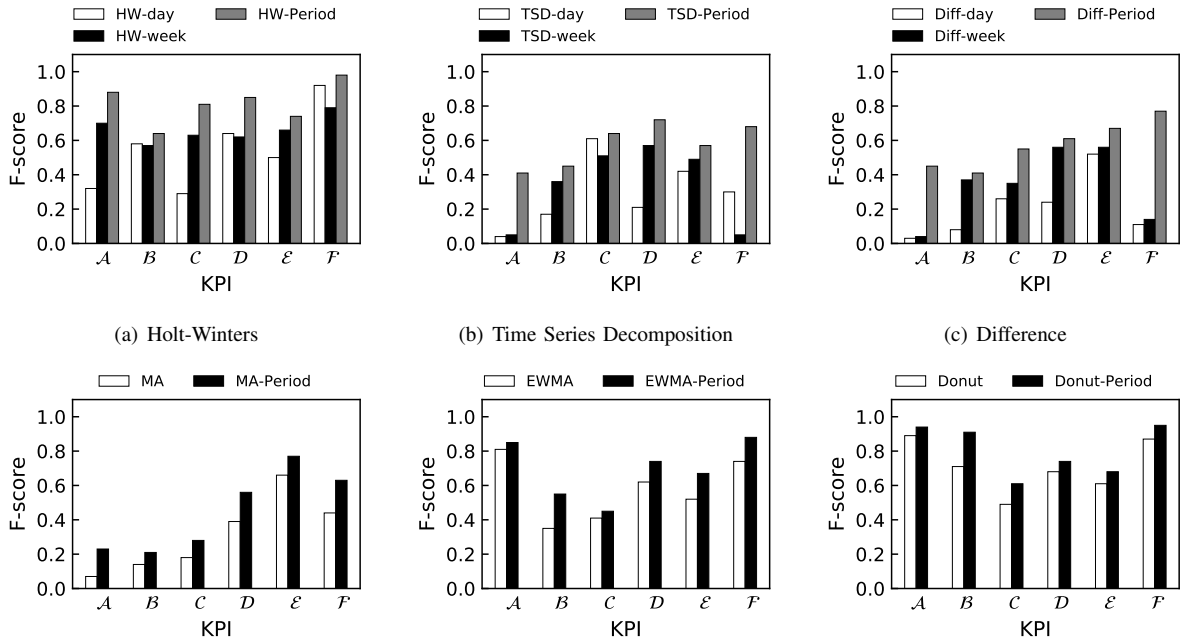
the season length as parameter, and "MA, EWMA, Donut" in Table III column 2 means the original anomaly detection approach. *-*Period* means applying *Period* framework to these anomaly detection methods (see Fig. 6 and Fig. 7). In detail, based on the offline daily subsequences clustering results on historical data, *-*Period* prepares k separate anomaly models for k historical sub-KPIs independently. Then for testing set, the data is assigned to the corresponding cluster according to the calendar and current date, and the corresponding model is used to detect anomalies.

D. Results and Analysis

Fig. 12 shows the best F-score comparison between *Period* and compared methods on six KPIs (Table I). Besides, in order to further illustrate the effectiveness of *Period*, the best F-score of another 50 KPIs with diverse periodicity profiles are displayed in Fig. 13 with box-plot in brief. Based on the results, we have five key observations as follows.

First of all, it is obvious that applying *Period* to anomaly detection methods can significantly improve the best F-score. In detail, it can improve the traditional anomaly detectors (HW, TSD, Diff, MW, EWMA) by up to 0.66, and improve the state-of-the-art unsupervised anomaly detection algorithm Donut by up to 0.2. Besides, by analyzing the best F-score of *Period* in comparison with baselines using the Wilcoxon test [45], we can make a conclusion that *Period* performs better than baselines with a 99% confidence level. Furthermore, in order to investigate the effect of the threshold on performance, we take HW and Donut as examples and plot the ROC graphs on KPI $\mathcal{A}$ to demonstrate the True Positive Rate (TPR) and False Positive Rate (FPR) under different thresholds. As displayed in Fig. 14, it is clear that with the increase of the threshold, *-*Period* with larger Area Under Curve (AUC) can achieve higher TPR and lower FRP than the baselines. In summary, the above observations show that *Period* is robust and generic enough to automatically adapt KPI anomaly detection methods to diverse periodicity profiles.

