Mobility and Dependence-aware QoS Monitoring in Mobile Edge Computing

Pengcheng Zhang, *Member, IEEE*, Yaling Zhang, Hai Dong, *Senior Member, IEEE* and Huiying Jin

Abstract—Mobile edge computing is a new computing paradigm that performs computing on the edge of a network. It provides services to users by deploying edge servers near mobile devices. Services may be unavailable or do not satisfy the needs of users due to changing edge environments. Quality of service (QoS) is commonly employed as a critical means to indicate qualitative status of services. It is particularly important to monitor QoS of services timely and effectively in the mobile edge environment. However, user mobility and dependencies among QoS values often cause the monitoring results to deviate from the real results in the mobile edge environment. Existing QoS monitoring approaches have not taken into account these problems. To address the problems, this paper proposes ghBSRM-MEC (Gaussian hidden BayeSian Runtime Monitoring for Mobile Edge Computing), a novel mobility and dependence-aware QoS monitoring approach for the mobile edge environment. This approach assumes that the QoS attribute values of edge servers obey Gaussian distribution. It constructs a parent property for each property, thus reducing the dependence between properties. During the training stage, a Gaussian Hidden Bayesian classifier is constructed for each edge server. During the monitoring stage, combining with a KNN algorithm, the classifier is changed dynamically based on user mobility to realize QoS monitoring in the mobile edge environment. The experimental results validate the feasibility, effectiveness, and efficiency of ghBSRM-MEC.

Index Terms—Cloud computing; mobile edge computing; QoS; monitoring; Bayesian classifier; K-nearest neighbor.

1 Introduction

Mobile edge computing (MEC) [1] is an emerging technology, which provides services by deploying an edge server (e.g., firewall, router, or similar devices) near mobile clients (e.g., smartphones, sensors, or similar edge ends), and between mobile clients and cloud servers. It features short response time and fast processing speed [2]. With the continuous development of various novel technologies, Web services are increasingly being applied in many fields of people's lives, including business, manufacturing, healthcare, entertainment, etc [3]. On the one hand, the number of Web services deployed in cloud servers is growing rapidly. On the other hand, these services are gradually moved to edge servers, i.e., mobile edge services, which reside in nearby edge servers to serve users. Different service providers may provide services with similar functions, and the performance of the same services may vary in different edge servers. How to select an appropriate service that meets the needs of users has therefore drawn many researchers' attention [4]. Thus, the concept of QoS (Quality of Service) is introduced. QoS represents the a set of non-functional attributes of services, including *response time*, *throughput*, *reliability* and *availability*, etc [5].

Users expect to select mobile edge services with guaranteed QoS. The QoS information of a service is usually disclosed by its service provider. Some providers may provide false QoS information to mislead users. This false information cannot be employed to objectively evaluate services. Therefore, to objectively and correctly evaluate the operation of services, it is particularly important to monitor QoS attributes timely and effectively at runtime [6], [7], [8].

Monitoring is one of the most effective ways to verify whether a service is valid at runtime [9]. In general, QoS attributes can be expressed by probabilistic quality attributes [10]. For example, *response time* can be described as "the probability that a service's response time for a customer's request is less than 3.6 seconds is more than 80%". The problem of QoS monitoring is therefore transformed into the probabilistic calculation and analysis whether the collected runtime information satisfies pre-defined QoS requirements. Based on the probabilistic quality attributes, researchers have proposed many QoS monitoring approaches in traditional network environments, respectively from the perspectives of probability calculation [11], traditional hypothesis testing theory [12], [13], [14], and Bayesian theory [15], [16], [17]. However, the existing approaches are infeasible in mobile edge environments due to the following two major problems.

The first problem is *user mobility and edge server differences challenge the applicability of traditional QoS monitoring approaches in distributed mobile edge environments*. The edge servers served for same mobile clients may vary in mobile edge environments due to changes of user positions [18]. The quality of the same services located in different edge servers may vary due to changes of server conditions and network environments. During the process of QoS monitoring, when users move to new edge servers, the historical QoS data on old edge servers may become invalid references for same services. In contrast, the traditional monitoring

- *P. Zhang, Y. Zhang and H. Jin are with the College of Computer and Information, Hohai University, Nanjing, China & State key Laboratory of Networking and Switching Technology, Beijing, China*
  E-mail: pchzhang@hhu.edu.cn; 2460154274@qq.com; 367046895@qq.com
- *H. Dong is with the School of Computing Technologies, RMIT University, Melbourne, VIC 3001, Australia*
  Email: hai.dong@rmit.edu.au

Manuscript received XXXX XX, XXXX; revised XXXX XX, XXXX.

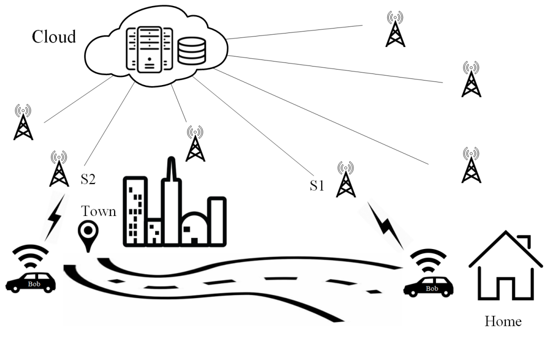


Fig. 1. Service invocation scenario in mobile edge computing

approaches do not need to consider changes of edge servers since services are all stored and provisioned in centralized environments. Traditional approaches that only make decisions based on the centralized historical QoS data may cause false monitoring results. For example, let us assume that a user *Bob* is driving from his home to a small town for a vacation. He departs from his home and uses Google Maps to navigate by calling the edge server *S1*, as shown in Figure 1. When *Bob* arrives at the small town, Google Maps is invoked through a new edge server (i.e., *S2*) with different server conditions and network environments. Now the question is how to monitor whether the QoS of Google Maps provisioned by this edge server meets *Bob*'s requirements. For example, *Bob* expects the response time of Google Maps to be within 3.6s for most of this journey. We can translate this requirement into a probabilistic description: the probability that response time is less than 3.6s is more than 80%. Traditional QoS monitoring approaches would use the historical data of *S1*. For the monitoring results of *S2*, the historical data in *S1* becomes invalid due to the server variation. Consequently, we need to use the historical data of the current edge server (i.e., *S2*) or peripheral edge servers during the monitoring process.

The second problem is *dependence among QoS values results in the deviation of monitoring results.* One of the most crucial factors affecting QoS values is the context information. On the user side, proximate users tend to employ similar IT infrastructure. Hence the QoS values from same services are likely to be dependent for these proximate users [19]. On the service side, the performance of a service relies on its running environment. The QoS values of MEC services are probably dependent in same running environments [20]. In MEC, edge servers are used to perform service execution and data storage, which can be viewed as the running environments. Therefore, there might exist dependence among the QoS values of same services from same edge servers invoked by proximate users. Existing QoS monitoring approaches neglect the influence among QoS values. They assume the QoS values are independent of each other, which may cause judgement delays on service monitoring results. In the sample scenario, when *Bob* arrives at the small town, Google Maps can be invoked through *S2*. Assume that *S2* contains the historical samples of response time of Google Maps invoked by other users. Now a relatively larger portion of the historical samples meet *Bob*'s requirement. The existing QoS monitoring approaches assume that the current monitored QoS is irrelevant to the historical samples in *S2*. This would greatly reduce the monitoring probability that *Bob*'s requirement is met, and thus might lead to the delay or deviation of the monitoring results.

To solve the above two main problems, this paper proposes ghBSRM-MEC (Gaussian hidden BayeSian Runtime Monitoring for Mobile Edge Computing), a novel mobility and dependence-aware QoS monitoring approach under mobile edge computing. In general, the approach is divided into two stages: *training* and *monitoring* stages. During the *training stage*, to reduce the dependence between attribute values, we construct parent attributes for QoS attributes, which represent the influence of the other attributes on an attribute. In the Bob's sample scenario, when Bob arrives at the small town, Google Maps can be invoked through *S2*. The current monitoring sample is more dependent on the historical data of *S2*. Constructing parent attributes is an effective method to reflect this dependence. Based on the parent attributes, a corresponding Gaussian hidden Bayesian classifier is constructed for each edge server. During the *monitoring* stage, the following three situations are considered based on user mobility: *i)* The user does not invoke a new edge server. We monitor the service based on the Bayesian classifier constructed upon the historical QoS data of the current edge server. *ii)* The user moves to a new edge server which contains a classifier trained upon its historical data. In this case, the classifier on the new edge server is employed for monitoring. *iii)* The user moves to an edge server without historical data. In this case, the KNN algorithm is adopted to select the adjacent edge servers to obtain monitoring results. Finally, ghBSRM-MEC is validated by experiments on both real data sets and simulated data sets.

In summary, the main contributions of this paper are described as follows:

- *We devise a novel QoS monitoring approach considering user mobility and edge server differences in mobile edge environments.* A Bayesian classifier is constructed for performing monitoring in each edge server. The classifiers can be switched dynamically adapting to user mobility. The classifiers reference QoS information from peripheral servers based on the KNN algorithm for monitoring.
- *We design an effective method to lessen the impact of QoS attribute value dependence on monitoring results.* A parent attribute is established for each QoS attribute to construct a hidden Bayesian classifier to reduce the dependence among QoS attribute values. It is also able to address the problem of service failure detection delays caused by traditional Bayesian classifiers.
- *We design a series of experiments to comprehensively validate ghBSRM-MEC.* By fusing a Shanghai Telecom data set [1] and a traditional QoS data set [2], we obtain a mobile edge dataset applicable for edge QoS monitoring evaluation. The experimental results show the feasibility, effectiveness, and efficiency of the ghBSRM-MEC approach. It also proves

1 http://sguangwang.com/TelecomDataset.html
2 http://wsdream.github.io/dataset/wsdreamdataset1.html

that ghBSRM-MEC outperforms state-of-the-art approaches."

