Multivariate QoS Monitoring in Mobile Edge Computing based on Bayesian Classifier and Rough Set

Pengcheng Zhang*, Yaling Zhang*, Hai Dong† and Huiying Jin*
*College of Computer and Information, Hohai University, Nanjing, China, 211100
†School of Science, Royal Melbourne Institute of Technology, Melbourne, Australia
Email: pchzhang@hhu.edu.cn, 2460154274@qq.com, hai.dong@rmit.edu.au, 367046895@qq.com

***Abstract*—Mobile edge computing transfers computing and storage from traditional cloud servers to edge servers, presenting new challenges to quality assurance of edge services. Quality of Service (QoS) is considered as a defacto standard to evaluate similar services with different quality. Given the fact that QoS values are highly dynamic in complex edge environments, QoS monitoring is viewed as a promising technique to comprehensively and effectively understand QoS status of edge services. Due to the distributed storage of historical QoS data and the changeable edge environments, traditional QoS monitoring approaches cannot be directly applied into mobile edge computing. To address this problem, this paper proposes a novel multivariate QoS monitoring approach, called Rs-mBSRM (multivariate BayeSian Runtime Monitoring using Rough set). First, the weights of different QoS attributes are quantified and obtained according to the historical samples based on rough set theory. Second, a Bayesian classifier is constructed for each corresponding edge server during the training stage. Finally, during the monitoring stage, considering the distributed data storage, the classifier is dynamically switched and the attribute weights are also updated due to user mobility. Our experimental results on public data sets show that Rs-mBSRM is better than existing QoS monitoring approaches and is more suitable for mobile edge computing.**

***Keywords*-Mobile edge computing; Quality of Service; Monitoring; Bayesian classifier; Rough set**

I. Introduction

Mobile edge computing [1] emerges as a novel computing paradigm, in which multiple decentralized nodes can process and store data close to data sources. In this way, the computing and transmission stress on traditional centralized cloud can be offloaded to decentralized edge servers. There is no doubt that deploying services in edge servers will provide users with better experiences, such as *faster response times*, *greater throughput*, and *higher reliability*. These non-functional attributes of services are also called qualities of service (QoS) [2]. For service providers, QoS can ensure the quality of service operations by timely updating service quality status and replacing failed/unsatisfied services. Service consumers are concerned with how to choose services of high quality from many services with similar functions.

Service monitoring is one of the promising techniques to obtain the current state of edge services [3]. Some traditional monitoring approaches [4], [5] deploy monitoring systems to obtain QoS values for evaluation. However, these approaches are not only costly, but also limit themselves in the determined QoS values. In fact, the QoS values of a service might change rapidly in a dynamic network environment. Therefore, on the one hand, it is not appropriate for deciding whether QoS requirements are fulfilled based on the determined values. On the other hand, the requirements given by service consumers are often very vague. For example, a user may request a service to be normal for most of its usage period. Consequently, researchers have proposed novel approaches called probabilistic quality attributes [6], which is used to describe the fuzzy requirements of QoS attributes. For example, the probability that the service's response time is within 1s is greater than 85%. Based on probabilistic quality attributes, Chan et al. [7] first defined probabilistic standard of non-functional attributes using Probabilistic Computational Tree Logic (PCTL) [8]. However, the error rate of this approach is relatively high. The other researchers [9], [10], [11], [12] proposed several QoS monitoring approaches based on hypothesis testing. The main idea is to judge Whether or not the hypothesis is true according to existing conditions. In recent years, with the development of machine learning, Bayesian theory has been applied for QoS monitoring [13], [14], [15], [16]. The main feature of this idea is that the probabilistic status of current events can be inferred by training historical data. Zhang et al. proposed wBSRM [14] and IgS-wBSRM [15] to consider the impact of environmental factors and solve the timeliness problem of samples. However, these approaches only consider a single QoS attribute. In many cases, users may need to monitor more than one QoS attribute. For example, a service's response time meets user needs but its throughput does not. Consequently, whether or not the service meets user needs is still unclear. To comprehensively monitor multivariate QoS attributes of service, Zhang et al. proposed M-BSRM [16], which can weight multivariate QoS attribute values based on user preference and fuse the weighted QoS values using information fusion. In this way, multivariate QoS monitor can be achieved using Bayesian classifiers.

However, these approaches still contain the following two major gaps when multivariate QoS monitoring is performed

 P. Zhang, Y. Zhang, H. Dong and H. Jin, "Multivariate QoS Monitoring in Mobile Edge Computing based on Bayesian Classifier and Rough Set," 2020 IEEE International Conference on Web Services (ICWS), Beijing, China, 2020, pp. 189-196, doi: 10.1109/ICWS49710.2020.00032.

in mobile edge computing.

