IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 11, 2019, accepted March 29, 2019, date of publication April 15, 2019, date of current version April 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2909750

A Collective Anomaly Detection Approach for Multidimensional Streams in Mobile Service Security

YU WENG AND LEI LIU
School of Information Engineering, Minzu University of China, Beijing 100081, China

Corresponding author: Lei Liu (aishangzoulu@gmail.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61772575, and in part by the National Key Research and Development Program of China under Grant 2017YFB1402101.

ABSTRACT Anomaly detection in many applications is becoming more and more important, especially for security and privacy in mobile service computing domains with the development of mobile internet and mobile cloud computing, in which data are typical multidimensional time series data. However, the collective anomaly detection for multidimensional streams exists lots of problems, owing to the differences between the anomaly detection in multidimensional time series and univariate time series data. For example, the temporal continuity of multidimensional time series is much weaker than univariate time series and it is unreasonable to judge the entire multidimensional data as an anomaly if a certain dimension is abnormal. In this paper, we consider the statistical features of the subsequence of streams, proposing a novel collective anomaly detection algorithm for multidimensional streams based on iForest in a cloud environment, namely iForestFS. When using different features about mobile cloud services' metrics suggested by domain knowledge, iForestFS could detect different kinds of anomalies for mobile service security. Furthermore, we implement a distributed iForestFS using spark framework in order to improve time performance and scalability. The experimental results performed on three datasets (mainly about network security) derived from the UCI repository demonstrate that the proposed algorithm can effectively detect a collective anomaly of multidimensional streams in the security domain.

INDEX TERMS Collective anomaly detection, time series, isolation forest, security in mobile service, multidimensional stream processing.

I. INTRODUCTION

Mobile Cloud Computing (MCC) is a combination of three main parts:they are mobile device, cloud computing and mobile internet [3]. With the help of MCC, a mobile user gets a rich application delivered over the Internet and powered by cloud-backed infrastructure. The importance of Cloud Computing is increasing and it is receiving a growing attention in the scientific and industrial communities. Nowaday's the major concern for mobile user is security and protection in mobile cloud computing. In mobile cloud security domain, data to be analyzed such as system resource metrics(CPU usage) and so on are typical time series data. Time series data is an important high-dimensional data, which was a sequence composed of sampling values of a certain physical quantity of objective objects at different time points arranged in chronological order and was widely used in economic management and engineering fields. Time series data itself has the features of high-dimension, complexity, dynamic, high-noise and easy to achieve large-scale. As a result, time series data mining is one of the ten challenging problems in data mining research area [26].

According to the type of anomaly, it can be divided into point anomalies, the contextual sequence anomalies and collection of anomalies. According to the application domains, it can be divided into intrusion detection, fraud detection, medical anomaly detection and sensor network detection. According to the methods of detection [2], it can be classified into classification based, neural network based, Bayesian network based, support vector machine based, rule based,

The associate editor coordinating the review of this manuscript and approving it for publication was Shuiguang Deng.

distance of nearest neighbors based, clustering based and probability statistics based. In the analysis of time series data, it is of great expectation to find out how these time series data are related at different time periods. This relationship is generally manifested as frequent patterns of change in time series and patterns of infinitesimal changes. This rare pattern of change is called anomalous pattern. The anomaly detection algorithm can be easily applied to many security fields. For example, in the field of intrusion detection, by detecting the abnormal pattern of network traffic, we can infer whether there is intrusion behavior. As in the case of time-series data, a single position can be identified as a contextual outlier, or a set of positions may be identified as a collective outlier.

The security related issues in Mobile Cloud Computing are introduced in two categories: the security for mobile users and the security for data [3]. Security for Mobile Users: Mobile devices such as cellular phone, PDA, and Smartphone are exposed to numerous security threats like malicious codes (e.g., virus, worm, and Trojan horses) and their vulnerability. In addition, with mobile phones integrated global positioning system (GPS) device, they can cause privacy issues for subscribers [20]. Securing Data on Clouds: Although both mobile users and application developers benefit from storing a large amount of data/applications on a cloud, they should be careful of dealing with the data/applications in terms of their integrity, authentication, and digital rights [20]. With the rapid development of information technology and mobile internet, more and more large companies' application volume and server cluster scale have been exploded to serve more mobile users, so that they have to arrange full-time staff to monitor servers for security propose. These server operators often have to be responsible for monitoring hundreds of servers, and they must be dealt with immediately if an anomalous security event occurs. The traditional way in the industry is to monitor hundreds of system indicators by installing monitoring software on the server. If a certain indicator reaches a certain threshold, an alarm is sent to the operation and maintenance personnel, but such threshold-based abnormal alarms maybe wrong to a certain degree. We look at this problem from the perspective of multidimensional time series anomaly detection in mobile service security. Each server has many system resource indicators such as CPU usage, memory usage, IO read/write rate, network read/write rate, etc., and the application on the server also has many indicators, if the application is a JAVA program, we need to monitor the JVM metrics at the same time. If the application relies on the message middleware, it needs to monitor the read/write rate of the message at the same time. Besides it, other metrics such as quality of service(QoS is a critical step in service selection in 5G network environments [28]) and network location(network location could be used for service recommendation [29] [27]) are also can be used for finding security problem in mobile cloud service. Each of the above indicators corresponds to a single time series, so there are hundreds or thousands of time series data for a single server. Considering that a large application may correspond to hundreds of servers, the dimension of multidimensional streams is also grown explosively. By analyzing the anomaly detection of security in mobile service, we find that the collective anomaly detection of multidimensional time series data still has the following problems:

Problem 1: differences between the anomalies of multidimensional time series and the anomalies of single-variable time series data. Owing to the temporal continuity of time series data, univariate time series data anomalies are generally defined as a point of abrupt changes (corresponding to point anomaly) or an unusual shape (corresponding to collective anomalies). However, in mobile service security scenario, the time series data of one system metric, such as a sudden increase in CPU utilization of a certain server at a certain time, cannot be properly judged as an abnormality, because it may be caused by the peak of the business every day; The IO read/write rate suddenly rises during a certain time interval, which may be caused by the application's scheduled read and write offline tasks. However, if some dimensions of multidimensional data are all abrupt during the same time interval, and it is very likely that it is an abnormality. For example, the network quality of the mobile environment is relatively poor, and it is not proper to judge whether it is abnormal through the network time series alone. The multiple time series of the system metrics should be considered comprehensively to judge the abnormality (such as intrusion detection, application failure, etc).

Problem 2: multidimensional time series data has the characteristics of fast,realtime and large scale. Traditional anomaly detection algorithms may not be able to handle nearly real-time anomaly detection, so that abnormal points are flowing past without detecting. Besides it, unlike the anomaly detection algorithm on the static or offline dataset, the whole samples can be put in memory and computed. The streaming data is time-sensitive and difficult to save, which causes that the anomaly detection algorithm could only process the current data stream.

Other problems: multidimensional time series data is large-scale and abnormal samples only account for a small proportion of the overall sample. Manually labeling data is costly, and labeling the entire samples is not feasible.