The remaining parts of the paper are organized as follows. Section 2 summarizes the related QoS monitoring approaches in recent years and points out the problems in mobile edge environments. Section 3 introduces the relevant concepts used in our approach. Section 4 gives a detailed description of the ghBSRM-MEC approach. We validate ghBSRM-MEC based on real-world and simulated datasets in Section 5. Finally, Section 6 summarizes the paper and gives prospects in the future.

2 Related work

2.1 Traditional QoS Monitoring

Traditional monitoring approaches aim at employing monitoring tools to obtain real-time data or timely user feedback.

Zeng et al. [21] monitored QoS by designing a QoS observation model. The monitoring system can systematically detect QoS and route service operation events. However, because of defining enterprise-level metrics and evaluation formulas, this approach does not consider user requirements and has poor scalability. Radovanovic et al. [22] deployed a monitoring system in the cloud environment based on TR-069, also known as CWMP (CPE WAN Management Protocol) [23], which is a remote management protocol. Its cloud based access interface provides necessary information for mobile applications, making acquired QoS parameters visualized. This monitoring system is based on individual requirements of end-devices, which are feasible in most cases. Michlmayr et al. [24] proposed a QoS monitoring framework that combines both client-side and server-side monitoring. It is based on event handling and informs interested users of current QoS values and violated service-level protocols. If the QoS does not meet the requirements, it may trigger adaptive actions. However, the performance overhead of the proposed framework is too large. Coppolino et al. [25] integrated two frameworks, SocIoS and QoS-MONaaS (QoSMONitoring as a Service). The SocIoS framework utilizes the QoS monitoring component developed in the SRT-15 project. QoS-MONaaS is used for monitoring the SocIoS application. Raimond et al. [26] proposed a non-intrusive online monitoring approach based on SLA (Service Level Agreement). The agreement violation can be discovered by inferring the type of exchanged message and the timestamp. However, the extensibility of this approach needs to be further investigated. This approach only considers time-related attributes, ignoring the fact that SLA may impose requirements on non-time attributes. These approaches rely on the instrumental capacity to acquire real-time data and timely user feedback, which is costly and lacks flexibility. In addition, they cannot meet fuzzy requirements of users.

2.2 Probabilistic QoS Monitoring

Probabilistic monitoring approaches have emerged in recent years. These approaches transform users' fuzzy requirements into probabilistic descriptions, and make monitoring decisions by using historical data for statistical analysis. At present, there are three primary types of QoS monitoring approaches according to the underlying theories.

The first type is based on the traditional probability calculation. Chan et al. [11] first proposed a probabilistic monitoring approach. PCTL (probabilistic computation tree logic) language was used to define the probabilistic quality standard of non-functional attributes. Then they calculated the ratio of the number of successful samples to the total number of monitoring samples and compared it with the pre-defined probability standard. If it meets the pre-defined standard, the service will be considered as normal. Otherwise, the service is regarded as abnormal. The approach has not been analyzed and validated using statistics, and it always brings larger errors.

The second type of probabilistic monitoring approach is based on the hypothesis testing theory. Sammapun et al. [12] first calculated the probability that the number of successful samples accounted for the total number of samples and then used a hypothesis test to determine whether the system meets the probabilistic quality attribute criteria at a given confidence level. Grunske et al. [13] proposed a probabilistic monitoring method called ProMo based on acceptance sampling and continuous hypothesis testing. This approach extends the existing statistical model testing technology at runtime and defines the probabilistic logic CSL^{mon} for monitoring. CSL^{mon} is a subset of CSL (continuous stochastic logic). In their approach, SPRT (Sequential Probabilistic Ratio Test) [27] is used to validate the correctness of the CSL^{mon} formula at level α and 1-β. In their further work, Grunske et al. [14] improved SPRT with continuous monitoring by regressing statistical analysis and reusing the previous hypothesis test results. However, when the actual probability of the system is in the undecided area, the approach still fails to make a conclusion.

The third type is based on the Bayesian theory. The characteristic of the Bayesian approaches is to add historical empirical data to present prediction judgment, which combines prior probability and likelihood probability to express uncertainty. Zhu et al. [15] first proposed a Bayesian probabilistic QoS monitoring approach, namely BaProMon. By calculating Bayesian factors, it checks the runtime information to estimate whether the monitoring results support the original hypothesis or alternative hypothesis. The differences are increased and the model equivalence is maintained by reusing the previous monitoring results. However, the prior distribution has a great impact on the validity of this approach. A major problem is that it is very difficult to choose a proper prior distribution. Zhang et al. [16] proposed a weighted naive Bayesian probabilistic monitoring approach, namely wBSRM. It considers the influence of environmental factors and makes the monitoring more practical for QoS. The TF-IDF algorithm is used to quantify the impact of environmental factors on monitoring, where the quantitative value is used as the weight of environmental factors to weigh each sample. In their further work [17], they proposed an improved approach, called lgS-wBSRM. This approach makes use of a sliding window mechanism to discard early redundant samples in time. Then it is combined with the information gain theory to update weights dynamically. In addition, to fully consider the multiple QoS requirements of users, they proposed M-

BSRM (Multivariate BayeSian Runtime Monitoring) [28], which calculates multivariate QoS values by assigning their attribute weights.

With the prosperity of mobile edge computing, service migration in mobile edge computing also becomes a research hotspot. Wang et al. [29] described the service migration problem as a Markov Decision Process (MDP) and proposed a mathematical framework to design the optimal service migration strategy. Zhang et al. [30] proposed a service migration strategy based on multi-attribute decision making, which can effectively migrate services to appropriate servers. In addition, many researchers have carried out studies in the areas related to QoS evaluation in mobile edge computing, such as QoS prediction and monitoring. Yin et al. [31] employed CNN to perform neighbor selection by learning deep features of complex edge devices and achieved accurate QoS prediction results in mobile edge computing. Liu et al. [20] proposed an improved artificial bee colony (ABC) algorithm to optimize the support vector machine (SVM). They improved Case-Based Reasoning (CBR) to predict multiple QoS attribute values according to the predicted workload and other task-related contextual factors. In contrast, the research on QoS monitoring in mobile edge environments is still underway. Zhang et al. [32] proposed a monitoring approach named Rs-mBSRM (multivariate BayeSian Runtime Monitoring using Rough set) for mobile edge computing, which calculates the weights of different QoS attributes based on rough set theory and historical samples. Then the comprehensive value is calculated for monitoring.

Nevertheless, these monitoring approaches including traditional QoS monitoring approaches are less applicable to mobile edge environments due to their ignorance of user mobility and the dependence among QoS values.

To address the monitoring result deviation problem caused by user mobility and dependence among QoS values, this paper proposes a novel QoS monitoring approach using a Gaussian hidden Bayesian classifier combined with a KNN algorithm in mobile edge environments.

3 Preliminaries

The principles of mobile edge computing, services, edge services, hidden naive Bayes and KNN are introduced in Section 3.2, Section 3.1, Section 3.3 and Section 3.4, respectively.

3.1 Mobile Edge Computing

With the development of the IoT (Internet of Things) technology, data over networks experiences an explosive growth. The cloud computing model based on centralized management cannot meet the needs of big data processing, on account of the following reasons: 1) *Real time*. The massive amount of real-time data generated by edge devices makes the performance of cloud computing gradually reach its bottleneck [33]. How to meet the requirements of low response time for emerging interconnected applications is the main direction of future research [34]. 2) *Privacy protection*. In the cloud computing model, various user privacy data is uploaded to cloud centers, which increases the risk

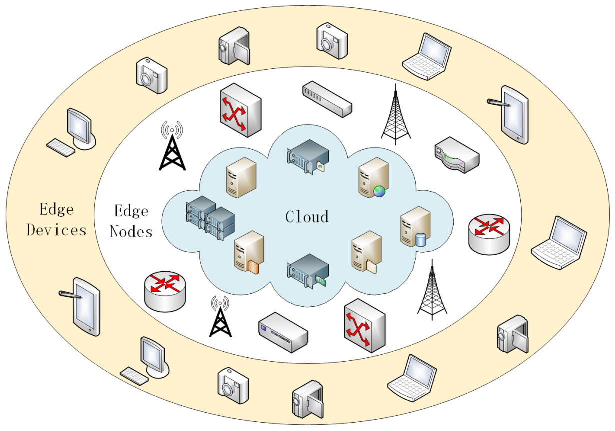


Fig. 2. Component distribution for MEC

of user privacy data leakage [35]. 3) *Energy consumption*. The explosive growth of data dramatically increases the energy consumption of cloud centers [36]. Computing performed on edge devices emerges as a new computing model to satisfy the needs of IoT applications, resulting in the concept of mobile edge computing.

Mobile edge computing (MEC) refers to a new computing model that performs computing on the network edge by deploying MEC servers between cloud centers and mobile devices. Its infrastructure can be divided into two parts: computing units and communication units. Most computing tasks and data of cloud centers are migrated to distributed edge servers for execution and storage. Since edge servers are deployed near data sources, edges can process and analyze data in real time to reduce latency [36]. For example, in online shopping platforms, it will improve user experience, if the operation of shopping cart updating is migrated from cloud centers to edge servers. As shown in Fig. 2, due to the geographical proximity of MEC servers to users, the response time to user requests is greatly reduced [37]. Directly uploading private data from home computers to cloud centers would increase the risk of user privacy breach. In MEC, edge servers can be used to process and protect the privacy data before being sent to cloud centers, which can reduce privacy disclosure risks. Shifting computing capabilities from cloud centers to edge nodes not only reduces energy consumption of cloud centers, but also optimizes the energy consumption of data transmission. For devices with limited power, such as drones, it is especially important to reduce the transmission power consumption.