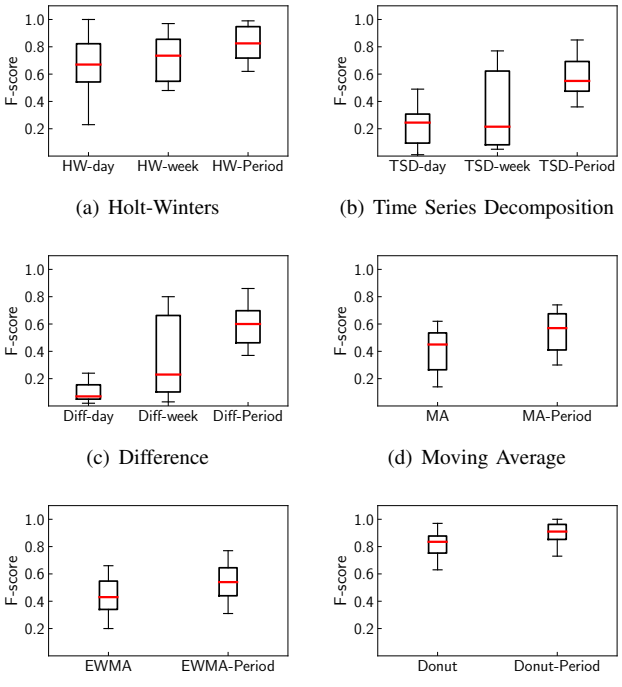
Second, as aforementioned, the selection of seasonality parameter has a great impact on anomaly detectors (HW, TSD, Diff) and it is labor intensive to configure the seasonality parameter manually when facing a large number KPIs. Besides, although some KPIs have perfect weekly periodicity, such as KPI $\mathcal{B}$ and $\mathcal{C}$, *-week methods still perform not ideally on these KPIs. This is because there are also similar daily patterns within a season, for example, weekdays in KPI $\mathcal{B}$.



(a) Holt-Winters (b) Time Series Decomposition (c) Difference

(d) Moving Average (e) Exponential Weighted MA (f) Donut

Fig. 12: Best F-score of *Period* in comparison with baseline algorithms on six KPIs.

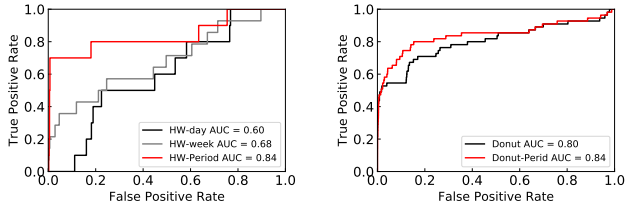


(a) Holt-Winters (b) Time Series Decomposition

(c) Difference (d) Moving Average

(e) Exponential Weighted MA (f) Donut

Fig. 13: Improved performance by *Period* on another 50 KPIs.



(a) Holt-Winters (b) Donut

Fig. 14: ROC graphs of Holt-Winters and Donut on KPI $\mathcal{A}$.

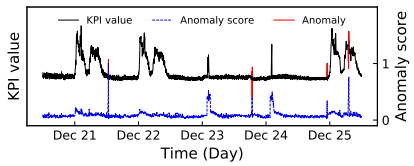
Simply using the length of week as the seasonality parameter will ignore the similarity within a season and lose much information. However, with *Period*, all similar patterns will be assigned into a cluster and share an anomaly detection model, so that the historical similar patterns can be fully utilized.