In mobile cloud computing, we should enable the mobile terminals to have access to powerful and reliable computing resources anywhere and anytime by building a virtual computing environment between the front-end mobile terminals and the back-end cloud servers. Exploiting cloud computing resources/services at the mobile devices makes them thin clients who run various light mobile applications and transfer their computation overhead to the cloud. Thus, we can detect network intrusion in cloud other than in mobile devices. Our method is about the security of mobile cloud service. The goal is to design a network intrusion detector in cloud to detect and prevent anomalous behavior from mobile devices. First, we design a collective anomaly detection method for multidimensional streams. Second, we need to use the data

features suggested by domain knowledge to train the intrusion detector.

The main contributions of this paper are as follows: Firstly, we propose a collective anomaly detection algorithm for multidimensional streams in mobile cloud environment and furthermore improve time performance and scalability of the algorithm by implementing a distributed version. Secondly, the experiment on network security datasets demonstrates that the proposed algorithm is available and effective for anomaly detection in mobile service security domain.

The next part of this paper are organized as follows: Section 2 describes related work. In Section 3, the method proposed in this paper is gradually demonstrated. First, comparing the differences between statistical and machine learning based methods on anomaly detection, and introducing isolation forest algorithm. Secondly, explaining the statistical features we selected. Thirdly, the collective anomaly detection algorithm iForestFS for multidimensional streams is proposed and analyzed. Forth, furthermore redesigning and implementing the algorithm to a distributed version based on selected distributed computation engine. In Section 4, we first demonstrate the experimental datasets and then analyze the experimental results to validate our proposed method. Finally, the conclusions and future research directions are provided.

II. RELATED WORK

In 2009, Chandola *et al.* published ''anomaly detection: A survey'' [1] on ACM CSUR. In the paper, an overview of structured and comprehensive anomaly detection research is sought. Based on the basic approach taken by each technology, they classify the existing technology into different categories. For each category, a fundamental anomaly detection method is provided, and then they show the differences between other methods in this category and the fundamental method. In 2012, Chandola et al. further analyzed the detection of discrete sequence data anomalies and published a survey about the anomaly detection of discrete sequence data in the IEEE Transactions on Knowledge and Data Engineering named ''Anomaly detection for discrete sequences: A survey'' [2]. They classify the existing research into three different categories based on the statement of the problem that it is trying to solve. These questions are structured as follows: 1) to identify abnormal sequences in normal sequence databases; 2) to identify subsequences within an unusually long sequence; and 3) to identify patterns in frequently abnormal sequences. And showed how the concepts of these issues differ from one another, discussing their relevance in different application areas.

Some researchers use time series classification technology for time series anomaly detection. Unlike algorithms commonly used in time series analysis, time series classification takes the entire time series as input, the purpose of which is to assign a discrete label to the sequence. It is more difficult than the general classification problem, mainly because of the unequal length of the time series data to be classified, which makes the ge-eral classification algorithm cannot be directly applied. Even for the same time series, the general classification algorithm is still not suitable for direct application because the values of different sequences in the same location cannot generally be directly compared. In order to solve these difficulties, there are usually two ways: first, to define the appropriate distance measure (the most commonly used distance measure is the DTW distance) so that similar sequences in this measure have the same classification label, and such methods are domain-independent. Second, we first model the time series (using the dependency of the data in the sequence to establish the model), then use the model parameters to construct the equidistant vectors to represent each sequence, and finally train and classify them by the general classification algorithm, and such methods belong to domain-related methods. In [25], methods above are analyzed, and two kinds of methods which are domain-independent and domain-related are compared on different synthetic datasets and actual datasets respectively. The results show that when the training data is less, the domain-related algorithms are more appropriate; on the other hand, domain-independent algorithms are relatively less affected by noise. Some researchers also use other machine learning algorithms. In 2003, Zhang *et al.* used a hierarchical hidden Markov model for anomaly detection [32].

Time series data are large scale and it is very difficult to label the data manually. Some researchers have proposed some unsupervised anomaly detection algorithms. In 2008, Liu *et al.* proposed an unsupervised outlier-based isolated forest algorithm ''Isolation forest'' [13]. Isolated forest is an Ensemble-based fast anomaly detection method with linear time complexity and high precision. It is a state-of-the-art algorithm that meets the requirements of big data processing. It belongs to Non-parametric and unsupervised methods. It is suitable for the anomaly detection of continuous-valued data, and the anomaly is defined as ''easily isolated outlier'' - a point that can be understood as a remote point from a sparsely populated and densified population. Statistically, the sparsely populated areas in the data space indicate that the probability of the data occurring in this area is low, so that the data falling within these areas can be considered abnormal. In 2012, they optimized the iForest algorithm for isolated forests [14], which takes advantage of subsampling to achieve lower linear time complexity and smaller memory requirements, and to effectively handle swamping and masking effects. Empirical assessment shows that iForest has better processing time on AUC than ORCA, SVR, LOF and random forest, and has good robustness. iForest also works well in high-dimensional problems with many irrelevant attributes and in the absence of anomalies in training samples.

Time series data have typical contextual features. Some researchers use statistical features and machine learning methods for anomaly detection. In 2014, Vallis proposed a novel sequence anomaly detection technique [23], which mainly used statistical learning to detect anomalies. The data used included the measurement of application and system operation log. In addition, the algorithm uses robust

statistical measures, that is, median, median absolute deviation (MAD), and underlying segment approximation long-term trends to accurately detect anomalies. In 2015, Yahoo's Laptev used a variety of statistical features (trend, seasonal, spectral entropy, etc.) of time series data and then clustered them into clusters C based on these features. After clustering, they use the devia-tion between the centroids to detect the intercluster anomalies, and the deviation of the centroid in the cluster and the time series i to detect intraframe anomalies [11]. In the meantime, Yahoo has implemented an anomaly detection framework EGADS. Compared with other anomaly detection systems for different time series features of real-time and composite data, the EGADS framework improves accuracy and recall by $50\% - 60\%$.

In addition, the time series data has the features of large flow velocity, high dynamic, high dimensionality and complexity. Some researchers have studied how to improve the streaming data algorithm for anomaly detection of time series data and further improve the detection efficiency.

Time series data can be converted to symbolic representations, and such representations will potentially allow researchers to utilize algorithms from text processing. In 2003, Lin *et al.* proposed a time series notation [12] that is suitable for the streaming data algorithm. This notation is unique. It allows dimensionality/numerosity reduction. It also allows to define the distance measure method of the representation, reducing the limits of distance measure method on the original sequence. The latter feature allows running some symbolic data mining algorithms while producing the same result as the algorithm that manipulates the original data. Finally, their notation representation method could transform real and valuable data into streams with minimal time and space overhead.

In 2016, a subspace histogram technique, referred to as RS-Hash, has been proposed in [16]. The extension to the streaming setting is referred to as RSStream. This technique averages the log-density in grid regions of varying sizes and shapes in order to provide the final score. The approach uses randomized hashing for efficiency and requires linear time. This approach is suitable for sub-space outlier detection in high-dimensional data and therefore its efficiency is notable.