3.2 Services and Edge Services

Traditional services provision communication media between applications in clouds enabling software interoperations over the Internet [38]. A traditional service usually contains functional attributes and non-functional attributes [39]. The functional attributes usually include service inputs (I), service outputs (O), precondition weights (PRW), post condition weights (PCW), service descriptions (D), etc. The non-functional attributes mainly contain Quality of Service (QoS), such as response time, reliability, credibility, etc. Edge services are a new generation of services, which facilitate software inter-operations sunk to edge networks [40]. The functional and non-functional attributes of edge services are generally indifferent from traditional services. However, since edge services are sunk to edges of networks, they are highly mobile and dynamic in comparison to traditional services. The non-functional attributes of edge services are sensitive to user mobility and variations of server conditions and network environments [20].

3.3 Hidden Naive Bayes

Bayesian theorem is defined as the probability of event B occurring when event A occurs, and the probability of event A occurring when event B occurs is obtained. The formula can be expressed as follows:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

A Bayesian classifier bases on the Bayesian theorem, which combines between prior probability and class conditional density. Because of its high efficiency and simplicity, it is widely used in the classification problems of data mining. Its process is as follows: 1) for a specified sample to be classified, we calculate the probability of each category under the sample; 2) the sample belongs to the category with the maximum probability [41]. Let $C = \{c_0, c_1, ...c_j\}$ be a predefined set of categories and $X = \{x_1, x_2, ...x_n\}$ be a sample vector, $P(c_j|X)$ is the probability that X belongs to c_j, according to the Bayesian formula:

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

Assume that the attributes x_k and x_l of X are independent between each other, and $P(X)$ is same for all the categories, when X belongs to c_j. The Bayesian classifier formula can be simplified as follows:

$$C(X) = \underset{c_j \in C}{argmax} \left\{ P(c_j) \prod_{i=1}^{n} P(x_i|c_j) \right\} \tag{3}$$

where $C(X)$ is the category of X. It assumes that attributes are independent of each other in the Bayesian classifier, which ignores the dependency between the attribute values [42]. One way to improve the Bayesian classifier is to weaken the independence between attributes and to create a hidden parent attribute for each attribute, which represents the influence of the other attributes on an attribute. The improved Bayesian classifier structure is shown in Fig. 3. $\pi(x_i)$ is the hidden parent attribute of x_i, which represents the effect of the other attributes on x_i. The formula can be expressed as follows:

$$C(X) = \underset{c_j \in C}{argmax} \left\{ P(c_j) \prod_{i=1}^{n} P(x_i|\pi(x_i), c_j) \right\} \tag{4}$$

3.4 KNN

K-nearest neighbor (KNN) is a basic method for classification and regression [43], which has been used in many application areas of data mining, such as data classification and predictive analysis [44], [45], [46]. Its primary principle is, for a given instance, finding out the k nearest neighbors in

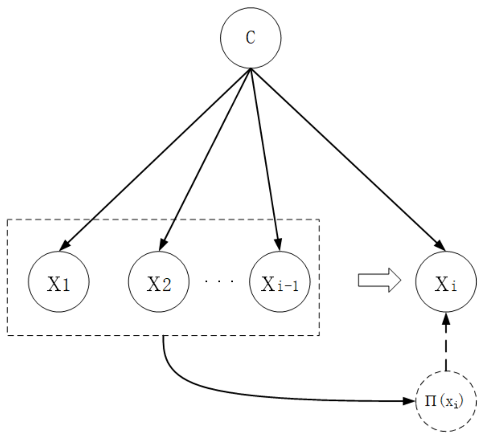


Fig. 3. Structural diagram of Hidden Naive Bayes

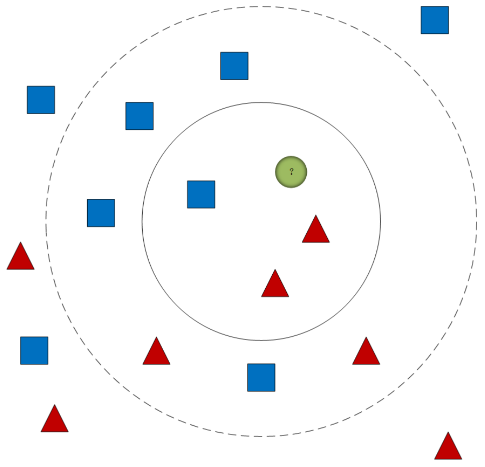


Fig. 4. The sample graph of KNN

the training set based on distance measurement and making predictions based on the information of the k nearest neighbors. Euclidean distance, Manhattan distance, and cosine distance can be used to measure the distance.

Generally, a"voting" mechanism can be used for the classification purpose. The labelled category which appears most frequently in the k nearest neighbors can be selected for prediction. For example, there are two categories of sample data in Fig. 4. One is square, and the other is triangle. The circular is the sample to be classified. If $k = 3$, there are two triangles and one square near the unlabelled sample. The unlabelled sample thus belongs to the category of triangle according to the "voting" result of the k nearest neighbors. When $k = 9$, the unlabelled sample belongs to the category of square. The regression usually results from averaging the values of the k nearest neighbours, with the purpose of numerical prediction of unlabelled samples. An improvement of these methods is weighted "voting" and average, where the weight measures the distance between a neighbor and the unlabelled sample. The closer a neighbor, the greater its weight.

4 The ghBSRM-MEC Approach

Section 4.1 provides an overview of ghBSRM-MEC. The detailed theoretical description of ghBSRM-MEC is given in Section 4.2. The algorithmic description of ghBSRM-MEC is introduced in Section 4.3.

4.1 Overview of ghBSRM-MEC

In MEC, due to the dynamic changes of user locations, monitoring results based on the historical data of old edge servers might become invalid for services in new edge servers. Inspired by the scenario that *Bob* navigated with Google Maps while driving, the ghBSRM-MEC approach is proposed for QoS monitoring in MEC. It can reduce monitoring errors caused by the dependence between QoS attributes, via constructing parent attributes among the QoS attributes. It is also capable of dynamically switching edge servers, and monitoring edge servers without historical data by employing the data of peripheral edge servers and the KNN algorithm. The main framework of ghBSRM-MEC is shown in Fig. 5. It primarily comprises the following three steps:

Step 1: Data collection and preprocessing. Data collection includes obtaining locations of edge servers and QoS data in each edge server. The locations of edge servers are used to determine the geographical distribution of edge servers. The QoS data of edge servers is used to construct corresponding classifiers for each edge server. Data preprocessing is mainly responsible for filtering invalid samples of edge servers. In addition, in our approach, synthetic sample data according to QoS requirements is generated as the validation samples in the experiment. In *Bob*'s example, the data we need to collect is the locations of the edge servers (e.g. *S1*, *S2* and so on.) invoked by *Bob*, and the historical QoS data(i.e. response times) of the Google Maps accessed by *Bob* and other users in those edge servers.

Step 2: Classifier construction. First, a probabilistic QoS requirement is defined according to a user's actual requirement (e.g. the probability that the service's response time is less than 3.6s is greater than 80%). Based on the historical data in each edge server, when a new QoS sample is added, whether the QoS sample meets the probabilistic QoS requirement is checked. By calculating the value of the parent attribute of the QoS attribute, the parameters of the model are obtained. Finally, the corresponding classifier is constructed. In *Bob'* example, the QoS standards are defined according to *Bob*'s requirements. Next, a Bayesian classifier is constructed by using the historical sample data of the edge servers near Bob's home and the small town, where the sample data records the response time of Google Maps accessed by *Bob* and other users on the same servers.

Step 3: Edge QoS monitoring. The ghBSRM-MEC approach performs QoS monitoring by judging whether the user moves to a new edge server and the condition of the server. If the user does not move out of the service range of the current server, this approach obtains the monitoring results by the classifier in the current edge server. If the user moves to a new edge server where there is historical data, the monitoring results are obtained according to the historical data of the new edge server. If the new edge server does not contain the historical data, the *k* nearest edge servers are selected to calculate the posterior probabilities, which are weighted by their distance away the new edge server, to obtain the monitoring results. In *Bob*'s example, when *Bob* departs from his home and calls the edge server *S1*, the monitoring results are derived from the historical data of *S1*. When *Bob* arrives at the edge server *S2* near the small town, if the edge server contains historical data, the monitoring results are derived from the historical data of *S2*. If *S2* does not contain historical data, this approach will employ the edge servers around *S2* to perform the monitoring.

4.2 Theoretical Description

The first and second steps of ghBSRM-MEC are described in Section 4.2.1. The third step of ghBSRM-MEC is presented in Section 4.2.2.

4.2.1 Data collection, preprocessing and classifier construction

The data collection phase aims to collect the location information of edge servers and the historical QoS data stored in edge servers. Edge server locations are used to find adjacent edge servers when there is no historical QoS data in the server accessed by a user. The historical data in edge servers is used to construct the Bayesian classifiers to make monitoring decisions when users invoke services from the edge servers. The data preprocessing intends to filter out invalid data, such as the response time of -1. In the experiment, a set of data is randomly generated according to the QoS requirements to verify the validity of our approach. For example, if a QoS requirement is the probability that a service's response time is less than 3.6s is greater than 80%, we will inject unsatisfied QoS monitoring samples into some sample sequences, the response time of which is out of the required range, e.g. [3.6,500].