- *The distributed storage of data in the mobile edge environment makes the traditional monitoring approaches no longer applicable*. Instead of centralized data storage and computing in a traditional environment, most data storage and computing are performed in a highly distributed way, where each edge server is only responsible for data storage and computing requested for that server [1]. When a user calls a new edge server, the historical data in the old edge server is no longer valid for monitoring [17].
- *The dynamic network conditions in mobile edge environments make it impossible to objectively evaluate the effectiveness of different QoS attributes for a service based on the weights given by users' preferences*. For users who do not understand the concept of QoS, it is difficult to reasonably assign weights. In addition, in the mobile edge environment, QoS values vary remarkably in different edge servers [18]. In contrast, the weights assigned by users are usually static, which cannot adapt to such a changeable environment.

To fill in the gaps, this paper proposes a novel monitoring approach, namely, multivariate BayeSian Runtime Monitoring using Rough set (Rs-mBSRM), which can realize objective and effective multivariate QoS monitoring in mobile edge computing. The contributions of this paper are summarized as follows:

- *We propose a multivariate QoS monitoring approach for mobile edge computing*. During the training phase, the approach makes use of rough set theory to objectively evaluate the importance of multiple QoS attributes for each edge server, obtain the comprehensive weight values and construct the corresponding classifier. During the monitoring phase, when a user invokes a new edge server, to ensure the real-time monitoring performance, the classifier is dynamically switched, the attribute weights are updated, and the classifier for the new edge server is used for monitoring. When there is no historical data in the new edge server, the surrounding k edge servers are selected for monitoring based on the idea of kNN (k-Nearest Neighbor).
- *We design a series of experiments to validate the effectiveness of Rs-mBSRM*. To overcome the problem that the traditional data set does not meet the edge characteristics, we use the method of data set fusion to construct an edge data set for our experiments.

The remaining of the paper is organized as follow. Section II introduces the related work of probabilistic QoS monitoring. The theoretical foundation of our approach is introduced in Section III. Section IV introduces our approach in detail. Section V explains the evaluation process. Finally Section VI summaries the paper.

II. Related Work

Chan et al. [7]. proposed a .net-based framework, which employed the Probabilistic Computational Tree Logic (PCTL) language to define probabilistic constraints of non-functional attributes. They deployed Windows Management Instrumentation API (Application Programming Interface) for constraint checking. Lee et al. [9]. delivered a runtime probabilistic verification approach based on statistical analysis. This approach is implemented in the Monitoring and Checking (Mac) framework. Grunske et al. [10] designed $ProMo$, which is a monitoring approach based on acceptance sampling and sequential hypothesis testing. They utilized CSL^{MON}, a subset of continuous stochastic logic (CSL), to define probabilistic attributes. The attributes were verified with Sequential Probabilistic Ratio Test (SPRT) [19]. Zhang et al. [11] developed a property specification language – Probabilistic Timed Property Sequence Chart (PTPSC), an extension of Property Sequence Chart (PSC) [20]. Based on this language, a probability monitor is automatically generated by combining with the process of continuous statistical assumptions. To realize continuous monitoring, Grunske et al. [12] improved SPRT. By reusing the results of the previous hypothesis test, the time of statistical data is greatly reduced and the monitoring results are ensured. Zhu et al. [13] proposed a probabilistic monitoring approach based on Bayesian statistics, called $BaProMon$. By calculating the Bayesian factors to check the runtime information, the monitoring results are obtained based on hypothesis tests.

Since QoS values are dynamical and changeable in different environments, and previous approaches do not consider the environmental impact for QoS. Zhang et al. [14] proposed weighted BSRM (wBSRM) based on the principle of Bayesian classification. This approach combines the influence of environmental factors on the monitored samples and quantifies the sample weights based on the TF-IDF algorithm in the training stage. To ensure the dynamic and real-time monitoring performance, an extended wBSRM framework, IgS-wBSRM [15], was designed. IgS-wBSRM can dynamically update the classification results obtained during the training phase by using the sliding window mechanism and the information gain theory, so as to make the monitoring results more accurate and effective. However, all these approaches are designed for a single QoS attribute. To support multivariate monitor, Zhang et al. proposed the first work called Multivariate BRSM (M-BSRM) [16] to monitor multivariate QoS demands of users. M-BSRM adopts the information fusion theory to weight multiple QoS attributes based on user preferences to obtain comprehensive values, so as to more effectively monitor QoS.

In general, these approaches cannot solve the two gaps mentioned in mobile edge computing. To realize multivariate QoS monitoring, this paper proposes a novel QoS monitor-

ing approach named Rs-mBSRM.

III. Preliminaries

A. Mobile edge computing

Due to the limitation of resources, the problems such as high latency and instability in the traditional cloud computing models are inevitable. Mobile edge computing is one of the core technologies in the 5G era [21]. It provides services for application developers and service consumers on the edge of the network. The goal of mobile edge computing is to provide computing, storage, and network bandwidth close to data sources or users, thereby relieving the resource stress of cloud computing. Mobile edge computing, as a distributed architecture, migrates the applications, data and services from network center nodes (such as data centers and cloud centers) to network edge nodes (such as edge servers and mobile terminals) for processing [1]. This paradigm turns the bulk of the computation that was handled entirely by the central node into smaller and more manageable pieces and distributes them to the edge nodes. Using this architecture, data transmission and processing efficiency are dramatically improved. Consequently, mobile edge computing is more suitable to handle big data tasks.