Some researchers have improved the isolated forest algorithm to accommodate the anomaly detection of streaming data. In 2011, Tan *et al.* proposed a stream-based half-space tree (HS-Trees) [22], a fast single-class anomaly detector for ever-evolving data streams. It only requires normal data for training and works well with very few exceptions. The model is characterized by Ensemble's random HS-Trees. Because it does not require refactoring models to accommodate evolving data streams, it is very efficient. In 2016, Guha *et al.* proposed an exceedingly simple subspace outlier detection algorithm [9], which can be implemented in a few lines of code, and whose complexity is linear in the size of the dataset and the space requirement is constant. The approach uses randomized hashing to score data points and has a neat subspace interpretation. Furthermore, the approach can be easily generalized to data streams. We present experimental results showing the effectiveness of the approach over other state-of-the-art methods. In 2014, the algorithm [24] proposed by Wu *et al.* is a fast and accurate density estimator implemented by multiple fully randomized space trees (RS-Trees), named RS-Forest. The piecewise constant density estimate of each RS-tree is defined on the tree node into which an instance falls. Each incoming instance in a data stream is scored by the density estimates averaged over all trees in the forest. Two strategies, statistical attribute range estimation of high probability guarantee and dual node profiles for rapid model update, are seamlessly integrated into RS-Forest to systematically address the ever-evolving nature of data streams.

In order to deal with the problem of stream data anomaly detection more efficiently, some researchers use distributed computing framework to improve time efficiency. In 2012, Zaharia *et al.* proposed a new programming model, D-Streams [30], which provides a high-level programming API, strong consistency, and efficient fault recovery. D-Streams supports a new efficiency-enhancing recovery mechanism that goes beyond traditional replication and upstream backup solutions in streaming databases: recovering lost state in parallel across the entire cluster. We have prototyped D-Streams, which is an extension of the Spark cluster computing framework, which we call Spark Streaming. It allows users to seamlessly mix stream processing, batch processing, and interactive queries. In 2014, Solaimani *et al.* used Apache Spark to process streaming service performance data (detecting anomalies in VMware-based multi-data source cloud data centers) [18] and quickly detected anomalies. They compared the average processing delays of tuples between Spark and Storm during clustering and prediction. It was found through experiments that Spark is much faster than Storm. In same year, in the paper [17], Solaimani *et al.* have taken up the challenge of anomaly detection in VMware based cloud data centers. We have employed a Chisquare based statistical anomaly detection technique in Spark.

Security and privacy issues of mobile cloud including several fields: Identity and Access Management, Mobile Network Security Vulnerabilities, Trust Management, Privacy Management and Data Protection, Encryption and Key Management, Risk Management, Physical Security,Malware,Intrusion Detection and Prevention [21]. Many researchers apply anomaly detection algorithms into many security fields [10], [15]. For example, in the field of intrusion detection, by detecting the abnormal pattern of network traffic, we can infer whether there is intrusion behavior. Especially, the latency brought by unstable wireless networks and computation failures caused by constrained resources limit the development of mobile computing. Some researchers focus on how to improve the computing power of IoT devices [7], [31]. Other researchers focus on cloud services to transfer IoT devices computation to backend cloud servers. Anomaly detection in mobile service security not only help us find network intrusions, but also could help us

find the energy consumption anomaly for mobile devices, [6] proposed a service composition method to from an energy consumption perspective; find the mobility anomaly of mobile devices [4] proposed an mobility-aware method for mobile service selection; and find the risk of mobile services, [5] proposed a way to reduce risk of service composition.

III. DESIGN OVERVIEW

Numerous methods have been proposed in the literature in order to determine significant change points in a multidimensional data stream. Many change point detection techniques have been studied independently in the database literature and the outlier detection literature. The reality is that these two areas are too closely related to be treated separately. The sudden changes in aggregate local and global trends in the underlying data are often indicative of anomalous events in the data. Our methods are essentially transforming collective anomaly detection into point anomaly detection.

A. ISOLATION FOREST AND ITS VARIANTS FOR STREAMING ANOMALY DETECTION

In the probability statistics based anomaly detection algorithms, the algorithm often considers the statistical features of the time series (e.g. distribution, deviation, etc.). In some specific application domain, the anomaly is just judged based on the experience threshold (e.g. 3sigma threshold in Gauss distribution model) and the local fluctuation points are easily identified as abnormal points. However, in machine learning based clustering and other anomaly detection algorithms, the contextual statistical features (such as trend and seasonality) of the time series data are not taken into consideration. It is easy to miss the contextual information. At the same time, time series data has the features of high dimensionality, complexity and high noise. Thus, it is very difficult to label the training data, it is more suitable to use unsupervised algorithm.

Liu *et al.* proposed a unique anomaly detection algorithm in 2008 - the Isolated Forest Algorithm [13] and improved this algorithm in 2012. Commonly, anomalous instances are those objects that their attributes values are very different from the normal instances and are easier to being divided than normal instances. [14]. Later, Ding and Fei implemented an algorithm for anomaly detection of streaming data based on the isolated forest algorithm and the sliding window technique [8]. Sun *et al.* proposed an algorithm based on isolated forest for user behavior anomaly detection [19] and applied it in the enterprise environment with good performance. iForest's detailed algorithms and assessments can be seen in [14], we only describe the subject here part of the iForest algorithm.

An iForest consists of multiple isolation trees, namely iTree, which are created by choosing attributes and the values of attributes randomly. At each node in the isolation trees, the samples are divided into two parts based on the selected attributes and its values. Here, the attributes are selected randomly and the split value for this selected attribute is chosen randomly as well between the minimum value and maximum value of this selected attribute. Commonly, anomalous instances are those objects that their attributes values are very different from the normal instances and are easier to being divided than normal instances. In the process of isolation, they are also closer to the root and more easily divided than the normal instances. In order to alleviate the effects imported by the random characteristic in the process of building the isolation forest, we calculate the average depth of the instance in the forest which is composed multiple isolation trees and use the average depth as the anomalous score (calculate by formula (1)) of the instance. The lower score the instance has, the higher probability it an anomaly. In the above algorithm, the procedure of creating the iTree is a key step. The detail creating procedure of iTree algorithm can be seen in [14].