We begin to construct the Bayesian classifier for each monitored service in each edge server with historical data. We define $X = \{x_1, x_2, \cdots x_n\}$ as a QoS sample vector for a service, where $x_k (k \in [1, n])$ is the value of the k-th QoS attribute of the service, such as response time. $C = \{c_0, c_1\}$ is a predefined set of categories, where the samples that satisfy the probabilistic QoS requirement belong to c_0; otherwise they belong to c_1.

When a Bayesian classifier is used to classify the continuous QoS values, it usually assumes that the continuous variables obey certain probabilistic distribution. The training data is used to estimate the parameters of the distribution. Gaussian distribution is commonly used to represent the class conditional probability distribution of continuous variables [47].

For formula 4, the value of $P(c_j)$ usually bases on the maximum likelihood estimate of the sample, i.e., $P(c_j) = n_{c_j}/n$, where n_{c_j} indicates the number of samples belonging to c_j in the sample set of a QoS requirement, and n represents the total number of samples in the sample set. When a new service is monitored, the classifier needs to check whether or not it meets the probabilistic QoS requirement. Under the assumption of Gaussian distribution [48], the measurement of the probability can be realized by the probability density integral formula: $P(X{<}QoS\_Value) = \int_{-\infty}^{QoS\_Value} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-u)^2}{2\sigma^2}}$. For example, a QoS requirement can be defined as: "the probability that a service's response time is less than 3.7s is greater than 85%". The QoS_Value

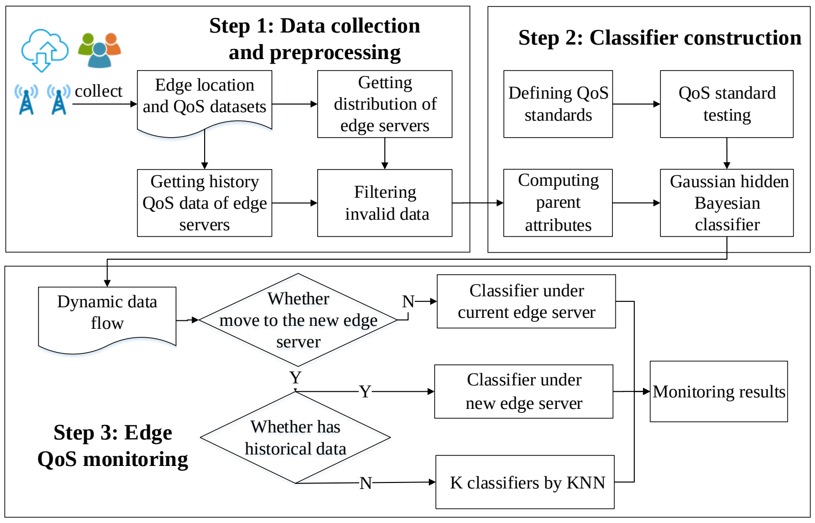

Step 1: Data collection and preprocessing
collect
Edge location and QoS datasets
Getting distribution of edge servers
Getting history QoS data of edge servers
Filtering invalid data
Step 2: Classifier construction
Defining QoS standards
QoS standard testing
Computing parent attributes
Gaussian hidden Bayesian classifier
Dynamic data flow
Whether move to the new edge server
N
Y
Classifier under current edge server
Classifier under new edge server
Whether has historical data
K classifiers by KNN
Monitoring results
Step 3: Edge QoS monitoring

Fig. 5. Overall structure diagram of ghBSRM-MEC

is 3.7, u represents the sample mean, and σ represents the sample standard deviation. Given a sample response time attribute x_k ($k \in [1, n]$), the value of its parent attribute is the mean of $x_1 \sim x_{k-1}$. The prior conditional probability is derived from formula 5, where $N_{c_j}()$ denotes the Gaussian distribution under the constraints of the class c_j, u_{x_i} and $\sigma^2_{x_i}$ are the mean and variance of the sample attribute x_i in the class c_j, and $u_{\pi(x_i)}$ and $\sigma^2_{\pi(x_i)}$ are the mean and variance of the parent attribute of x_i, i.e. $\pi(x_i)$, and $\rho = \frac{COV(x_i,\pi(x_i))}{\sigma_{x_i}\sigma_{\pi}(x_i)}$ is the correlation coefficient between x_i and $\pi(x_i)$.

4.2.2 *Edge QoS monitoring*

As shown in Fig. 6, it is assumed that there is a set of edge servers that provide services to active users in their service ranges. When a service in a server invoked by a user is monitored, the locations of the server and the user and the existence of the historical data must be taken into account. Considering the mobility of users, the monitoring is based on the three situations defined previously.

- *Case 1*: The user stays in the service range of the same edge server when the service is invoked, such as *User 1* and *User 2* in Fig. 6. Here the monitoring result is obtained by the classifier of the current server (*Edge 1*).
- *Case 2*: The user moves from the service range of one edge server to another, where the new edge server contains relevant historical QoS data, such as *User 4* in Fig. 6. In this situation, we monitor the QoS value according to the data in the new edge server.
- *Case 3*: The user moves from the service range of one edge server to another, where the new edge server does not retain relevant historical QoS data, such as *User 5* in Fig. 6. Under this circumstance, we rely on the KNN algorithm. First, we find k nearest edge servers around the invoked edge server. The distances between these edge servers are represented as $d = \{d_1, d_2, ...d_k\}$. The posterior probability of the currently monitored sample in the k edge servers is represented as $pro = \{p_1, p_2, ...p_k\}$. The posterior probability of the invoked edge server is calculated as: $afterpro = w_1 * p_2 + w_2 * p_2 + ...w_k * p_k$, where $w_1 + w_2 + ...w_k = 1$ and $w_k = x/d_k$, where x is the scale factor of w_k and d_k. The closer the distance, the greater the weight. The monitoring results can therefore be obtained according to the posterior probability.

4.3 Algorithmic Description

The detailed algorithms of the ghBSRM-MEC approach are described in this section. Algorithm 1 is a general description of ghBSRM-MEC, which is divided into a training stage and a monitoring stage. During the training stage, a Gaussian hidden Bayesian classifier is constructed for each edge server by using historical data (lines 1-4). During the monitoring stage, three situations are considered to calculate the posterior probability to obtain the monitoring results.

- *Case 1*: The posterior probability is calculated by the classifier in the current edge server (lines 6-7).
- *Case 2*: The posterior probability is calculated by the classifier in the new edge server (lines 8-10).
- *Case 3*: The k nearest peripheral edge servers are first selected (line 12). The posterior probability is calculated separately (line 13). The corresponding weight is calculated (line 14). The posterior probability is computed (line 15).

Finally, according to the posterior probability ratio, the monitoring result is obtained. A ratio greater than 1 indicates the probabilistic QoS criterion being satisfied, i.e., the monitoring result belongs to c_0 (lines 19-20). A ratio less than 1 indicates the criterion being unsatisfied, i.e., the monitoring result belongs to c_1 (lines 21-22). A ratio equivalent to 1 means in-decisive (lines 23-24).

Algorithm 2 and Algorithm 3 realize two parts of Algorithm 1. Algorithm 2 calculates the parameters of the

$$
\begin{aligned}
P(x_i|\pi(x_i),c_j) &= \frac{P(x_i,\pi(x_i)|c_j)}{P(\pi(x_i)|c_j)} \\
&= \frac{\frac{1}{2\pi\sigma_{x_i}\sigma_{\pi(x_i)}\sqrt{1-\rho^2}} exp\{-\frac{1}{2(1-\rho^2)}[\frac{(x_i-u_{x_i})^2}{\sigma_{x_i}^2} - 2\rho\frac{(x_i-u_{x_i})(\pi(x_i)-u_{\pi(x_i)})}{\sigma_{x_i}\sigma_{\pi(x_i)}} + \frac{(\pi(x_i)-u_{\pi(x_i)})^2}{\sigma_{\pi(x_i)}^2}]\}}{\frac{1}{\sqrt{2\pi}\sigma_{\pi(x_i)}} exp\{-\frac{(\pi(x_i)-u_{\pi(x_i)})^2}{2\sigma_{\pi(x_i)}^2}\}} \\
&= \frac{1}{\sqrt{2\pi}\sigma_{x_i}\sqrt{1-\rho^2}} exp\{-\frac{1}{2(1-\rho^2)}(\frac{x_i-u_{x_i}}{\sigma_{x_i}} - \rho\frac{\pi(x_i)-u_{\pi(x_i)}}{\sigma_{\pi(x_i)}})^2\} \\
&= \frac{1}{\sqrt{2\pi}\sigma_{x_i}\sqrt{1-\rho^2}} exp\{-\frac{1}{2\sigma_{x_i}^2(1-\rho^2)}[x-(u_{x_i}+\rho\frac{\sigma_{x_i}}{\sigma_{\pi(x_i)}}(\pi(x_i)-u_{\pi(x_i)}))]^2\} \\
&= N_{c_j}(u_{x_i}+\rho\frac{\sigma_{x_i}}{\sigma_{\pi(x_i)}}(\pi(x_i)-u_{\pi(x_i)}), \sigma_{x_i}^2(1-\rho^2))
\end{aligned} \tag{5}
$$

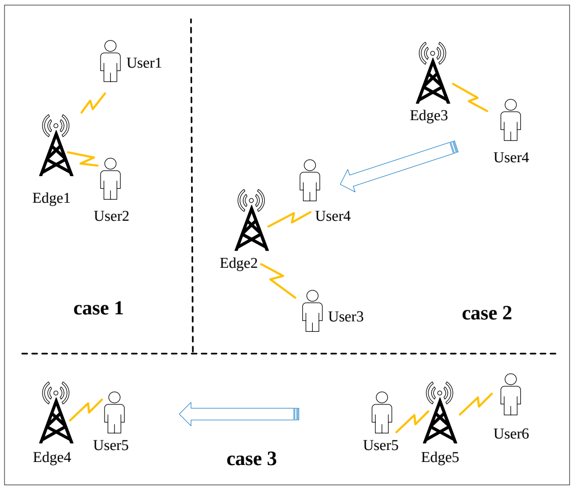


Fig. 6. A case of mobile edge monitoring

Bayesian classifier during the training stage. The new training samples are added in line 2. The value of the parent attribute is calculated in line 3. The probabilistic requirement test is carried out in lines 4-13. The prior probability is calculated in lines 14-15. The other parameters are calculated in line 16. Algorithm 3 calculates the posterior probability during the monitoring stage. The two types of conditional probability, $condition\_proc_0$ and $condition\_proc_1$ are calculated in lines 2-3. The two types of posterior probability, $afterProc_0$ and $afterProc_1$ are calculated in lines 4-5 and the posterior probability is calculated in line 6. Algorithm 4 describes the test of the current samples over the probabilistic requirement. The mean and variance are calculated in lines 1-2. The probability that the current samples satisfy the threshold is measured in line 3. It is compared with the predefined probability to obtain the category of the current samples (lines 4-7).