Third, despite MA and EWMA do not need the season length as parameters, various periodicity profiles also have a marked impact on the performance of them. This is because both MA and EWMA are time series prediction models which make strong assumptions on the stationarity and seasonality of input KPI data, but imperfect or complex periodicity will destroy the stationarity and seasonality. However, with *Period*, each sub-KPI with perfect daily periodicity can satisfy the assumption and lead to better performance.

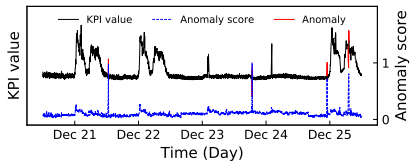
Fourth, despite Donut performs better and more stable than other five anomaly detectors, it is also a little sensitive to the diverse periodicity profiles, since the original Donut has no time or date information in its input variables, thus it is difficult for Donut to capture the periodicity profile while learning the normal pattern to detect anomalies in training data. While with *Period*, we can concatenate all subsequences in a cluster into a new sub-KPI with clear daily periodicity and apply original Donut on each seasonal sub-KPI respectively, which can improve the best F-score by up to 0.2.

Fifth, even though different anomaly detection methods perform differently on different KPIs, *Period* can improve the performance of all method. For example, the absolute best F-score of some *-*Period* algorithms are not very good (e.g., TSD-*Period*, Diff-*Period* and MA-*Period* in KPI $\mathcal{A}$, $\mathcal{B}$)

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.2019.2919327, IEEE Transactions on Network and Service Management



(a) Donut



(b) Donut-Period

Fig. 15: A toy example to show the anomaly scores of Donut, with and without *Period*.

because these original detectors' incapability to deal with these KPIs. Even so, *Period* can still improve their best F-score by up to 0.41. On the other hand, original Donut is able to deal with KPI $\mathcal{A}$ and $\mathcal{B}$ reasonably well, and *Period* can still improve its best F-score by up 0.2.

Furthermore, in order to illustrate how *Period* detects anomalies more accurately, we take the KPI $\mathcal{B}$ as an example and plot the anomaly scores (normalized between 0 and 1) given by Donut, with and without *Period*. As Fig. 15 shows, the original Donut without *Period* (Fig. 15(a)) regards weekends (Dec. 23 and Dec. 24) as unexpected patterns and gives relative high anomaly scores, which will lead to some false alarms. However, with the adaptation of *Period* (Fig. 15(b)), weekdays and weekends are used to train a Donut model separately, which will give more reasonable anomaly scores and detect anomalies more accurately.

In conclusion, the above results fully demonstrate that *Period* is robust and generic to automatically adapt KPI anomaly detection methods to diverse periodicity profiles.

E. Comparison with Multivariate Time Series Anomaly Detection

One intuitive method to solve KPI anomaly detection with diverse periodicity profiles is multivariate time series anomaly detection (MTSAD for short) [46]–[48] by encoding date features into variables (see Algorithm 1, day of week, off day or not, etc). Considering that Donut is the state-of-the-art unsupervised KPI anomaly detection and shows good and stable performance in the above experiments, we compare the performance of Donut-*Period* with one of the state-of-the-art MTSAD method LSTM-NDT [46]. The results on six KPIs are displayed in Fig. 16.

It is clear that Donut-*Period* achieves a much better F-score compared with MTSAD. We have the following observations based on the results. First, MTSAD mainly focuses on the overall status of an entity containing multiple metrics (entity-level) and univariate time series anomaly detection only aims to detect anomalies in one metric (metric-level) [46]–[48]. In our scenario, it may be a little inappropriate to solve the problem with MTSAD. Essentially, we only need to detect anomalies in the raw KPI (metric-level) instead of entity-level. Although we get multivariate time series by encoding

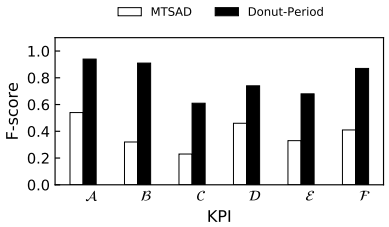


Fig. 16: Best F-score comparison between Donut-*Period* and MTSAD.