Algorithm 1 *iForest*(X, T, N)

Input: input dataset X, number of trees T, subsampling size N

Output: a set of iTrees

1: Initialize $Forest = \{\}$
2: set iTree height h = ceiling$(log_2 N)$
3: **for** $i = 1$ to T **do**
4: $\quad X' \leftarrow sample(X, N)$
5: $\quad Forest \leftarrow Forest \cup iTree(X', 0, h)$
6: **end for**
7: **return** *Forest*

$$S(x, N) = 2^{\frac{E(h(x))}{c(N)}}$$
$$E(h(x)) = \frac{1}{L}\sum_{i=1}^{L} h_i(x) \quad (1)$$

Time series data naturally has the characteristics of high real-time streaming data, fast flow rate and large amount of data. Ordinary models are often unable to perform real-time and efficient detection, and the model training of the algorithm is fast enough. Reference [22] proposed a flow half-space tree based on isolation forest. The definition of the HS-tree with depth h is a complete binary tree with $2^{h+1} - 1$ nodes, all of which have the same depth, h. There are three main differences between the construction of HS-trees and isolated forests:

1) When constructing a tree, the algorithm selects a random selection dimension q in the workspace for node expansion, and uses the value p of the midpoint of q (the value of q is randomly selected in the isolated forest), and divides the workspace into two halves. Space, thus creating the child node on the left and the child node on the right. This node extension will continue until all leaf nodes reach the maximum depth h or the subspace is not separable.
2) Add mass information to the node of the tree to record the quality distribution of the data in the workspace of the node. The mass distribution contains the following information: (i) minimum and maximum values,

TABLE 1. Statistical features of time series.

Feature name	Feature Description
Periodicity (frequency)	Periodicity is very important for determining seasonality
Trend	If there is a long-term change in the average level, it exists
Seasonality	When the time series is affected by seasonal factors, such as one day a month in a year or in a week
Mean	The average value
Variance	Sequence volatility metric
Auto-correlation	Express long-range dependency
Skewness	Measurement of symmetry
Kurtosis	Measures if the data has reached a peak relative to the normal distribution
Hurst exponent	Long-term memory of time series
Lyapunov exponent	Measuring the divergence speed of the nearby trajectories

respectively storing each dimension of the workspace represented by the node; (ii) the variables r and l respectively record the mass distribution of the data stream captured in the reference window and the latest window; (iii) the variable h records the depth of the current node; (iv) Left and Right respectively Represents the left and right child nodes of the current node, and each child node is associated with a half space.

3) In the definition of anomaly score, the isolated forest only considers the depth of the node, while the HS-Tree comprehensively considers the depth and the mass information of the node.

B. STATISTICAL FEATURES

The key difference between time series and other domains anomaly detection is that the time series has temporal continuity, and anomaly in time series is often defined as change on temporal continuity. Thus, we fully consider the temporal continuity by computing statistical features of time series and then leveraging them into the following methods.

According to the Table(1), the calculating process of statistical features are as follows:

1) TREND AND SEASONALITY

The trend is a long-term change in the time series, which is a major component of the time series. Seasonality is the periodic nature of a time series, which indicates how the time series changes periodically. To get the trend and seasonality of time series data, you need to decompose the time series data.

STL ('Seasonal and Trend decomposition using Loess') is a time series decomposition method using robust local weighted regression as a smoothing method. Loess (locally weighted scatterplot smoothing, LOWESS or LOESS) is a local polynomial regression fitting, which is a common method for smoothing two-dimensional scatter plots. It combines the simplicity of traditional linear regression with the flexibility of nonlinear regression. When estimating the value of a response variable, first take a subset of the data from its vicinity, and then perform linear regression or quadratic regression on the subset. The regression uses a weighted least squares method, that is, the closer to the estimated point, the greater the weight. The final use of the obtained local regression model to estimate the value of the response variable. In this way, the point-by-point operation is performed to obtain the entire fitting curve.

After leveraging STL method to obtain the trend and seasonality of the time series, calculate the distances of the trend and seasonal curves from the original curve are calculated as trend and seasonal features respectively.

2) MEAN AND VARIANCE

Mean,where the arithmetic mean is the most common, calculated as:

$$mean = \frac{\sum_{i=1}^{N} X_i}{N} \tag{2}$$

Varaiance,in the statistical description, the variance is used to calculate the difference between each variable (observation) and the population mean. In order to avoid the sum of the mean deviations being zero, the sum of the squares of the mean differences and the content of the samples, the statistics use the sum of squared mean deviations to describe the degree of variation of the variables. The formula for calculating the variance of the population:

$$\sigma^2 = \frac{\sum(X-\mu)^2}{N} \tag{3}$$

3) SKEWNESS AND KURTOSIS

Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive or negative, or undefined.

$$Skew(X) = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] \tag{4}$$

kurtosis is a measure of the ''tailedness'' of the probability distribution of a real-valued random variable. In a similar way to the concept of skewness, kurtosis is a descriptor of the shape of a probability distribution and, just as for skewness, there are different ways of quantifying it for a theoretical

distribution and corresponding ways of estimating it from a sample from a population. Depending on the particular measure of kurtosis that is used, there are various interpretations of kurtosis, and of how particular measures should be interpreted.

$$Kurt(X) = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \frac{E[(X-\mu)^4]}{\left(E[(X-\mu)^2]\right)^2} \quad (5)$$

C. COLLECTIVE ANOMALY DETECTION FOR MULTIDIMENSIONAL STREAMS

So much related works show that the isolation forest algorithm has a very good performance in the field of anomaly detection. iForest is an excellent anomaly detection method and is suitable for continuous value data. Mobile service metrics data are typical temporal data. Thus, we choose iForest as the fundamental algorithm and change it to apply in multidimensional streams. Considering statistical features and streaming processing limitation, this paper proposed a collective anomaly detection algorithm for multidimensional streams based on isolation forest. The following table describes the key symbols:

S	multidimensional streams
n	the number of samples in an iTreeFS
T	a isolation-based tree, iTreeFS
$Node$	a node in an iTreeFS
h	the current depth of a node
t	the number of iTreeFS in an ensemble iForestFS
$maxDepth$	maximum depth (level) of a tree
$mass$	quality distribution of a node in an iTreeFS
w	sliding window size

Algorithm 2 *iTreeFS*(X)

Input: input data X
Output: an iTreeFS
1: **if** X cannot be divided **then**
2: **return** exNodeSize $\leftarrow |X|$
3: **else**
4: let Q be a list of attributes in X
5: randomly select an attribute $q \in Q$
6: let $p = (minq + maxq)/2$
7: $X_l \leftarrow filter(X, q < p)$
8: $X_r \leftarrow filter(X, q \geq p)$
9: **return** $inNode\{Left \leftarrow iTree(X_l), Right \leftarrow iTree(X_r), SplitAtt \leftarrow q, SplitValue \leftarrow p, mass\}$
10: **end if**

Algorithm 2 illustrates the construction process of an iTreeFS, and the algorithm outputs an isolation tree iTreeFS. The process of the algorithm is to randomly select an attribute q in the attribute set Q of the input data X, and then calculate the average value p of the maximum value and the minimum value of q, and divide X into Xl and Xr according to q and p. Loop through the process until X cannot be divided or the node reaches the maximum depth of the tree. It is worth noting that each node on the iTreeFS tree has an important attribute mass, which stores the number of streaming samples flowing through the node, representing the quality distribution of the space in which the node is represented. In general, the more samples a space has, the more likely it is to be normal, and on the contrary it is more likely abnormal.