B. Naive Bayesian classifier

Bayesian method is a normal representation and inference method of uncertain knowledge [22]. Bayesian theorem is a theory of calculating posterior probability based on the prior probability of a hypothesis and the probability of observing different data under the given hypothesis. The main idea of Bayesian classification algorithm is to calculate the probability of occurrence of all categories under given observation conditions, i.e., posterior probability. Then the category to be classified belongs to the category with the highest posterior probability. $X = \{x_1, x_2, ...x_D\}$ represents a sample to be classified with D attributes. $Y = \{y_1, y_2, ...y_k\}$ represents k categories. As shown in equation 1, X belongs to y_k with the highest probability.

$$y_k = \underset{y_k \in Y}{argmax}\, P(y_k|X) \tag{1}$$

According to Bayesian theorem:

$$P(y_k|X) = \frac{P(X|y_k)P(y_k)}{P(X)} \tag{2}$$

where $P(X)$ is a constant for $P(y_k|X)$, and $P(y_k)$ is the prior probability of y_k,

$$P(X|y_k) = P(x_1|y_k)P(x_2|y_k)...P(x_D|y_k) = \prod_{d=1}^{D} P(x_d|y_k) \tag{3}$$

In this way, Bayesian classification formula can be written as follows:

$$y_k = \underset{y_k \in Y}{argmax}\, P(y_k) \prod_{d=1}^{D} P(x_d|y_k) \tag{4}$$

C. Rough set theory

The main idea of rough set theory is to use known knowledge to describe imprecise or uncertain systems [23]. Compared with other theories in dealing with uncertain problems, the theory has the advantage that it does not need any prior information outside a data set. Consequently, the uncertainty description or treatment of the problem is relatively objective. Rough set theory has been applied in many fields, such as pattern recognition, data mining, image processing, etc [23]. Attribute importance measures the classification ability of attributes to an information system. Rough set theory can be used to measure the importance of attributes [24]. If an attribute is deleted from a knowledge representation system, the greater the change of the classification capability, the higher importance the attribute.

IV. The Rs-mBSRM Approach

A. The Overview of Rs-mBSRM

The overall framework of Rs-mBSRM is shown in Fig. 1. It contains three steps:

- The *first step* is data collection and preprocessing. The collected data includes the location information and the historical QoS data of edge servers. Each QoS data sample includes the values of multiple QoS attributes of a service in each edge server that is invoked by users. After cleansing invalid historical QoS data, each attribute value is normalized for the convenience of aggregation. The weight of each attribute is then determined based on rough set theory. Finally, the comprehensive QoS value in a data sample is calculated by aggregating the weighted values of each QoS attribute.
- The *second step* is classifier construction. By testing user-defined probabilistic standards, the prior information is calculated to construct a Bayesian classifier for each monitored service in an edge server.
- The *third step* is edge based multiple QoS monitoring. A Bayesian classifier is selected upon user mobility. If the monitored QoS data sample is obtained from the user's previous server, the classifier in the previous server will be continuously used for QoS monitoring. If the user's monitored data sample is gained from a new edge server that contains historical data, the classifier is selected from the new server and the attribute weights of the sample need to be updated based on the historical data. When there is no historical data in the new edge server, k surrounding edge servers need to be selected to determine the monitoring results.

B. The Detail of Rs-mBSRM

1) Data Collection and Preprocessing: The collected data includes the location of the edge server and the QoS data for services invoked in each edge server. The location of the edge server is used to find adjacent edge servers