date features into variables, these date feature time series are not metrics to characterize the status of an entity, and it may be unreasonable to encode discrete variables into time series. Besides, the multi-dimensional vector in our problem only contains one raw KPI variable and several date feature variables, and the value of raw KPI only plays a small role in the vector, which may affect the performance of the model and fail to detect anomalies. Second, the majority of popular MTSAD algorithms in recent years are based on neural network and suffer from high computational complexity. According to our observation, it takes about several hours to train the LSTM-NDT model. Third, compared with MTSAD, our proposed model *Period* is more intuitive and enhances the interpretability. It can discover special days with different patterns for operators.

In summary, MTSAD and *Period* tackle KPI anomaly detection with diverse periodicity profiles from different angles. More specifically, MTSAD tries to proposes a generic anomaly detection model incorporating date information to solve the problem, while *Period* is based on the existing anomaly detection methods and aims to enable these methods to adapt to diverse periodicity profiles. However, existing MTSAD methods are not applicable to the problem. In our future work, we may try other deep learning models incorporating date information based on the idea of MTSAD.

VI. Discussion and Future Work

Period has two main limitations. First, *Period* assumes that the basic seasonality of KPI is one day, so we can cut the whole KPI into daily subsequences directly. It is intuitive because most KPIs are associated with daily human activities, for example, the transactions per second of an Internet bank is higher in the day time than that at night in a common diurnal cycle. However, *Period* cannot handle a KPI with arbitrary seasonality such as monthly and yearly. Second, we demonstrate the effectiveness of *Period* only on the 56 KPIs collected from the bank's IT systems. However, we cannot guarantee *Period* to be able to always perform well on all types of time series data. For example, *Period* cannot be applied simply to KPIs with both trend pattern and periodic pattern, and detrending preprocessing is needed.

About the future work, as we mentioned before, *Period* cannot handle a KPI with arbitrary seasonality such as monthly and yearly. We may consider designing an algorithm to discover the basic seasonality automatically (e.g., Periodogram, ACF or other methods) and divide subsequences according to the detected basic seasonality. Second, it would be more convincing to use more KPIs from other Internet-based applica-

tions (e.g., search engine and online shopping) or various time series data in other domains, so that we can find some cases that our algorithm cannot handle and improve it to be more generic. Third, we may consider designing a more effective model based on the idea of multivariate time series anomaly detection or trying other deep learning models incorporating date information to solve the problem.

VII. Conclusion

In this paper, we present an automatic and generic framework called *Period* to enable anomaly detection methods to deal with KPIs with diverse periodicity profiles. This is the first work to study this problem, to the best of our knowledge. The core idea of *Period* is transforming the anomaly detection on a given KPI with unknown periodicity profile into anomaly detection on k sub-KPIs with clear daily periodicity, and the k sub-KPIs can be acquired by our proposed daily subsequences clustering technique. Based on that, *Period* successfully detect periodicity profiles and distinguish different daily patterns with a high accuracy about 0.95 on average. More importantly, our evaluation using 56 real-world KPIs demonstrates that *Period* can significantly improve the best F-score by up to 0.66 for several anomaly detection methods.

We believe *Period* is robust and generic enough to assist existing anomaly detection methods in adapting diverse periodicity profiles automatically, so as to detect anomaly more accurately and ensure more reliable service in real applications.

Acknowledgement

We thank the anonymous reviewers for their valuable feedbacks. We thank Juexing Liao for her helpful suggestions and proofreading. This work has been supported by the Beijing National Research Center for Information Science and Technology (BNRist) key projects, the Global Talent Recruitment (Youth) Program and Okawa Research Grant.

References

[1] H. Yan, A. Flavel, Z. Ge, A. Gerber, and et al., "Argus: End-to-end service anomaly detection and localization from an isp's point of view," in *2012 Proceedings IEEE INFOCOM*, 2012.

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, Jul. 2009.

[3] 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 *Proceedings of the 2015 Internet Measurement Conference*. ACM, 2015, pp. 211–224.