Algorithm 3 *UpdateMass*$(x, Node)$

Input: an instance x, a node in an iTreeFS $Node$
Output: none
1: $Node.mass++$
2: **if** $Node.h < maxDepth$ **then**
3: Let $Node\_$ be the next level of $Node$ that x traverses
4: $UpdateMass(x, Node\_)$
5: **end if**

Algorithm 3 describes how to update the mass attribute (distribution information of a node). The quality distribution Node.mass++ of each node on the path of the final leaf node Node* is traversed from the root node of the tree when detecting whether the continuously flowing instance x is abnormal. By doing so, the probability that the instance to be tested falls within the subspace where the node is located can be counted. The smaller the Node.mass is, the smaller the anomalous probability is, and on the contrary the possibility of anomaly is greater.

Algorithm 4 *iForestFS*(S, t, n, w)

Input: multidimensional streaming data S, number of trees in each iForestFS t, subsampling size n, sliding window size w
Output: a iForestFS
1: Split S into $\{S_1, S_2, \cdots, S_N\}$ with sliding window size w or same time interval, initialize $X = \{\}$;
2: **for** $i = 1$ to N **do**
3: Calculate statistical features (Table(1)) for each dimension of S_i, named X_i
4: add X_i to X
5: **end for**
6: **return** $iForestFS =$ invoke $iForestFS(X, t, n)$

Algorithm 4 illustrates the iForestFS's construction process for multidimensional streams. The process of the algorithm is to decompose the stream data S of dimension m into subsequences Si according to window w or same time interval. For each subsequence Si, calculating the specific statistical features of Si and obtaining the feature vector Xi refer to Table 1. Using Xi as input data, calling the streaming isolation forest algorithm iForestFS. In order to enable the algorithm to detect collective anomaly of streaming data, it is necessary to initialize an isolation forest iForestFS with algorithm 4 using a small number of samples before the multidimensional streaming data arrives.

1) ABNORMAL SCORE

Assuming that Si is a subsequence that has been divided from stream S by the same time interval or same sliding window

TABLE 2. Distributed computing framework for stream processing.

Framework	storm	trident	spark streaming	samza	flink
Streaming model	native	micro-batching	micro-batching	native	native
Api	compositional	compositional	declarative	compositional	declarative
Guarantees	at-least-once	exactly-once	exactly-once	at-least-once	exactly-once
Fault-tolerance	record acks	record acks	rdd based checkpointing	log-based	checkpointing
State management	not built-in	delicate operators	delicate D-Stream	stateful operators	stateful operators
Latency	very low	medium	medium	low	low
Throughput	low	medium	high	high	high
Maturity	high	high	high	medium	low

size w, namely S1, S2, SN are composed of stream S. First, calculate the score of Si on each tree in iForestFS, and the formula is as follow:

$$Score(S_i, T_j) = Node^*.mass * 2^h \tag{6}$$

where Si is the subsequence to be tested, Tj is a tree in the forest, and $Node^*$ indicates the leaf node that Si traverses in the isolated tree iTreeFS Tj, and $Node^*$.mass indicates the quality distribution information of the space where the node is located(namely how many samples have fallen into this space before), h represents the height of this node. Then, the anomaly score of Si is obtained by summing the abnormal scores calculated for each tree in the forest:

$$Score(S_i) = \frac{\sum_{j=1}^{t} Score(S_i, T_j)}{t} \tag{7}$$

where t is the number of trees in the forest iForestFS. Score(Si) is the abnormal score of Si. It should be noted that in the prediction stage, with the arrival of streaming data, the sliding window technology is used to segment and obtain Si. When calculating the abnormal score of Si, it is necessary to update the quality information mass of the tree node, indicating the number of instances fallen into the node representation. In the space, algorithm 3 is used for updating the quality information mass.

The basic method of determining whether a test sample Si is outlier:

1) If the outlier score of a sample is very close to 1, the sample can be defined as an abnormal instance;
2) If the outlier score of a sample is much less than 0.5, it can be regarded as a normal instance very safely;
3) If the outlier scores of all samples are close to 0.5, then the entire sample set can be considered to have no obvious abnormal instances.

D. DISTRIBUTED IFORESTFS

When dealing with streaming data, some researchers use a mature distributed computing framework to process streams. Since different distributed stream computing engines have different system architectures, usage scenarios, and processing performance, such as spark streaming [30] can support a variety of data sources, kafka streams can process events one by one under millisecond delay. The apache storm has a high throughput.

The multidimensional stream data anomaly detection algorithm has very high time efficiency requirements. In order to improve the operation efficiency of the algorithm, the algorithm is improved into a distributed algorithm that can perform anomaly detection on a plurality of machines simultaneously. The specific improvement method is to first change all the key processes of the algorithm to parallel computing, and then select the appropriate mature distributed computing framework, and then design and implement the algorithm through the API provided by the framework.

We compare many distributed compute frameworks (Table(2)), and consider the performance, guarantees, and fault-tolerance. Spark Streaming is a computing framework with high maturity and throughput, and the latency is also medium. Meanwhile, Spark has a better ecology in the open source community and Spark MLlib could provide some useful tools for us to analyze our methods. So we finally choose spark-streaming as the distributed compute framework.

As shown in Figure(1), the design overview of iForestFS and distributed iForestFS can be divided into two parts: data source and anomaly detection. The data source part is the input data: outside environment, system resource, server performance, and application metrics. When input data arrives, they should be cleaned and preprocessed first. The anomaly detection part is the main idea of iForestFS. We take full advantage of the temporal continuity of time series and redesign iForest using Spark Streaming in order to improve time performance and scalability of iForestFS.

IV. EXPERIMENT AND EVALUATION

In order to verify the proposed methods, we have conducted extensive experiments using the three real datasets of the UCI University Machine Learning Library to evaluate the collective anomaly detection algorithm iForestFS based on iForest and statistical features in mobile service security domain. The KDDCUP99 database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment. In the KDDCUP99 dataset, the intrusion attacks fall into four main categories:DOS: denial-of-service, e.g. syn flood;R2L: unauthorized access from a remote machine, e.g. guessing password; U2R: unauthorized access to local superuser (root) privileges, e.g., various ‘‘buffer overflow’’ attacks;probing: surveillance and other probing, e.g., port scanning. When using the KDDCUP99 dataset, iForestFS could be trained