Algorithm 1 edgeMonitoring

```
Require: The sample stream of each edge server S; QoS
    probabilistic standard $\beta$; QoS threshold $QoS\_Value$;
Ensure: Monitoring result $c_j$;
 1: if training stage then
 2:     for each edge do
 3:         ghBayesian(S, $\beta$, QoS_Value)
 4:     end for
 5: else
 6:     if isMove==False then
 7:         afterpro=computerAftPro($x_k$, oldedge)
 8:     else
 9:         if isEmpty==False then
10:             afterpro =computerAftPro($x_k$, newedge)
11:         else
12:             $Top_k$(nearestedge)
13:             pro[k]=computerAftPro($x_k$, edge[k])
14:             w[k]=computerWi(d[k])
15:             afterpro=w[1]*pro[1]+ w[2]*pro[2]+...w[k]*pro[k]
16:         end if
17:     end if
18: end if
19: if afterpro> 1 then
20:     return $c_0$
21: else if afterpro<1 then
22:     return $c_1$
23: else
24:     return indecisive
25: end if
```

5 EXPERIMENTS

This section validates ghBSRM-MEC through a mixture of simulated and real-world QoS data sets in mobile edge computing environments. The experiments are conducted upon the Geany [3] development platform. Python is utilized to design and implement ghBSRM-MEC. All the other environmental parameters are described in Table 1. Section 5.1 outlines our research questions. Section 5.2 describes the detailed data sets we used. Section 5.3 introduces the process of data set fusion in detail. Section 5.4 provides a brief introduction of the existing QoS monitoring approaches to compare with ghBSRM-MEC. Finally, Section 5.5 discusses the experimental results.

5.1 Research Questions

Our experiments attempt to answer the following research questions:

- *RQ1*: Is ghBSRM-MEC feasible for QoS monitoring in mobile edge computing?

[3] https://www.geany.org/

```
Algorithm 2 ghBayesian
Require: The training samples $T = \{x_1, x_2, ...x_n\}$; QoS probabilistic requirement $\beta$; QoS threshold $QoS\_Value$;
Ensure: Prior probability;
 1: while $x_k \in T$ do
 2:     n++
 3:     PI$x_k$=computeParent($x_k$)
 4:     if requirementDecision($x_k$)==$c_0$ then
 5:         $n_{c_0}$++
 6:         $c_0$.append($x_k$)
 7:         PI$c_0$.append(PI$x_k$)
 8:     else
 9:         $n_{c_1}$++
10:         $c_1$.append($x_k$)
11:         PI$c_1$.append(PI$x_k$)
12:     end if
13: end while
14: pro_$c_0$ = $n_{c_0}$/n
15: pro_$c_1$ = $n_{c_1}$/n
16: $compute(uc_jx, \sigma c_jx, uc_jPIx, \sigma c_jPIx, \rho)$
```

```
Algorithm 3 computeAftPro
Require: The monitoring samples $S = \{x_1, x_2, ...x_n\}$;
Ensure: Posterior probability;
 1: while $x_k \in S$ do
 2:     $condition\_proc_0 += computeConProc_0$
        $(x_k, uc_0x, \sigma c_0x, uc_0PIx, \sigma c_0PIx, \rho)$
 3:     $condition\_proc_1 += computeConProc_1$
        $(x_k, uc_1x, \sigma c_1x, uc_1PIx, \sigma c_1PIx, \rho)$
 4:     $afterProc_0 = log(pro\_c_0) + condition\_proc_0$
 5:     $afterProc_1 = log(pro\_c_1) + condition\_proc_1$
 6:     pro=afterProc$_0$/afterProc$_1$
 7:     return pro
 8: end while
```

TABLE 1
Experimental environment parameter

Configuration item	Experimental environment parameter
RAM	4GB
CPU	Intel Core i5-4200M 2.5GHz
Hard disk	5400 rpm HD
System	Windows 10

```
Algorithm 4 requirementDecision
Require: The sample stream of each edge server S; QoS probabilistic standard $\beta$; QoS threshold $QoS\_Value$;
Ensure: classification result $c_j$;
 1: u=mean(S)
 2: $\sigma$=std(S)
 3: $c = \int_{-\infty}^{QoS\_Value} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-u)^2}{2\sigma^2}}$
 4: if c>=$\beta$ then
 5:     return $c_0$
 6: else
 7:     return $c_1$
 8: end if
```

- *RQ2*: Is ghBSRM-MEC more effective and efficient than the state-of-the-art QoS monitoring approaches?
- *RQ3*: Is ghBSRM-MEC capable of handling the challenge of user mobility in mobile edge computing?
- *RQ4*: How does Top-k edge servers affect the monitoring results?

We designed *RQ1* to validate the feasibility of ghBSRM-MEC. *RQ2* is used to analyze whether ghBSRM-MEC is more effective and efficient than traditional QoS monitoring approaches. *RQ3* is used to validate the capability of ghBSRM-MEC on handling user mobility. We execute ghBSRM-MEC in two different environments. *RQ4* is used to test the impact of k on monitoring performance.

5.2 Data Set Description

As shown in Table 2, three data sets are used. The first data set [1] is *Shanghai Telecom* data set [18] [49]. This data set includes the locations of 3233 base stations and IDs of users who called services from the base stations. We select some base stations to simulate the mobile edge monitoring environment. The second data set [2] uses the real-world Web QoS data set published by the Chinese University of Hong Kong (CUHK). This data set records the response time, throughput and locations of 5825 real-world services called by 339 users. The third data set is a data set that injects randomly generated unsatisfied QoS monitoring samples into the fused *Shanghai Telecom CUHK* data sets (The details of data fusion can be found in Section 5.3). This data set can be employed to validate ghBSRM-MEC.

5.3 Data Set Fusion

We use the idea of data fusion to construct the edge environment and validate the proposed approach. QoS data sets in mobile edge environments are stored in geographically distributed repositories. In contrast, the existing QoS data sets are mostly obtained from centralized repositories. Thus we need to create a new QoS data set for mobile edge environments. We respectively employ the base station locations from the *Shanghai Telecom* data set and QoS data published by CUHK. We merge the two data sets to generate a data set that satisfies the edge characteristics.

First, services from the QoS data set in the same locations are considered to be in the same groups and the collected QoS data is grouped accordingly. Second, some base stations are randomly selected to form the geographical distribution of edge servers, Finally, an edge environment is constructed by determining edge locations and assigning the grouped sample QoS data to corresponding edge servers. The detailed steps are:

- **Step 1: QoS data grouping.** The QoS data is grouped according to the locations of their corresponding services. The services with the same latitude and longitude values are viewed to be in the same edge server. In this way, we can get a set of grouped QoS data. For example, a service set from the *CUHK* data set is formed according to the following latitude and longitude coordinates: (-10.0, -55.0). This service set contains 21 services. That is to say, these services reside in the same edge server. Here shows an example

TABLE 2
Information of data sets

Date sets	Type	Description
1	Public	3233 locations of base stations and user ID
2	Public	5825 real-world Web services used by 339 distributed users
3	Private	Synthetic data set that injects unsatisfied QoS monitoring samples according to QoS requirements

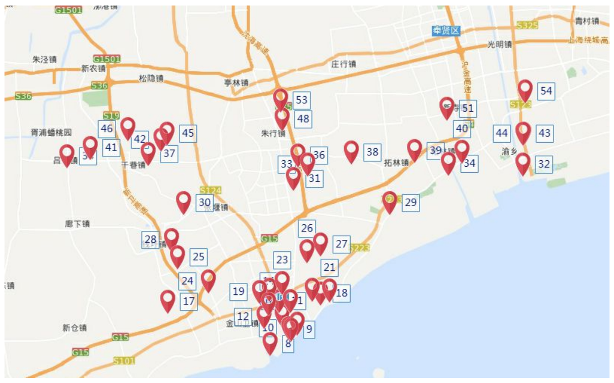


Fig. 7. Edge Server Distribution Map

how QoS data of the services is grouped. The original CUHK QoS data set can be represented as:

$RespTime$	$Serv_1$	$Serv_2$	$Serv_3$	...	$Serv_n$
$User_1$	5.982	0.228	0.237	...	6.777
$User_2$	2.13	0.262	0.273	...	0.263
$User_3$	0.854	0.366	0.376	...	0.173
$User_4$	0.693	0.226	0.233	...	0.095
...	...	...	...	...	
$User_m$	1.285	0.222	0.232	...	0.13

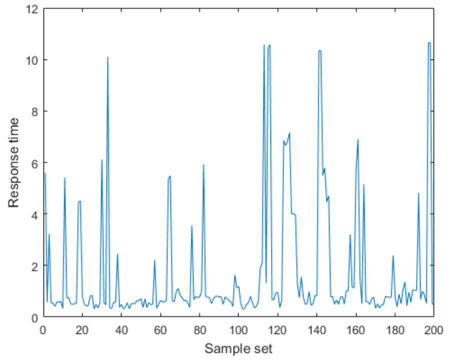
If $Serv_1$ and $Serv_n$ have the same latitude and longitude, these two services can be assumed to reside in the same edge server, e.g. S_1. We group the QoS values of these two services into a sample vector $x_1, x_2, \cdots x_{2m}$.