[4] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2018, pp. 187–196.

[5] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: Methods, evaluation, and applications," in *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '03. New York, NY, USA: ACM, 2003, pp. 234–247.

[6] D. R. Choffnes, F. E. Bustamante, and Z. Ge, "Crowdsourcing service-level network event monitoring," in *Proceedings of the ACM SIGCOMM 2010 Conference*, ser. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 387–398.

[7] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1939–1947.

[8] J. Dromard, G. Roudière, and P. Owezarski, "Online and scalable unsupervised network anomaly detection method," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 34–47, 2017.

[9] A. A. Mahimkar, H. H. Song, Z. Ge, A. Shaikh, J. Wang, J. Yates, Y. Zhang, and J. Emmons, "Detecting the performance impact of upgrades in large operational networks," in *Proceedings of the ACM SIGCOMM 2010 Conference*, ser. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 303–314.

[10] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, 2017.

[11] Y. Chen, R. Mahajan, B. Sridharan, and Z.-L. Zhang, "A provider-side view of web search response time," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 243–254.

[12] M. Ma, S. Zhang, D. Pei, X. Huang, and H. Dai, "Robust and rapid adaption for concept drift in software system anomaly detection," in *Proceedings of the 2018 IEEE International Symposium on Software Reliability Engeering (ISSRE)*. IEEE, 2018.

[13] Offical holiday calendar, http://www.gov.cn/zhengce/content/2017-11/30/content_5243579.htm.

[14] API for public holiday calendar, https://github.com/qiu8310/holiday.

[15] P. G. Gould, A. B. Koehler, J. K. Ord, R. D. Snyder, R. J. Hyndman, and F. Vahid-Araghi, "Forecasting time series with multiple seasonal patterns," *European Journal of Operational Research*, vol. 191, no. 1, pp. 207–222, 2008.

[16] M. Züfle, A. Bauer, N. Herbst, V. Curtef, and S. Kounev, "Telescope: A hybrid forecast method for univariate time series," in *Proceedings of the International work-conference on Time Series*, 2017.

[17] Anodot, https://www.anodot.com/.

[18] Splunk, https://www.splunk.com/.

[19] Datadog, https://www.datadoghq.com/.

[20] Dynatrace, https://www.dynatrace.com/.

[21] Prometheus, https://www.elastic.co/.

[22] Grok, https://www.grokstream.com/.

[23] Kibana, https://www.elastic.co/.

[24] T. A. Meir Toledano, "Fast automated detection of seasonal patterns in time series data without prior knowledge of seasonal periodicity," U.S. Patent 10 061 677B2, May 17, 2018.

[25] M. G. Elfeky, W. G. Aref, and A. K. Elmagarmid, "Periodicity detection in time series databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 7, pp. 875–887, 2005.

[26] M. Toller and R. Kern, "Robust parameter-free season length detection in time series," in *SIGKDD Workshop on Mining and Learning from Time Series*, 2017.

[27] M. Vlachos, P. Yu, and V. Castelli, "On periodicity detection and structural periodic similarity," in *Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005*, 04 2005.

[28] F. Rasheed, M. Alshalalfa, and R. Alhajj, "Efficient periodicity mining in time series databases using suffix trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 1, pp. 79–94, 2011.

[29] B.-K. Yi and C. Faloutsos, "Fast time sequence indexing for arbitrary lp norms," in *Proceedings of the 26th International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers Inc., 2000.

[30] 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, pp. 359–370.

[31] T. Warren Liao, "Clustering of time series data-a survey," *Pattern Recogn.*, vol. 38, no. 11, pp. 1857–1874, Nov. 2005.

[32] J. Paparrizos and L. Gravano, "k-shape: Efficient and accurate clustering of time series," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 2015, pp. 1855–1870.

[33] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah, "Time-series clustering – a decade review," *Information Systems*, 2015.

