This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2944596, IEEE Transactions on Services Computing

Context-aware and Adaptive QoS Prediction for Mobile Edge Computing Services

Zhizhong Liu, Quan Z. Sheng, *Member, IEEE,* Xiaofei Xu, *Member, IEEE,* Dianhui Chu, Wei Emma Zhang, *Member, IEEE*

Abstract—Mobile edge computing (MEC) allows the use of its services with low latency, location awareness and mobility support to make up for the disadvantages of cloud computing, and has gained a considerable momentum recently. However, the dynamically changing quality of service (QoS) may result in failures of QoS-ware recommendation and composition of MEC services, which significantly degrades users' satisfaction and negates the advantages of MEC. To address this issue, by considering user-related and service-related contextual factors and various MEC services scheduling scenarios, we propose two context-aware QoS prediction schemes for MEC services. The first scheme is designed for the situations when MEC services are scheduled in real-time, which contains two context-aware real-time QoS estimation methods. One method can estimate the real-time multi-QoS of MEC services and the other method can estimate the real-time fitted QoS of MEC services. The second scheme is designed for the situations when MEC services are scheduled in the future. This scheme includes two context-aware QoS prediction methods. One method can predict the multi-QoS of MEC services and the other method can predict the fitted QoS of MEC services. Finally, adaptive QoS prediction strategies are developed in the light of characteristics of the proposed QoS prediction methods. According to these strategies, the most appropriate QoS prediction method can be scheduled. Extensive experiments are conducted to validate our proposed approaches and to demonstrate their performance.

Index Terms—Mobile Edge Computing; Context-awareness; Adaptive QoS Prediction; Case-Based Reasoning; Support Vector Machine; Artificial Bee Colony Algorithm

1 Introduction

Recently, the rapid proliferation of mobile applications and the Internet of Things (IoT) has posed new requirements (e.g., ultra-low latency, location awareness, mobility support) on cloud infrastructure [1]–[3]. These requirements have become even more critical as more and more smart devices are getting involved in our daily lives (e.g., smart cities, smart homes, smart elderly care). However, current cloud computing paradigm is unable to meet these new requirements. To address the issue, mobile edge computing (MEC) has been proposed as the key technology to assist wireless networks with cloud computing-like capabilities, so as to provide low-latency and mobility support services directly from the network edge [4], [5]. MEC brings more opportunities for users to run compute-intensive calculations in real time. Some mobile applications, such as augmented reality, face recognition, interactive gaming, surveillance systems and natural language processing, will greatly benefit from MEC [6].

- *Z. Liu is with the School of Computer and Control Engineering, Yantai University, Yantai 264005, China. Email: lzzmff@126.com.*
- *Q.Z. Sheng is with the Department of Computing, Macquarie University, Sydney, NSW 2109, Australia. Email: michael.sheng@mq.edu.au.*
- *X. Xu and D. Chu are with the College of Computer Science and Technology, Harbin Institute of Technology, Weihai 264209, China. Email: xiaofei@hit.edu.cn, cdh@hitwh.edu.cn.*
- *W.E. Zhang is with School of Computer Science, the University of Adelaide, SA 5005, Australia. Email: wei.e.zhang@adelaide.edu.au. Dianhui Chu is the corresponding author.*

Manuscript received XX, XXXX; revised XX, XXXX.

With the rapid development of MEC, more and more MEC services with similar functionalities but different quality of service (QoS) are available on the network edge. QoS becomes the key basis for QoS-aware recommendation and composition of MEC services. Unfortunately, under the dynamic and complex environment, QoS attributes of MEC services (e.g., response time, availability, reliability) are changing frequently. This always causes failures of MEC service applications, which significantly degrades users' satisfaction and negates the advantages of MEC. To overcome this problem, it is essential to conduct QoS prediction for MEC services [7].

There are two crucial mechanisms that affect QoS prediction for MEC services. The first one is context-aware mechanism. In fact, some user-related and service-related contextual factors (e.g., the type and volume of the task to be processed, workload of MEC services, the speed of the network) have great impact on the QoS of MEC services. There exists important incidence relation between contextual factors and QoS of MEC services, and a change of any contextual factor will cause change of MEC services' QoS [8]. Therefore, it is necessary to study context-aware QoS prediction method for MEC services.