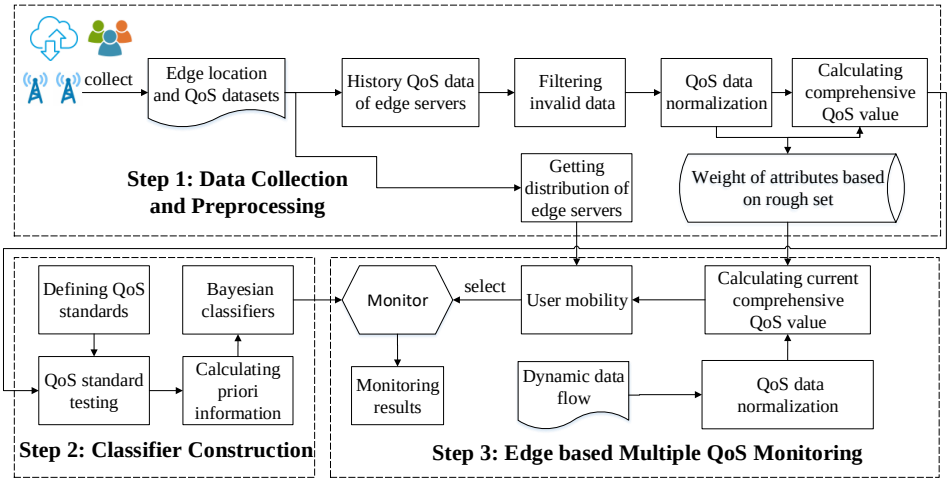


Figure 1: Overall workflow diagram of Rs-mBSRM

to monitor when the edge server does not have historical data. The QoS samples of each edge server are described as: $X = \{x_1, x_2, ..x_n\}$, where $x_i = \{a_1, a_2...a_d\}$ $(i \in [1, n])$ represents the i-th sample. Each sample records the QoS data of a service invoked by all the users of an edge server. Each sample contains d attributes, and $a_j (j \in [1, d])$ is the j-th attribute value of x_i. Data filtering mainly focuses on filtering invalid samples, such as samples with attribute values of -1. Since different attributes have different units and degrees of variation, we need to normalize the attributes data when the comprehensive QoS is evaluated. In addition, QoS attributes can be divided into positive constraints and negative constraints, so two standardized formulas are used to map these attribute values into [0,1]. For attributes with positive constraints such as *throughput* and *reliability*, where the higher the value, the better the QoS, we use formula 5 to normalize the attribute values. For attributes with negative constraints such as *response time*, where the smaller the value, the better the QoS, we use formula 6 to normalize the attribute values. a_{ij} is the original value of the i-th sample and the j-th attribute.

$$r_{ij} = \begin{cases} \dfrac{a_{ij} - a_j^{min}}{a_j^{max} - a_j^{min}}, a_j^{max} - a_j^{min} ! = 0 \\ 1 \qquad\qquad , a_j^{max} - a_j^{min} = 0 \end{cases} \tag{5}$$

$$r_{ij} = \begin{cases} \dfrac{a_j^{max} - a_{ij}}{a_j^{max} - a_j^{min}}, a_j^{max} - a_j^{min} ! = 0 \\ 1 \qquad\qquad , a_j^{max} - a_j^{min} = 0 \end{cases} \tag{6}$$

QoS monitoring is regarded as a dichotomous problem. $C = \{c_0, c_1\}$ is defined, where c_0 represents the condition that the comprehensive QoS meets the probabilistic standard, and c_1 represents the comprehensive QoS does not meet the probabilistic standard. Two kinds of posterior probabilities are obtained by constructing a Bayesian classifier, and the ratio of posterior probabilities is defined as $p = afterproc_0/afterproc_1$. When p>1, the monitoring result is c_0, and vice versa. When the weights of QoS attributes are calculated based on rough set theory, we believe that the attribute with greater changes of posterior probability ratio before and after removing the attribute should be given a higher weight. The specific process of weight calculation of the j-th attribute is as follows:

- For the i-th sample, the comprehensive QoS value before removing the j-th attribute is: $comprehensiveValue = \frac{1}{d}\sum_{k=1}^{d} r_{ik}$. The posterior probability ratio is denoted as $afterpro_i$.
- For the i-th sample, the comprehensive QoS value after removing the j-th attribute is: $comprehensiveValue_i = \frac{1}{d-1}\sum_{k=1,k\neq i}^{d} r_{ik}$. The posterior probability ratio is denoted as $afterpro_{ij}$.
- The sum of the differences in the posterior probability ratios of m training samples before and after removing the j-th attribute is $pro_j = \sum_{k=1}^{m} |afterpro_k - afterpro_{kj}|$.
- The weight of the j-th attribute is: $w_j = \frac{pro_j}{\sum_{k=1}^{d} pro_k}$.

Finally, the comprehensive QoS value of the i-th sample is: $comprehensiveQoS = \sum_{j=1}^{d} r_{ij} * w_j$.

2) Classifier Construction: When Bayesian classifiers are used to classify continuous data, it usually assumes that the data obey the Gaussian distribution. For QoS monitoring, a Bayesian classifier can be described as:

$$c_j = \underset{c_j \in C}{argmax}\, P(c_j) \prod_{i=1}^{n} P(x_i | c_j) \tag{7}$$

When a Bayesian classifier is trained by the historical data in each edge server, the probabilistic standard is tested by: $P(X{>}QoS\_Value) = 1 - \int_{-\infty}^{QoS\_Value} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-u)^2}{2\sigma^2}} dx$, where QoS_Value is the threshold of the comprehensive

QoS value, u represents the sample mean, and σ represents the sample standard deviation. If P is greater than probabilistic standard s, the current sample belongs to c_0; otherwise the current sample belongs to c_1. For example, the probability that the comprehensive QoS is greater than 0.5 is greater than 80%, where $QoS\_Value = 0.5$, $s = 80\%$. The prior probability $p(c_j)$ is calculated based on the maximum likelihood estimation:

$$P(c_j) = \frac{n_{c_j}}{n} \tag{8}$$

where n_{c_j} refers to the number of samples that belong to c_j, and n refers to the total number of training samples. And $P(x_i|c_j)$ is calculated by:

$$P(x_i|c_j) = N(u, \sigma^2) \tag{9}$$

where N represents the normal distribution of the sample, and x_i is the comprehensive QoS value.