$RespTime$	x_1	...	x_m	x_{m+1}	...	x_{2m}
S_1	5.982	...	1.285	6.777	...	0.13

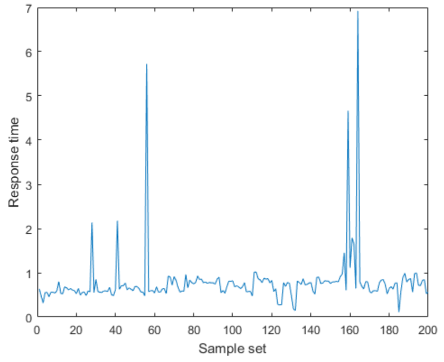
- **Step 2: Edge location selection.** 60 base station locations are randomly selected from the *Shanghai Telecom* data set. Their geographical distribution is shown in Fig. 7.
- **Step 3: Edge data set formation.** We randomly assign the QoS data groups obtained from step 1 to the edge server locations collected in step 2. Therefore, a mobile edge environment is constructed. Fig. 8 shows the response time of edge server IDs of 1, 20 and 36, respectively. We can see that the response time attribute data in the edge servers is all continuous data, which satisfies the premise that the ghBSRM-MEC approach is built on the assumption that the QoS data follows Gaussian distribution.

5.4 Compared Approaches

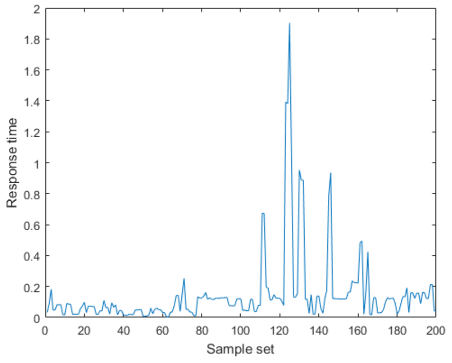
We introduce the following approaches to compare with ghBSRM-MEC. They are the most effective QoS monitoring



(a)



(b)



(c)

Fig. 8. Some QoS data of different edge servers: (a) edge1, (b) edge20, and (c) edge36.

approaches in recent years.

- iBSRM [15]: The first probabilistic QoS monitoring approach based on Bayesian statistics. By calculating the Bayesian factor, monitoring decisions can be made based on hypothesis testing. Previous monitoring results can be reused to achieve continuous monitoring.
- wBSRM [16]: A probabilistic monitoring approach considering the impact of environmental factors on QoS. TF-IDF (term frequency-inverse document frequency) is used to quantify the impact of the environmental factors. The quantitative value is used to weigh the samples. Historical data can be trained to construct a weighted Bayesian classifier for monitoring and decision-making.
- IgS-wBSRM [17]: An approach extended from wBSRM. This approach can dynamically update the early initialized weight by calculating information gain and combining it with a sliding window mechanism.

5.5 Experimental Results

We conduct a series of experiments on all the data sets to investigate the capability of ghBSRM-MEC on feasibility, effectiveness, efficiency, and user mobility handling, as well as the impact of k on the performance of ghBSRM-MEC. Since all the edge servers share similar features, in each part of the experiment, we show the results on some randomly selected edge servers.

5.5.1 Feasibility

The first group of experiments is designed to validate the feasibility (*RQ1*) of ghBSRM-MEC. We use the real data sets (i.e. fused data sets 1 and 2) to test the monitoring performance under different QoS requirements in different edge servers.

Three edge servers with IDs of 1, 10 and 20 from the 60 edge servers are selected to demonstrate the feasibility. First, the first 2,000 QoS samples in each server are used to train a Gaussian hidden Bayesian classifier. The remaining 3,000 QoS samples are employed for monitoring validation. The posterior probability greater than 1 represents that the monitoring results is in c_0, i.e. meeting the QoS requirement. If the posterior probability is less than 1, the monitoring results fall into c_1, which do not meet the QoS requirement.

Fig. 9 shows the simultaneous monitoring results of the three edge servers. They subsequently correspond to the following three QoS requirements, "the probability of response time less than 2.5 is greater than 85%", "the probability of response time less than 2.5 is greater than 80%" and "the probability of response time less than 1.9 is greater than 85%". If the monitoring result satisfies the QoS requirement, the monitoring result is "1"; otherwise it is "-1". If the results cannot be judged, it is "0". The vertical lines indicate changes in the monitoring status. As can be seen from the figure, service failure is detected at 4-388 samples in edge server 1, service failure is detected at 35-1268 in edge server 10, and more frequent service failure is detected in edge server 20. In mobile edge environments, the data among edge servers is independent. Services from different

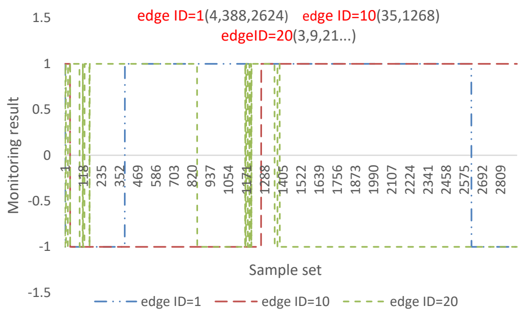


Fig. 9. Monitoring results for different edge servers

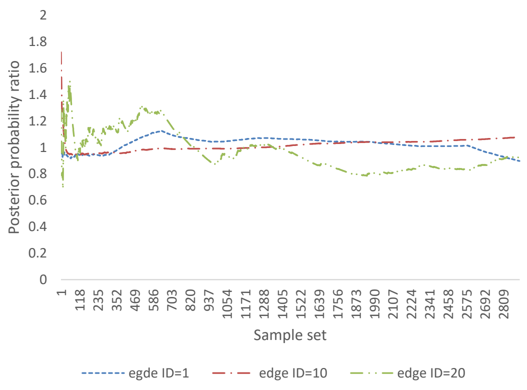


Fig. 10. Posterior probability of different edge servers

edge servers can be independently monitored under their corresponding probabilistic criteria. Fig. 10 shows the ratios of the posterior probability that satisfy the probabilistic QoS criteria and the ones that do not meet the probabilistic criteria. A ratio greater than 1 indicates the probabilistic QoS criteria being satisfied. A ratio less than 1 indicates the criteria being unsatisfied. A ratio equivalent to 1 means indecisive. Because the probability is continuously changing, the monitoring results can be observed more intuitively.

5.5.2 Effectiveness and Efficiency

We conduct the second group of experiments to validate the effectiveness and efficiency (*RQ2*) of ghBSRM-MEC. We use the synthetic data sets to compare the performance of ghBSRM-MEC with the existing approaches: IgS-wBSRM, wBSRM, and iBSRM, under the same QoS requirement and in the same edge server.

The edge server with ID 1 is randomly selected to run these approaches, where the QoS requirement is described as "the probability of response time less than 3.6s is greater than 85%". The first 2,000 QoS samples from the real data set are utilized for training. Among the 3,000 monitoring samples, a number of unsatisfied samples with response time greater than 3.6s are injected in the samples numbered between 1 and 500 and between 1,000 and 2,000 with the probability of more than 15%. The monitoring results are shown in Fig. 11. ghBSRM-MEC first detects the injected service failure when the sample number reaches 106. IgS-wBSRM is the 2nd fastest. wBSRM and iBSRM cannot detect the service failure at this time. In the second phase (i.e.

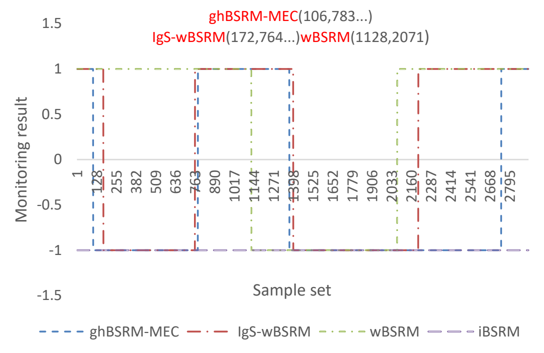


Fig. 11. Monitoring results of compared approaches

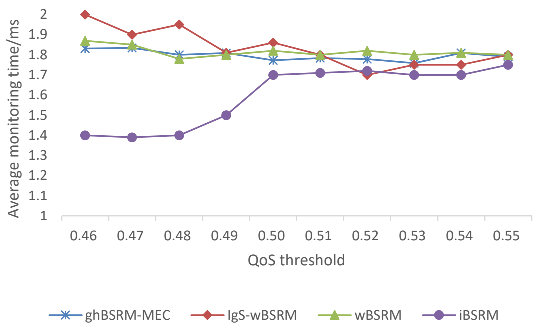


Fig. 12. Average monitoring time of compared approaches

1,000-2,000), ghBSRM-MEC still detects the service failure faster than IgS-wBSRM. wBSRM cannot detect the previous service failure until this phase. iBSRM still maintains the misjudgment. In general, we can see that ghBSRM-MEC's monitoring results are generally consistent with the injected unsatisfied samples. During the process of Bayesian classifier construction, parent attributes are taken into account to reduce the dependence between attribute values, which can be used to make more effective decision-making.