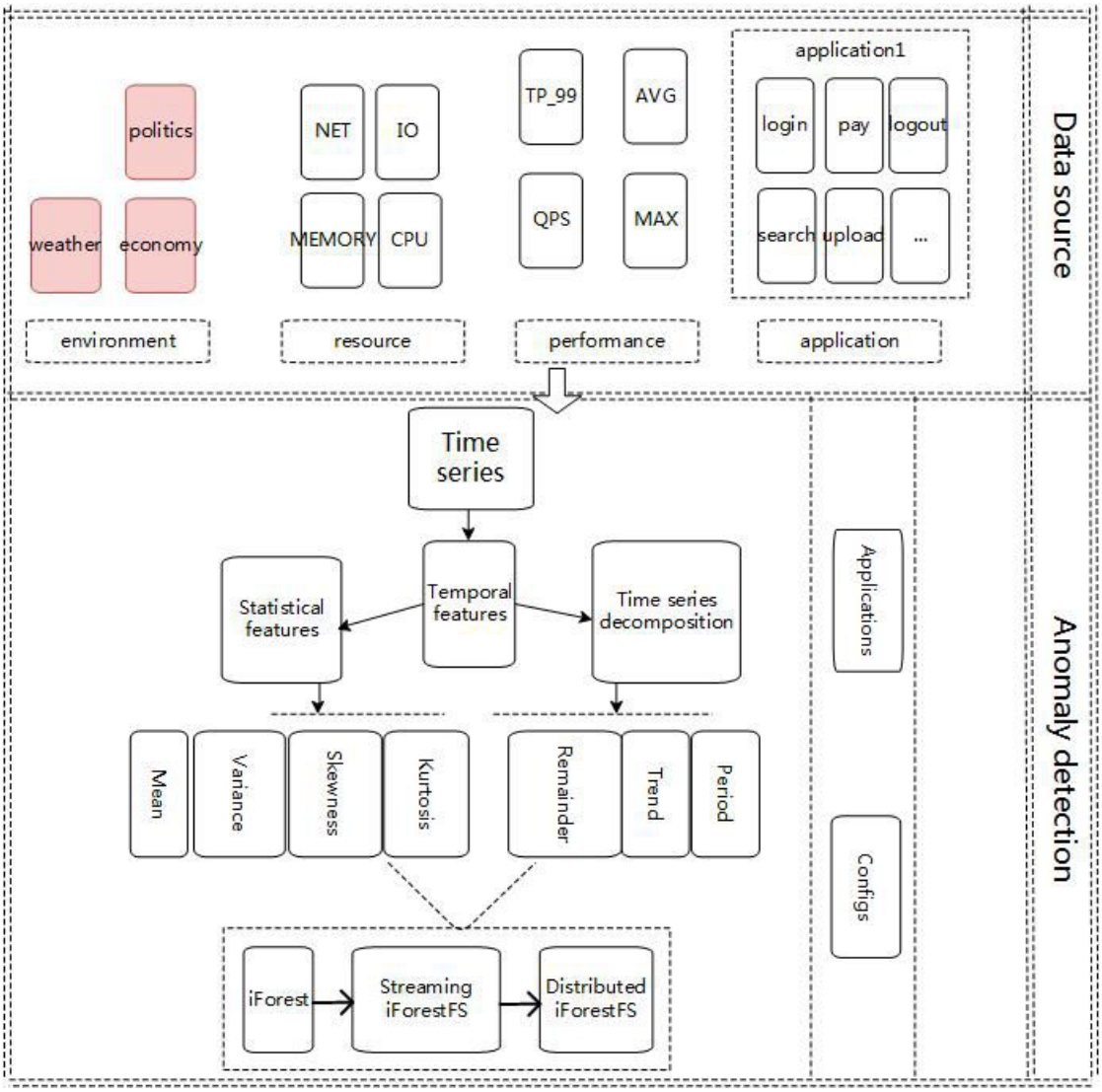

FIGURE 1. Design overview of iForestFS and distributed iForestFS.

as a network intrusion detector. Because the iForestFS algorithm is an unsupervised method, the category attribute is not needed in the exception process. The label of the abnormal data is only used to evaluate the final anomaly detection performance.

Our experiment was run on a server equipped with Intel Xeon E5-2620 @2.40GHZ (2 processors), 64GB RAM, and centos7 operating system. The algorithm described in Section 3 is programmed using the Java language on the INTELLIJ IDEA integrated development environment and uses the machine learning library WEKA(WEKA site is https://www.cs.waikato.ac.nz/ml/weka/) to process and analyze the results of the experiments.

A. DATASETS

Table(3) shows the details of the three datasets selected because they contain known data using the anomaly class as a basic fact and these datasets have been used many times in the anomaly detection algorithm literature to evaluate algorithm performance [14], [22].

TABLE 3. Information of four datasets.

Dataset Name	M(#instances)	N(#attributes)	Anomaly Threshold
Http	567,498	22	0.4%
Smtp	95,156	22	0.03%
Shuttle	49,097	9	7.15%

The Http dataset and the Smtp dataset are two subsets of the KDD-CUP99 network intrusion data. The KDD-CUP99 dataset contains 4,898,431 instances, each of which has 41 attributes (duration, service, srcBytes, dstBytes etc.) and a class label. In our experiment, we only selected the numerical attributes (22 in total). The Shuttle dataset contains 9 attributes. Approximately 80% of the data belongs to class 1

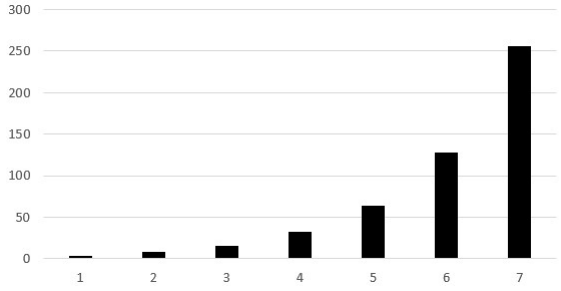

FIGURE 2. The size of sliding window (y-axis) versus the i-th experiments (x-axis).

and we regard labeled 2, 3, 5, 6, 7 as the anomaly class. In each dataset, the class label attribute is used only to evaluate the algorithm.

Because the iForestFS algorithm only processes numerical attributes, all nouns and bool attributes are removed. The attributes selected in our experiments are continuous values. It should be noted that the proposed method is for collective anomaly detection in multidimensional streams but the dataset aforementioned is used for anomalous point detection. Thus, we do a lot of work to transform the dataset. First, we spilt the dataset into sequences by window size w, which is a parameter of the experiment. Second, we label the sequences using the origin label of the point in the dataset, when the anomalous points account for up to 50% of the sequence, this sequence is considered to be an abnormal sequence, otherwise the sequence is normal. Finally, we detect whether the sequence is anomalous, namely detect collective anomaly.

B. EXPERIMENT SETTINGS AND RESULT EVALUATION

1) IFORESTFS

After the dataset is built, the primary task is to train the anomaly detector algorithm iForestFS to perform the collective anomaly detection in the sliding window frame. The iForestFS anomaly detection method has several important parameters, namely the size of the sliding window and the scale of ensemble learning. For the former, because the size of the sliding window w is time-sensitive, there is no parameter setting method for the theoretical direction. In our experiment, as shown in Figure(2), the sliding window value ranges from 4 to 256.

For the latter, the number of iTreeFS in iForestFS and the number of samples_size used to build iTree are included. For the itree_num value, there is actually no theoretical direction for ensemble learning. Liu *et al.* [14] found that the path length generally converges well when itree_num $=$ 100. So, in our experiment, we use itree_num $=$ 100 and sample_size $=$ 256 as defaults.