3) Edge based Multiple QoS Monitoring: Once a current monitored sample is obtained and normalized, the major mission of the QoS monitoring stage is to calculate the posterior probability ratio of the sample via the classifier of the server. The monitoring result of the sample is determined by step 2. If the posterior probability ratio is greater than 1, the monitoring result belongs to c_0; otherwise, the monitoring result belongs to c_1. If the sample is obtained from the new edge server that contains historical data, the sample attribute weights are updated by rough set theory (Algorithm 2) and the comprehensive QoS value of the sample is monitored by the classifier of the new edge server (Step 2). If the new edge server does not contain historical data, the k nearest edge servers that contain historical data around the edge server are selected in terms of their distances $d_i(i \in [1, k])$ The comprehensive values of the sample in the k edge servers are calculated based on the attribute weights of each server. The posterior probability ratios of the sample in the k edge servers are calculated as $posterior_i(i \in [1, k])$. The final posterior probability ratio of the sample is the weighted aggregation of the posterior probability ratios of the sample in these servers, where the weights are inverse to these servers' distances. The aggregation is represented as $\sum_{i=1}^{k} \frac{x}{d_i} * posterior_i$, where x is calculated by $\sum_{i=1}^{k} \frac{x}{d_i} = 1$.

Algorithm 1 depicts the edge based multiple QoS monitoring process. Algorithm 2 describes the attribute weight calculation process via rough set theory.

Algorithm 1 edgeMulMonitoring

Require: The monitored sample $x_k = \{a_1, a_2, ...a_d\}$; The sample stream of each edge server S; probabilistic QoS standard β; QoS threshold QoS_Value;
Ensure: Monitoring result c_j;

```
if isMove == False then
    w[d] = computeWeight(oldedge)
    comvalue = w[1]*a[1]+w[2]*a[2]+...+w[d]*a[d]
    afterpro = computeAftPro(comvalue, oldedge)
else
    if isEmpty == False then
        w[d] = computeWeight(newedge)
        comvalue = w[1]*a[1]+w[2]*a[2]+...+w[d]*a[d]
        afterpro = computeAftPro(comvalue, newedge)
        //Calculate the posterior probability ratio
    else
        edge[k] = Top_k(nearest_edge)
        w[d] = computeWeight(edge[k])
        comvalue[k] = w[1]*a[1]+w[2]*a[2]+...+w[d]*a[d]
        pro[k] = computerAftPro(comvalue[k], edge[k])
        wegde[k] = computerWi(d[k])
        afterpro = wegde[1]*pro[1]+wegde[2]*pro[2]+... + wegde[k]*pro[k]
    end if
end if
if afterpro > 1 then
    return c_0
else if afterpro <1 then
    return c_1
else
    return indecisive
end if
```

- RQ1: Is rough set effective to improve the monitoring performance?
- RQ2: Is Rs-mBSRM more efficient than existing multivariate approaches on QoS monitoring such as M-BSRM?
- RQ3: Is Rs-mBSRM more suitable for mobile edge environments?
- RQ4: Does the number of peripheral servers selected affect the final monitoring results?

V. Experiment

In this section, we designed a series of experiments to validate the Rs-mBSRM approach based on public data. Our experimental evaluation was conducted on a PC with Intel(R) Core(TM) i5-4200M CPU@2.50GHz, 4.00GB RAM, and Windows 10. We develop and implement Rs-mBSRM on Geany using Python. The experiments were designed to investigate the following four research questions:

A. The Experimental Data

Our experimental data mainly includes three parts: i) A real-world QoS data set *QWS* [25]. This data set consists of 150 files. Each file name indicates a user's IP address. Each file contains the QoS data of 100 different services invoked by a user. From these files we can obtain four QoS attribute values: *response time*, *throughput*, *reliability*, and *availability*. ii) A *Shanghai Telecom* dataset [26], which contains location information of 3233 base stations and

Algorithm 2 computeWeight

Require: The QoS samples $S = \{x_1, x_2, ...x_m\}, x_i = \{a_1, a_2, ...x_d\}$; probabilistic QoS standard β; QoS threshold QoS_Value;
Ensure: Attribute weights $w[d]$;
1: **for** each sample **do**
2: $r[i][j]$ = normalizing($a[i][j]$)
3: $compvalue$ $(r[i][1] + r[i][2] + ...r[i][d])/d$
4: $compvalue_i$ = $(r[i][1] + ...r[i][j-1] + r[i][j+1] + ...r[i][d])/(d-1)$
5: $afterpro$ = computerAfterpro($compvalue$)
6: $afterpro_i$ = computerAfterpro($compvalue_i$)
7: $pro[j] = pro[j] + |afterpro - afterpro_i|$
8: **end for**
9: **for** each attribute **do**
10: $sum = sum + pro[j]$
11: **end for**
12: $w[j] = pro[j]/sum$

records of users who called these base stations. iii) A simulated data set according to the probabilistic criterion. Since real data set cannot reflect whether or not a service is normal, error samples are injected into the QWS data set to create a data set to validate the effectiveness of Rs-mBSRM.