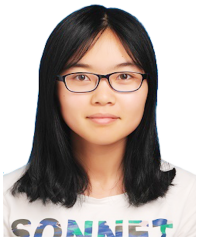
[34] C. Biernacki, G. Celeux, and G. Govaert, "Assessing a mixture model for clustering with the integrated completed likelihood," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 7, pp. 719–725, July 2000.

[35] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1967.

[36] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[37] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *KDD*, vol. 96, no. 34, 1996, pp. 226–231.

[38] D. J. Ketchen and C. L. Shook, "The application of cluster analysis in strategic management research: an analysis and critique," *Strategic management journal*, vol. 17, no. 6, pp. 441–458, 1996.
[39] M. T. Elbatta and W. M. Ashour, "A dynamic method for discovering density varied clusters," *Int. Journal of Signal Processing, Image Processing, and Pattern Recognition*, vol. 6, no. 1, pp. 123–134, 2013.
[40] D. Cai, X. He, X. Wang, H. Bao, and J. Han, "Locality preserving nonnegative matrix factorization." in *International Joint Conferences on Artificial Intelligence*, vol. 9, 2009.
[41] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and information systems*, 2005.
[42] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 3336–3341, Mar. 2009.
[43] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec 2007.
[44] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.
[45] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.
[46] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 387–395.
[47] P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.
[48] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1544–1551, 2018.



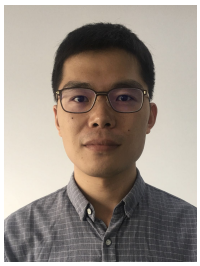
Nengwen Zhao received the B.E. degree in computer science and technology from Wuhan University, Wuhan, China, in 2017. She is currently working toward the Ph.D. degree at Tsinghua University, Beijing, China. Her research work focuses on artificial intelligence for IT operations (AIOps), including anomaly detection, time series analysis, *etc*.

Jing Zhu received his Ph.D degree on Computer Science from Tsinghua University in 2017. During his time as a doctor candidate, he primarily worked on computer networks, especially data center networks and next generation networks protocol designing. Now he serves as a postdoc in Tsinghua University, and his primary research orientation is system operation with artificial intelligence, including anomaly detection, malfunction information extraction, *etc*. He has published several papers on Transactions on Networking, ICNP, WWW, *etc*.

Yao Wang received the B.E degree in network engineering from Beijing University of Posts and Telecommunications, in 2018. He is currently pursuing the M.C.S. degree in Tsinghua University, Beijing, China. His research interests mainly focus on solving operation problems via machine learning approaches.

Minghua Ma received the B.E degree in computer science and technology from Xidian University, in 2016. He is currently pursuing the Ph.D. degree in Tsinghua University, Beijing, China. His research interests include artificial intelligence operations (AIOps), to apply state-of-the-art machine learning techniques to drastically improve the performance and reliability of the targeted networks/applications in the Internet.

Wenchi Zhang received the Bachelor of Engineering from the School of Computer Science, Northeast University in 2018. He is currently an algorithm engineer of BizSeer Technologies Co., Ltd, China, focusing on the field of artificial intelligence for IT operations.



Dapeng Liu received Ph.D. in computer science from Tsinghua University, Beijing, China, in 2016. He is currently the CEO in BizSeer Technologies Co., Ltd, China. His research interests mainly focus on solving operation problems via machine learning approaches.

Ming Zhang received the Bachelor of Engineering in computer software from the School of Computer Science, Beijing University of Technology, Beijing, China , in 2001. He is currently an engineer of the China Construction Bank.

Dan Pei is an Associate Professor in Computer Science Department at Tsinghua University in Beijing, China. Before that he was a researcher at AT&T Research. He received his Ph.D. degree from UCLA in 2005, and his Bachelor's and Master's degrees from Tsinghua University in 1997 and 2000. His current research interests are autonomous IT Operations and AIOps (AI for IT Operations). He is an ACM Senior Member and an IEEE Senior Member. He was selected as an "Expert of China Government's Global Talent Recruitment (Youth Program)" in 2012.