In anomaly detection domain, AUC (Area Under ROC) [13], [22] is commonly used to measure the overall performance of an anomaly detector, regardless of the threshold between true positive and true negative. Abnormal points are always considered positive during the abnormal process, and normal points are always detected as negative. To evaluate the performance of our proposed iForestFS anomaly detection algorithm, we used AUC on four datasets. The test results can be seen in Table(4) and Figure(3).

TABLE 4. AUC of different Datasets for different sliding windows size.

Sliding Window size	Dataset		
	Http	Smtp	Shuttle
4	0.938	0.962	0.956
8	0.931	0.969	0.813
16	0.947	0.974	0.738
32	0.958	0.982	0.097
64	0.976	0.989	0.021
128	0.987	0.991	-
256	0.991	0.995	-

From the experimental results shown in Figure 2(the value of x axis is in log scale), we can easily find out when the size of sliding window increases, the value of AUC increases correspondingly. When the size of the sliding window increases to the desired value, for example 256 for the Http dataset, iForestFS detection performance is stable and reliable, that is, there is no need to further increase the sliding window size. Because it increases processing time and memory size while there is no gain in detection performance. Besides it, if the sliding window size is too big, the statistical features may lost much infomation of the origin data. For Shuttle dataset, the result is really bad and the AUC cannot be calculated when k $>$ 64. This is because that Shuttle consist of discrete samples while KDDCUP99 are made up of contextual points with temporal continuity. The experimental results show that iForestFS is only suitable for multidimensional streams collective anomaly detection and the experiment on Shuttle dataset shows that our method is not suitable for discrete sequences.

As mentioned above, iForestFS is unsupervised. We could just judge whether an instance is an anomaly by its anomaly score. In mobile service security domain, high anomaly score just represents that the metrics of the system is abnormal, we need experienced system operators to furthermore analyze current system situation and find the deep reason.

2) DISTRIBUTED IFORESTFS

The datasets and hardware setting is same to last experiment. In this experiment, we use spark as the distributed compute framework. We mainly test the time performance and scalability of the distributed iForestFS.

Table(5) shows the time consuming between iForestFS and distributed iForestFS using spark-streaming framework. Here we use the above largest dataset Http for testing. The parameters of iForestFS model: slidingWindowSize $=$ 256, numTrees $=$ 100, maxSamples $=$ 256. We did not consider the training time because the model was built before the streaming data arrived. The prediction time is the sum of the prediction time of the sample in whole dataset. From the result of the experiment, the distributed iForestFS runs nearly 3.7 time faster in prediction phase.

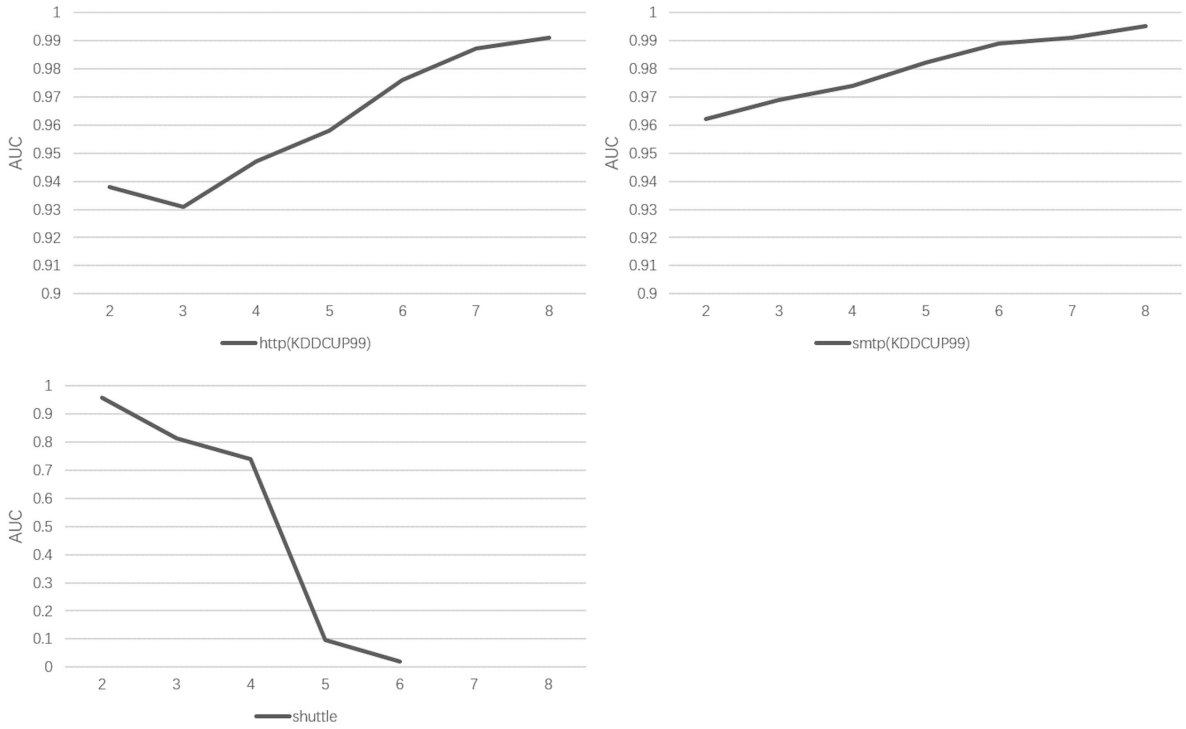

FIGURE 3. AUC performance (y-axis) versus the different sliding window size in log scale (x-axis).

TABLE 5. Time performance.

time cost (s)	**iForestFS**	**spark** (4 cores)
prediction	4.8	1.3

TABLE 6. Scalability performance.

time cost (s)	**1 core**	**2 cores**	**3 cores**	**4 cores**
prediction	4.6	3.1	2.4	1.3

Table(6) shows the scalability of the distributed iForest model. The testing dataset is still Http. The memory is set 1G per executor on Spark. The number of cores range from 1 to 4 cores. Model parameters of the distributed iForestFS: slidingWindowSize $=$ 256, numTrees $=$ 100, maxSamples $=$ 256. The calculation method of prediction time is same as above. From the result of the experiment, the distributed iForestFS shows good scalability, where prediction time decreases while the cores increase.

V. CONCLUSION AND FUTURE WORK

This paper considers the statistical features of time series data, based on iForest anomaly detection framework, and uses distributed compute framework named spark-streaming to implement a collective anomaly detection algorithm for multidimensional streams in mobile service security, namely iForestFS. Experiments were conducted on three real datasets and the results demonstrated that the proposed method was effective and efficient. Especially for the dataset KDDCUP99, iForestFS is trained as a network intrusion detector which has a good performance. When using different features about mobile cloud services' metrics suggested by domain knowledge, iForestFS could detect different kinds of anomalies for mobile service security. However, in our study, we only verified that our proposed method was suitable for collective anomaly detection for multidimensional streams but did not compare with existing methods. More importantly, there are still some problems that need further discussion.