Since traditional data sets do not meet the requirements of the edge environment, we employed the principle of data fusion to fuse the real and simulated QoS data set with the base station locations from the *Shanghai Telecom* data set, so as to construct a data set which can meet the edge characteristics. First, the users from *QWS* data set with same network numbers, i.e., the first byte of an IP address, are regarded as being in same edge servers. Therefore, the QoS data of the 150 users were divided into 46 groups according to the network numbers. Second, 46 base stations from the *Shanghai Telecom* data set were randomly selected as the locations of the edge servers. The geographical distribution of the 46 edge servers is shown in Fig. 2. Finally, the QoS data is randomly corresponding to the base stations.

Data preprocessing includes invalid data filtering and data normalization. The invalid QoS sample data contains attribute (such as *response time*) values of -1. The data of *response time*, *throughput*, *reliability* and *availability* were normalized by formulae 5 and 6 respectively.

B. Experimental Results

To address RQ1, we compared the monitoring performance between the Rs-mBSRM approaches with and without rough sets based on the QWS dataset. We randomly selected four edge servers (whose IDs are 11, 19, 21, and 39). The probabilistic QoS standards for the servers are described as follows: "the probability the comprehensive QoS value being greater than 0.5 is greater than 80%", "the probability the comprehensive QoS value being greater than 0.4 is greater than 80%", "the probability the comprehensive QoS value being greater than 0.4 is greater than 70%". For each edge server, the first 2000 QoS data samples are extracted for training, and the rest 5000 data samples are used for monitoring performance testing.

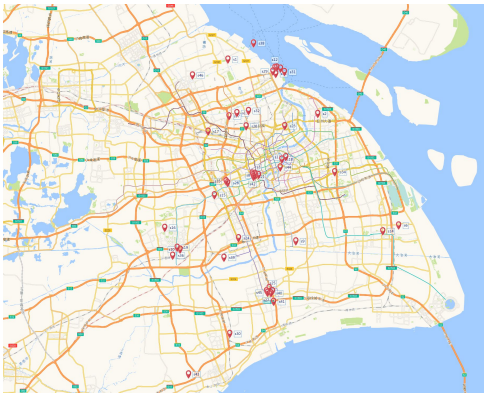


Figure 2: Distribution of edge servers

Table I: Attributes weights

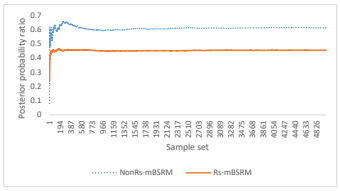
Edge server ID	w_{rt}	w_{tp}	w_{re}	w_{av}
11	0.1052	0.8053	0.0599	0.0296
19	0.0618	0.4894	0.0161	0.4326
21	0.0698	0.3810	0.2064	0.3427
39	0.1623	0.3411	0.0820	0.4146

Table I shows the rough set based weights of *response time*, *throughput*, *reliability* and *availability* of the four edge servers against their corresponding QoS standards. It can be seen that their weights of attributes are different because of different network conditions.

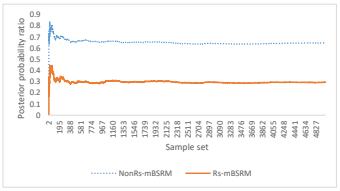
We used the posterior probability ratios to display the monitoring results to demonstrate their variations more intuitively. When the posterior probability ratio is greater than 1, it represents a probabilistic standard being met; otherwise the standard is not met. Fig. 6 shows the monitoring results of the four edge servers against their respective probabilistic QoS standards. NonRs-mBSRM refers to the Rs-mBSRM approach without rough sets, where the attribute weights are set as 0.25. The figure shows that the posterior probability ratio based on rough set is significantly lower than that without rough sets, which indicates that Rs-mBSRM can more effectively monitor service failures with rough sets.

To address RQ2, we compare Rs-mBSRM with M-BSRM [16], which is superior to existing single QoS monitoring approaches.

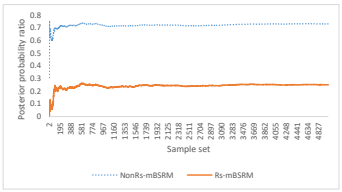
The edge server with ID of 20 is randomly selected. The probabilistic standard is defined as "the probability the comprehensive value being greater than 0.5 is greater 80%". Error samples are injected into 1000∼2000 and 3000∼4000