We analyze the efficiency of these approaches in terms of their *training time* and *monitoring time*. Based on the aforementioned probabilistic requirements, ghBSRM-MEC spends about 2.17s on training the 2,000 samples, which is slightly longer than wBSRM. This is because integral calculation needs to be performed many times when parameters are calculated during the training stage. During the monitoring stage, the time required for each monitoring approach to complete 3,000 monitoring samples under different QoS requirements is recorded, and the average monitoring time per sample is obtained. The execution time of the monitoring approaches under the same QoS requirement can effectively reflect their efficiency. This experiment is run in the environment described in Table 1, and the experimental results only represent their performance in this specific experimental environment. However, the relative operating efficiency among these approaches is believed to keep unchanged. We obtain the execution time of the monitoring algorithms under the same QoS requirements as shown in Fig. 12. In general, the average monitoring time of ghBSRM-MEC is no better than iBSRM, which is close to wBSRM and better than IgS-wBSRM.

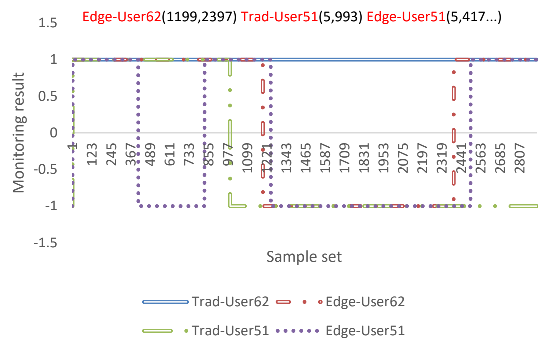


Fig. 13. Response time monitoring results in different environments

5.5.3 User Mobility Handling

To answer *RQ3*, we compare the monitoring results using ghBSRM-MEC in both the traditional environment and the mobile edge environment. Table 3 and Table 4 show the sequences of the edge servers, from which *user 51* and *user 62* call a service when they are moving.

Fig. 13 respectively shows the monitoring results of the two users in the traditional and mobile edge environments. The QoS requirement is described as ”the probability of response time less than 3.6s is greater than 85%”. In this figure, *Trad-User62* represents the monitoring results of *user 62* in the traditional environment, i.e., the monitoring results relying on the historical data of edge server 18. *Edge-User62* represents the monitoring results of *user 62* in the mobile edge environment, which is, the monitoring results based on the historical data of a series of edge servers called by *user 62*, i.e., edge server 18, edge server 21, edge server 26, edge server 27 and edge server 55. In the mobile edge environment, *user 62* detects the service failure when the sample number reaches 1,199. It can be presumed that the monitoring results vary because the user switches the servers due to the mobility. However, in the traditional environment, the monitoring results is maintained in a successful state due to the fact that only the historical data from the previous edge servers is used. Similarly, the monitoring results are also different in the two environments for *user 51*.

In addition, we monitor the second QoS attribute in the dataset, i.e. throughput. The QoS requirement is described as ”the probability of throughput more than 9.8 is greater than 80%”. The monitoring results of the two users in the traditional and mobile edge environments are shown in Fig. 14. Similarly, in the mobile edge environment, *user 51* detects the service success when the sample number reaches 833. However, in the traditional environment, the monitoring results is maintained in an unsuccessful state.

The experimental results show that edge server switching caused by user mobility may lead to the misuse of previous historical data and deviation in the monitoring results. This effectively proves the capability of the ghBSRM-MEC approach handling the user mobility challenge in the mobile edge environment.

5.5.4 Impact of k

The last group of experiments is designed to answer *RQ4*. We validate the effect of k in *Case 3*. When a user moves

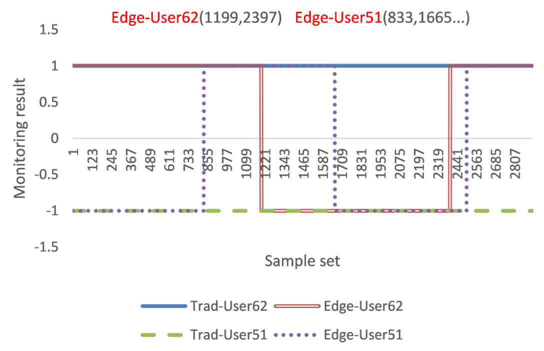


Fig. 14. Throughput monitoring results in different environments

TABLE 3
Edge servers called by user 51

User ID	51						
Edge Server ID	9	12	14	16	17	22	55

to an edge server without relevant historical data, the KNN algorithm is used to select the peripheral edge servers to perform the monitoring task. The experiments on randomly selected edge servers 1 and 25 show that the monitoring based on the k-nearest edge servers is feasible, when local edge servers do not contain relevant historical data. To reasonably and effectively obtain the appropriate k, our experiments predefine k in a broader range, and then narrow it down to an appropriate value according to the performance of k.

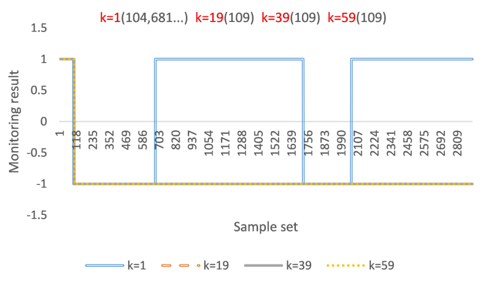
The following experiments are performed over the two edge servers. This group of experiments still bases on the synthetic data set. The QoS requirement is described as ”the probability of response time less than 2.8s is greater than 85%”. Similarly, in the ranges of 1-500 and 1,000-2,000 of the sample set, we inject more than 15% negative samples, in which the response time is greater than 2.8s.

Fig. 15 shows the monitoring results of edge server 1 on different values of k. Because there are only 60 edge servers in our experiment, we search the appropriate number of k in the whole range. Fig. 15(a) shows the monitoring results when $k = 1, 19, 39, 59$. As can be seen from the figure, when $k > 19$, the monitoring result no longer changes after successfully detecting service failure in the first phase. Consequently, we can conclude that the appropriate range of k is 1∼19. Fig. 15(b) shows the monitoring results when $k = 1, 5, 9, 14$. We find that when $k = 5$, the monitoring result is consistent with the appearance of negative samples in the first phase. However, the service failure in the second phase is still not detected. In Fig. 15(c), we test the monitoring results at $k = 1, 2, 3, 4, 5$, respectively. The results show that when $k = 1$ and $k = 2$ the proposed approach can detect service failure in both of the phases. When $k = 1$, the monitoring result is most accurate.

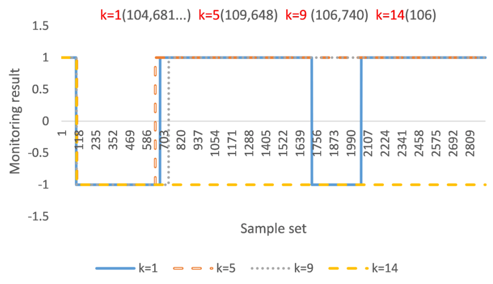
To further verify the generality of our approach, we select edge server 25 to perform the monitoring task. Similarly, the first step is to narrow down the range of k. As shown in Fig. 16(b), the monitoring results are not satisfied when

TABLE 4
Edge servers called by user 62

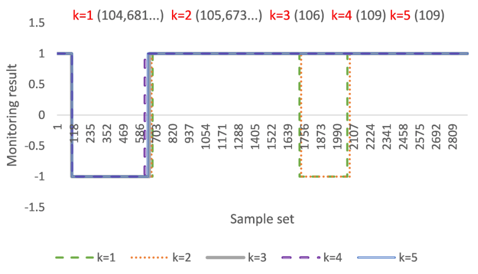
User ID	62				
Edge Server ID	18	21	26	27	55



(a)



(b)



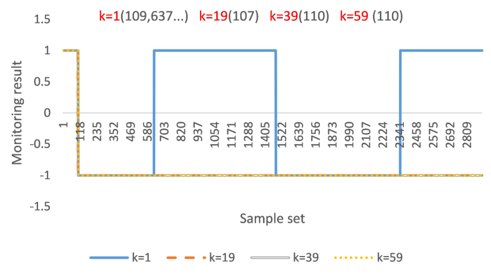
(c)

Fig. 15. Effect of K for edge server 1 in different ranges: (a) k=1∼59, (b) k=1∼14 , (c) k=1∼5.

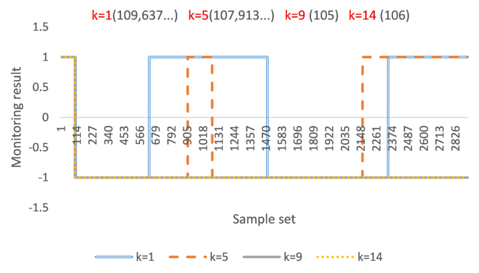
k>19. Then we narrow down the range of k to 1∼19. The monitoring results are shown in Fig. 16(c). When $k > 9$, the monitoring results are still unsatisfied. Finally, we test the monitoring results on $k = 1, 3, 5, 7, 9$, and get the best result when $k = 3$, as shown in Fig. 16(a). In general, the experimental results show that the value of k has an impact on the monitoring results. On the other hand, the optimal k is also different on different edge servers.