Our future work will focus on the following three directions: First, the size of the sliding window is fixed in this article, which may not be suitable for real applications. Second, we did not evaluate the influence of a single statistical feature on our anomaly detection effect. In actual applications, different data may have better detection effects by using different statistical features. Third, as streams flow, maybe dynamic reconstruction of the iTreeFS even iForestFS could lead to better performance.

REFERENCES

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

[2] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection for discrete sequences: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 5, pp. 823–839, May 2012.

[3] N. K. Chaubey and D. M. Tank, "Security, privacy and challenges in mobile cloud computing (MCC). A critical study and comparison," *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 4, no. 2, pp. 1259–1266, Feb. 2016.

[4] S. Deng, S. Huang, D. Hu, J. L. Zhao, and Z. Wu, "Mobility-enabled service selection for composite services," *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 394–407, May/Jun. 2016.
[5] S. Deng *et al.*, "Toward risk reduction for mobile service composition," *IEEE Trans. Cybern.*, vol. 46, no. 8, pp. 1807–1816, Aug. 2016.
[6] S. Deng, H. Wu, W. Tan, Z. Xiang, and Z. Wu, "Mobile service selection for composition: An energy consumption perspective," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 3, pp. 1478–1490, Jul. 2017.
[7] S. Deng, Z. Xiang, J. Yin, J. Taheri, and A. Y. Zomaya, "Composition-driven IoT service provisioning in distributed edges," *IEEE Access*, vol. 6, pp. 54258–54269, 2018.
[8] Z. Ding and M. Fei, "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window," *IFAC Proc. Vol.*, vol. 46, no. 20, pp. 12–17, 2013.
[9] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, "Robust random cut forest based anomaly detection on streams," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2016, pp. 2712–2721.
[10] M. S. Islam and S. A. Rahman, "Anomaly intrusion detection system in wireless sensor networks: Security threats and existing approaches," *Int. J. Adv. Sci. Technol.*, vol. 36, no. 1, pp. 1–8, Nov. 2011.
[11] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2015, pp. 1939–1947.
[12] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proc. 8th ACM SIGMOD Workshop Res. Issues Data Mining Knowl. Discovery*, Jul. 2003, pp. 2–11.
[13] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 413–422.
[14] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discovery Data*, vol. 6, no. 1, Mar. 2012, Art. no. 3.
[15] A. Sari, "A review of anomaly detection systems in cloud networks and survey of cloud security measures in cloud storage applications," *J. Inf. Secur.*, vol. 6, no. 02, p. 142, Mar. 2015.
[16] S. Sathe and C. C. Aggarwal, "Subspace outlier detection in linear time with randomized hashing," in *Proc. ICDM*, Dec. 2016, pp. 459–468.
[17] M. Solaimani, M. Iftekhar, L. Khan, and B. Thuraisingham, "Statistical technique for Online anomaly detection using spark over heterogeneous data from multi-source VMware performance data." in *Proc. IEEE Int. Conf. Big Data*, Oct. 2014, pp. 1086–1094.
[18] M. Solaimani, M. Iftekhar, L. Khan, B. Thuraisingham, and J. B. Ingram, "Spark-based anomaly detection over multi-source vmware performance data in real-time," in Proc. Symp. Comput. Intell. Cyber Secur. (CICS), Dec. 2014, pp. 1–8.
[19] L. Sun, S. Versteeg, S. Boztas, and A. Rao. (2016). "Detecting anomalous user behavior using an extended isolation forest algorithm: An enterprise case study." [Online]. Available: https://arxiv.org/abs/1609.06676
[20] H. Suo, Z. Liu, J. Wan, and K. Zhou, "Security and privacy in mobile cloud computing," in *Proc. 9th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jul. 2013, pp. 655–659.
[21] H. Takabi, S. T. Zargar, J. B. Joshi, D. B. Rawat, B. B. Bista, and G. Yan, "Mobile cloud computing and its security, privacy and trust management challenges," in *Security, Privacy, Trust, Resource Management Mobile and Wireless Communications*, D. B Rawat, B. B Bista, and G. Yan, Eds. Pennsylvania, PA, USD: IGI Global, 2014. doi: 10.4018/978-1-4666-4691.
[22] S. C. Tan, K. M. Ting, and T. F. Liu, "Fast anomaly detection for streaming data," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)* , vol. 22, 2011, p. 1511.
[23] O. Vallis, J. Hochenbaum, and A. Kejariwal, "A novel technique for long-term anomaly detection in the cloud," in *Proc. 6th USENIX Workshop Hot Topics Cloud Comput. (HotCloud)*, Philadelphia, PA, USA, 2014. [Online]. Available: https://www.usenix.org/conference/hotcloud14/workshop-program/presentation/vallis
[24] K. Wu, K. Zhang, W. Fan, A. Edwards, and S. Y. Philip, "RS-forest: A rapid density estimator for streaming anomaly detection," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2014, pp. 600–609.
[25] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proc. 23rd Int. Conf. Mach. Learn.*, Jun. 2006, pp. 1033–1040.
[26] Q. Yang and W. U. Xindong, "10 challenging problems in data mining research," *Int. J. Inf. Technol. Decis. Making*, vol. 5, no. 4, pp. 597–604, Dec. 2006.
[27] Y. Yin, L. Chen, Y. Xu, and J. Wan, "Location-aware service recommendation with enhanced probabilistic matrix factorization," *IEEE Access*, vol. 6, pp. 62815–62825, 2018.
[28] Y. Yin, Y. Xu, W. Xu, M. Gao, L. Yu, and Y. Pei, "Collaborative service selection via ensemble learning in mixed mobile network environments," *Entropy*, vol. 19, no. 7, p. 358, Jul. 2017.
[29] Y. Yin, F. Yu, Y. Xu, L. Yu, and J. Mu, "Network location-aware service recommendation with random walk in cyber-physical systems," *Sensors*, vol. 17, no. 9, p. 2059, Sep. 2017.
[30] M. Zaharia, T. Das, H. Li, S. Shenker, and I. Stoica, "Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters," *HotCloud*, vol. 12, p. 10, Jun. 2012.
[31] C. Zhang, H. Zhao, and S. Deng, "A density-based offloading strategy for IoT devices in edge computing systems," *IEEE Access*, vol. 6, pp. 73520–73530, 2018.
[32] X. Zhang, P. Fan, and Z. Zhu, "A new anomaly detection method based on hierarchical HMM," in *Proc. 4th Int. Conf. Parallel Distrib. Comput., Appl. Technol.*, Aug. 2003, pp. 249–252.

YU WENG received the Ph.D. degree in computer science from the University of Science and Technology, Beijing, China, in 2010. He is currently an Associate Professor of computer science with the Information Engineering Department, Minzu University of China. He has published more than 20 conference and journal papers. His current research interests include machine learning, cloud computing, and distributed computing.

LEI LIU is currently pursuing the master's degree in computer science and technology with the College of Information Engineering, Minzu University of China. His current research interests include message-oriented middleware and anomaly detection.