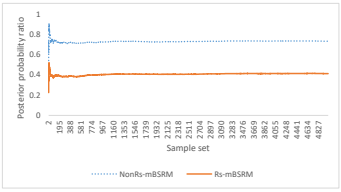
(a) Ratio when edge server ID=11



(b) Ratio when edge server ID=19



(c) Ratio when edge server ID=21



(d) Ratio when edge server ID=39

Figure 3: Posterior probability ratios of different edge servers

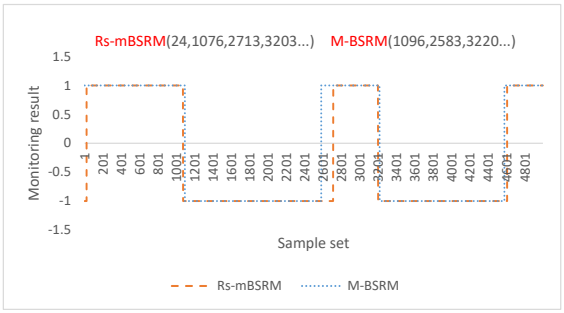


Figure 4: Monitoring results of different approaches

of the monitored sample sequence, i.e., the samples with the comprehensive values being less than 0.5 are more than 20% in the data set. The monitoring results of the two approaches are shown in Fig. 4. Rs-mBSRM and M-BSRM respectively detect the first group of service failures at 1076 and 1096. Since Rs-mBSRM can distinguish the different importance of the attributes based on rough sets, it can detect service failures more efficiently. Similarly, Rs-mBSRM outperforms M-BSRM for the second group of service failures. Table II compares the average $training\ time$ and $monitoring\ time$ of the two approaches against 6 different probabilistic QoS standards. Due to the complexity of calculating the attribute weights and the integrals of the continuous data, Rs-mBSRM takes slightly more time than M-BSRM. Rs-mBSRM is still deemed as efficient considering the relatively small scale (10s ms) of time spent on training and monitoring.

To address RQ3, we randomly selected $user1$ and $user14$ from the *Shanghai telecom* data set. The users' invoked edge servers are shown in Table III and Table IV, respectively. This experiment is based on the *QWS* data set. The probabilistic QoS standard is "the probability the

Table II: Training and monitoring time

Time(ms)		QoS threshold					
		0.48	0.49	0.50	0.51	0.52	0.53
Training	Rs-mBSRM	41.96	45.23	36.88	37.48	35.67	40.03
	M-BSRM	20.45	28.27	23.89	26.72	20.67	29.36
Monitoring	Rs-mBSRM	26.25	25.88	25.85	27.04	24.39	29.65
	M-BSRM	20.03	21.34	21.39	20.88	18.09	19.85

Table III: Edge servers called by user 1

User ID	1					
Edge server ID	3	4	8	21	22	42

Table IV: Edge servers called by user 14

User ID	14					
Edge server ID	5	12	26	29	31	33

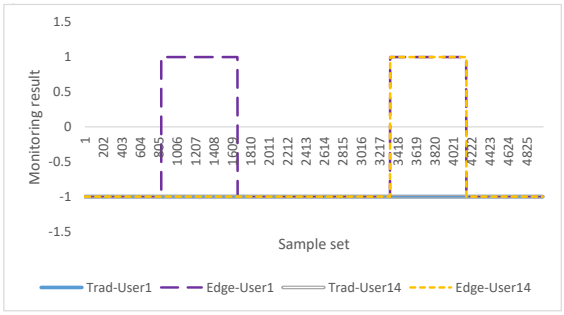
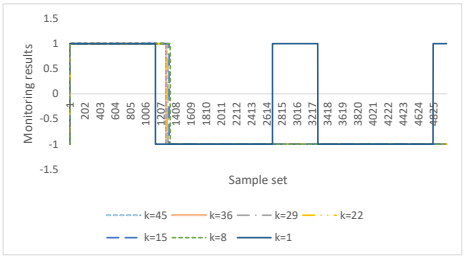


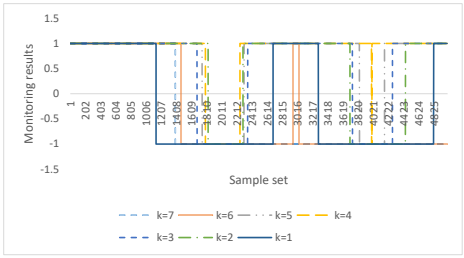
Figure 5: Monitoring results in different environments

comprehensive value being greater than 0.5 is greater 80%". Fig. 5 compares the monitoring results of the two users in the mobile edge and traditional environments. In the mobile edge environment, when users invoke new edge servers, the historical data from old edge servers is invalid and monitoring results are obtained from the new servers. As shown in Fig. 5, since the change of edge servers, the monitoring results of the service invoked by $user1$ varied at 833 and 1666 in the mobile edge environment. In contrast, in the traditional environment, the changes of the service state were not detected because monitoring results were obtained from the centralized storage. The similar case happened to $user14$. Consequently, Rs-mBSRM is more effective in the edge environment.

To address RQ4, we randomly selected an edge server with ID of 8. When the edge server does not contain historical data, the monitoring results are obtained from the k neighboring edge servers. The probability standard is "the probability the comprehensive value being greater than 0.5 is greater than 80%". Similarly, error samples are injected into 1000∼2000 and 3000∼4000 of the monitored sample sequence. According to the experimental results, the range of k is gradually narrowed down, and the optimal value of k is finally determined based on the monitoring performance. Since the total number of edge servers is 46, we narrow



(a) k=1∼45



(b) k=1∼7

Figure 6: Monitoring results for different k values

down the value of k from 45 to 1 with a step of 7. As shown in Fig. 6a, when $k \geq 8$, the service failures in the second stage cannot be monitored. Next, we narrow down k from 7 to 1. As shown in Fig. 6b, when k=1, the monitoring performance is optimal. Therefore, the number of selected peripheral edge servers can lead to different monitoring results.