6 Conclusions and Prospects

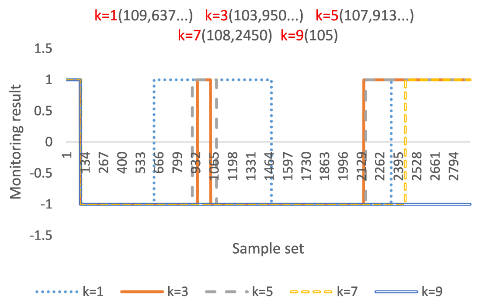
Traditional QoS monitoring approaches do not consider user mobility and data dependency, which might lead to deviation in monitoring results. In this paper, we present ghBSRM-MEC, a novel QoS monitoring based on Gaussian hidden Bayesian classification in the mobile edge environ-



(a)



(b)



(c)

Fig. 16. Effect of K for edge server 25 in different ranges: (a) k=1∼59, (b) k=1∼14, (c) k=1∼9.

ment. The experiments on real and simulated data sets show that the proposed method is feasible and effective.

For future work, the following directions will be considered: First, we will further explore the impact of k on monitoring performance when there is no relevant historical data in monitored servers. Second, since multivariate QoS attributes [50] may contain conflicts, we will consider how to monitor multivariate QoS attributes simultaneously in mobile edge environments.

7 Acknowledgements

The work is supported by the National Natural Science Foundation of China (61572171,61702159), the Natural Science Foundation of Jiangsu Province (BK20191297), the Fundamental Research Funds for the Central Universities of China (B210202075) and the Open Foundation of State key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) (SKLNST-2020-1-07).

References

[1] 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.
[2] 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.
[3] M. Urkia Kortabarria, "Web Service Performance on Heterogeneous Systems: A performance comparison between J2EE and. NET on heterogeneous systems," 2013.
[4] H. Billhardt, R. Hermoso, S. Ossowski, and R. Centeno, "Trust-based service provider selection in open environments," in *Proceedings of the 2007 ACM symposium on Applied computing*, pp. 1375–1380, ACM, 2007.
[5] J. El Haddad, M. Manouvrier, and M. Rukoz, "TQoS: Transactional and QoS-aware selection algorithm for automatic Web service composition," *IEEE Transactions on Services Computing*, no. 1, pp. 73–85, 2010.
[6] D. Gunter, B. Tierney, K. Jackson, J. Lee, and M. Stoufer, "Dynamic monitoring of high-performance distributed applications," in *Proceedings 11th IEEE International Symposium on High Performance Distributed Computing*, pp. 163–170, IEEE, 2002.
[7] A. M. Daniel and T. Menasc, "QoS issues in web services," *IEEE Internet Computing*, vol. 6, no. 6, pp. 72–75, 2002.
[8] L. Baresi and S. Guinea, "Towards dynamic monitoring of WS-BPEL processes," in *International Conference on Service-Oriented Computing*, pp. 269–282, Springer, 2005.
[9] M. Leucker and C. Schallhart, "A brief account of runtime verification," *The Journal of Logic and Algebraic Programming*, vol. 78, no. 5, pp. 293–303, 2009.
[10] L. Grunske, "Specification patterns for probabilistic quality properties," in *2008 ACM/IEEE 30th International Conference on Software Engineering*, pp. 31–40, IEEE, 2008.
[11] 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.
[12] 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.
[13] 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.
[14] L. Grunske, "An effective sequential statistical test for probabilistic monitoring," *Information and Software Technology*, vol. 53, no. 3, pp. 190–199, 2011.
[15] 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.
[16] 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.
[17] 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.
[18] 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.
[19] Y. Xu, J. Yin, S. Deng, N. N. Xiong, and J. Huang, "Context-aware QoS prediction for web service recommendation and selection," *Expert Systems with Applications*, vol. 53, pp. 75–86, 2016.
[20] Z. Liu, Q. 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, DOI: 10.1109/TSC.2019.2944596.
[21] 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.
[22] S. Radovanović, N. Nemet, M. Ćetković, M. Z. Bjelica, and N. Teslić, "Cloud-based framework for QoS monitoring and provisioning in consumer devices," in *2013 IEEE Third International Conference on Consumer Electronics Berlin (ICCE-Berlin)*, pp. 1–3, IEEE, 2013.

[23] H. Rachidi and A. Karmouch, "A framework for self-configuring devices using TR-069," in *2011 International Conference on Multimedia Computing and Systems*, pp. 1–6, IEEE, 2011.

[24] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar, "Comprehensive QoS monitoring of Web services and event-based SLA violation detection," in *Proceedings of the 4th international workshop on middleware for service oriented computing*, pp. 1–6, ACM, 2009.

[25] L. Coppolino, S. D'Antonio, L. Romano, F. Aisopos, and K. Tserpes, "Effective QoS monitoring in large scale social networks," in *Intelligent Distributed Computing VII*, pp. 249–259, Springer, 2014.

[26] 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.

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

[28] 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, DOI: 10.1109/TSC.2019.2952604.

[29] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge computing based on markov decision process," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1272–1288, 2019.

[30] D. Zhao, T. Yang, Y. Jin, and Y. Xu, "A service migration strategy based on multiple attribute decision in mobile edge computing," in *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, pp. 986–990, IEEE, 2017.

[31] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, "QoS prediction for service recommendation with deep feature learning in edge computing environment," *Mobile Networks and Applications*, pp. 1–11, 2019.

[32] P. Zhang, Y. Zhang, H. Dong, and H. Jin, "Multivariate qos monitoring in mobile edge computing based on bayesian classifier and rough set," in *2020 IEEE International Conference on Web Services (ICWS)*, pp. 189–196, IEEE, 2020.

[33] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[34] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A berkeley view of cloud computing," *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, vol. 28, no. 13, p. 2009, 2009.

[35] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 37–42, 2015.

[36] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, pp. 20–26, IEEE, 2016.

[37] B. I. Ismail, E. M. Goortani, M. B. Ab Karim, W. M. Tat, S. Setapa, J. Y. Luke, and O. H. Hoe, "Evaluation of docker as edge computing platform," in *2015 IEEE Conference on Open Systems (ICOS)*, pp. 130–135, IEEE, 2015.

[38] A. E. Youssef, "Exploring cloud computing services and applications," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 6, pp. 838–847, 2012.

[39] A. F. Huang, C.-W. Lan, and S. J. Yang, "An optimal qos-based web service selection scheme," *Information Sciences*, vol. 179, no. 19, pp. 3309–3322, 2009.

[40] X. Zhao, Y. Shi, and S. Chen, "Maesp: Mobility aware edge service placement in mobile edge networks," *Computer Networks*, vol. 182, p. 107435, 2020.

[41] G. E. Box and G. C. Tiao, *Bayesian inference in statistical analysis*, vol. 40. John Wiley & Sons, 2011.

[42] H. Zhang, L. Jiang, and J. Su, "Hidden naive bayes," in *American Association for Artificial Intelligence(Aaai)*, pp. 919–924, 2005.

[43] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN model-based approach in classification," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pp. 986–996, Springer, 2003.

[44] S. B. Imandoust and M. Bolandraftar, "Application of k-nearest neighbor (KNN) approach for predicting economic events: Theoretical background," *International Journal of Engineering Research and Applications*, vol. 3, no. 5, pp. 605–610, 2013.

[45] S. Begum, D. Chakraborty, and R. Sarkar, "Data classification using feature selection and kNN machine learning approach," in *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 811–814, IEEE, 2015.

[46] S. Thirumuruganathan, "A detailed introduction to K-nearest neighbor (KNN) algorithm," *Retrieved March*, vol. 20, p. 2012, 2010.

[47] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 338–345, Morgan Kaufmann Publishers Inc., 1995.

[48] Y. Xu, J. Yin, W. Lo, and Z. Wu, "Personalized location-aware QoS prediction for web services using probabilistic matrix factorization," in *International Conference on Web Information Systems Engineering*, pp. 229–242, Springer, 2013.

[49] 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.

[50] P. Zhang, H. Jin, H. Dong, W. Song, and L. Wang, "LA-LMRBF: online and long-term web service QoS forecasting," *IEEE Transactions on Services Computing*, 2019, DOI: 10.1109/TSC.2019.2901848.

Pengcheng Zhang received the Ph.D. degree in computer science from Southeast University in 2010. He is currently a full professor in College of Computer and Information, Hohai University, Nanjing, China, and was a visiting scholar at San Jose State University, USA. His research interests include software engineering, service computing and data science. He has published in premiere or famous computer science journals. He was the co-chair of IEEE AI Testing 2019 conference. He served as technical program committee member on various international conferences. He is a memeber of the IEEE.

Yaling Zhang is a M. S. candidate in the College of Computer and Information, Hohai University, Nanjing, China. She received her bachelor degree in computer science and technology from Anhui Normal University, Wuhu, China in 2018. Her current research interests include service computing and data mining.

Hai Dong is a Lecturer at School of Computing Technologies in RMIT University, Melbourne, Australia. He received a PhD from Curtin University of Technology, Australia, and a Bachelor degree from Northeastern University, China. His research interests include Service Computing, Distributed Systems, Cyber Security, Machine Learning and Data Analytics. He has published over 90 research articles in international journals and conferences. He is a senior member of the IEEE.

Huiying Jin is a PHD candidate in the College of Computer and Information, Hohai University, Nanjing, China. She received her bachelor degree in Software Engineering from Yangzhou University, Yangzhou, China in 2017. Her current research interests include service computing and data mining.