VI. Conclusion

In this paper, a novel QoS monitoring approach, Rs-mBSRM, is proposed to address the limitations of traditional QoS monitoring approaches in mobile edge computing. For future work, it is important to study the need of updating QoS attribute weights in same edge servers. In addition, we will explore the relevance among multivariate QoS attributes.

VII. Acknowledgements

The work is supported by the National Natural Science Foundation of China under Grant No. 61572171, the Natural Science Foundation of Jiangsu Province under Grant No. BK20191297, and the Fundamental Research Funds for the Central Universities under Grant No. 2019B15414.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[2] J. El Haddad, M. Manouvrier, and M. Rukoz, "TQoS: Transnational and QoS-aware selection algorithm for automatic web service composition," *IEEE Transactions on Services Computing*, no. 1, pp. 73–85, 2010.

[3] L. Baresi and S. Guinea, "Towards dynamic monitoring of WS-BPEL processes," in *International Conference on Service-Oriented Computing*, pp. 269–282, Springer, 2005.

[4] L. Zeng, H. Lei, and H. Chang, "Monitoring the QoS for web services," in *International Conference on Service-Oriented Computing*, pp. 132–144, Springer, 2007.

[5] F. Raimondi, J. Skene, and W. Emmerich, "Efficient online monitoring of web-service SLAs," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, pp. 170–180, ACM, 2008.

[6] L. Grunske, "Specification patterns for probabilistic quality properties," in *2008 ACM/IEEE 30th International Conference on Software Engineering*, pp. 31–40, IEEE, 2008.

[7] K. Chan, I. Poernomo, H. Schmidt, and J. Jayaputera, "A model-oriented framework for runtime monitoring of nonfunctional properties," in *Quality of Software Architectures and Software Quality*, pp. 38–52, Springer, 2005.

[8] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal aspects of computing*, vol. 6, no. 5, pp. 512–535, 1994.

[9] I. Lee, O. Sokolsky, J. Regehr, *et al.*, "Statistical runtime checking of probabilistic properties," in *International Workshop on Runtime Verification*, pp. 164–175, Springer, 2007.

[10] L. Grunske and P. Zhang, "Monitoring probabilistic properties," in *Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pp. 183–192, ACM, 2009.

[11] P. Zhang, W. Li, D. Wan, and L. Grunske, "Monitoring of probabilistic timed property sequence charts," *Software: Practice and Experience*, vol. 41, no. 7, pp. 841–866, 2011.

[12] L. Grunske, "An effective sequential statistical test for probabilistic monitoring," *Information and Software Technology*, vol. 53, no. 3, pp. 190–199, 2011.

[13] Y. Zhu, M. Xu, P. Zhang, W. Li, and H. Leung, "Bayesian probabilistic monitor: A new and efficient probabilistic monitoring approach based on bayesian statistics," in *2013 13th International Conference on Quality Software*, pp. 45–54, IEEE, 2013.

[14] P. Zhang, Y. Zhuang, H. Leung, W. Song, and Y. Zhou, "A novel QoS monitoring approach sensitive to environmental factors," in *2015 IEEE International Conference on Web Services*, pp. 145–152, IEEE, 2015.

[15] P. Zhang, H. Jin, Z. He, H. Leung, W. Song, and Y. Jiang, "IgS-wBSRM: A time-aware web service QoS monitoring approach in dynamic environments," *Information and software technology*, vol. 96, pp. 14–26, 2018.

[16] P. Zhang, H. Jin, H. Dong, and W. Song, "M-BSRM: Multivariate bayesian runtime QoS monitoring using point mutual information," *IEEE Transactions on Services Computing*, 2019.

[17] S. Wang, Y. Zhao, L. Huang, J. Xu, and C.-H. Hsu, "QoS prediction for service recommendations in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 134–144, 2019.

[18] Z.-Z. Liu, Q. Z. Sheng, X. Xu, D. Chu, and W. E. Zhang, "Context-aware and adaptive QoS prediction for mobile edge computing services," *IEEE Transactions on Services Computing*, 2019.

[19] A. Wald, "Sequential tests of statistical hypotheses," *The annals of mathematical statistics*, vol. 16, no. 2, pp. 117–186, 1945.

[20] P. Zhang, B. Li, and L. Grunske, "Timed property sequence chart," *Journal of Systems and Software*, vol. 83, no. 3, pp. 371–390, 2010.

[21] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[22] J. M. Bernardo and A. F. Smith, *Bayesian theory*, vol. 405. John Wiley & Sons, 2009.

[23] Z. Pawlak, "Rough set theory and its applications to data analysis," *Cybernetics & Systems*, vol. 29, no. 7, pp. 661–688, 1998.

[24] Y. Yao and Y. Zhao, "Attribute reduction in decision-theoretic rough set models," *Information sciences*, vol. 178, no. 17, pp. 3356–3373, 2008.

[25] Z. Zheng and M. R. Lyu, "Collaborative reliability prediction of service-oriented systems," in *2010 ACM/IEEE 32nd International Conference on Software Engineering*, vol. 1, pp. 35–44, IEEE, 2010.

[26] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 160–168, 2019.