The other mechanism is adaptability. Firstly, the needs of different formats of QoS data (e.g., fitted QoS, multi-QoS, which are formulated in Section 3.1) call for adaptive QoS prediction. For example, for QoS-aware MEC services recommendation, the fitted QoS values of MEC

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2944596, IEEE Transactions on Services Computing

services are always used. Yet in QoS-aware MEC services composition, concrete values of multi-QoS attributes of MEC services are needed to judge whether the composite MEC services can satisfy the global QoS constraints. Secondly, different MEC service scheduling scenarios require adaptive QoS prediction. For example, for real-time MEC service scheduling, it needs to estimate the QoS values based on real-time contextual factors, which can be extracted from the real-time environmental information. For future MEC service scheduling, it needs to predict the QoS based on the future contextual factors, and some of them need to be obtained through forecasting.

In the past few years, research works have been carried out on QoS prediction for Web services and cloud services, and several QoS prediction approaches have been proposed [9]–[11]. Undoubtedly, previous research works have achieved good results and are useful for improving the accuracy of QoS. However, these methods are not suitable for predicting QoS of MEC services due to the following reasons. Firstly, previous methods do not consider user-related and service-related context, and cannot realize context-aware QoS prediction for MEC services. Secondly, they do not consider different formats of QoS data (multi-QoS or fitted QoS). Thirdly, they only focus on single service scheduling scenario and cannot predict QoS for various MEC services scheduling scenarios adaptively.

To address the problem of context-aware and adaptive QoS prediction for MEC services, we take several important contextual factors and various MEC service scheduling scenarios into the consideration, and propose two context-aware QoS prediction schemes. The first scheme is designed for the situations when MEC services are scheduled in real-time with needs of different QoS data formats. The second scheme is designed for the situations when MEC services are scheduled in the future with the needs of different QoS data formats. Moreover, we propose the adaptive QoS prediction strategies to realize adaptive QoS prediction for MEC services. The main contributions of this paper are as follows:

- For real-time MEC services scheduling with needs of different QoS data formats, we propose a context-aware real-time QoS estimation scheme that includes two QoS estimation methods. One method can estimate the real-time multi-QoS of MEC services, and the other method can estimate the real-time fitted QoS of MEC services.
- For future MEC services scheduling with needs of different QoS data formats, we propose a context-aware QoS prediction scheme which consists of two QoS prediction methods. One method can predict the multi-QoS of MEC services, and the other method can predict the fitted QoS of MEC services.
- We develop a set of adaptive QoS prediction strategies. According to these strategies, the most suitable QoS prediction method can be selected to predict appropriate QoS for a certain MEC service scheduling scenario.

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 outlines the motivation and overview of our approach. Section 4 elaborates the context-aware real-time QoS estimation scheme. Section 5 gives details about the context-aware QoS prediction scheme. Section 6 presents the adaptive prediction strategies. Finally, Section 7 focuses on the performance verification of our proposed methods and Section 8 provides some concluding remarks.

2 Related Work

Over the last few years, the prosperous research on QoS prediction has led to a multitude of results, which can be generally grouped into three categories according to the evolution of QoS prediction research: *contextless QoS prediction*, *time or location aware QoS prediction* and *context-aware QoS prediction*. In the rest of this section, we overview the works that are closely related to these categories.

2.1 Contextless QoS Prediction

Contextless QoS prediction refers to QoS prediction only based on QoS data. In 2007, Shao et al. [12] proposed a QoS prediction approach based on collaborative filtering (CF) by making similarity mining and prediction from consumers' experiences. Zheng et al. [13] proposed a hybrid CF model which fuses the user-based and item based CFs together. To improve the performance of QoS prediction based CF, Wu et al. [14] proposed a ratio-based method to calculate the similarity between users or between items by comparing the attribute values directly. To address the instantaneity of QoS values, Chen et al. [15] proposed a novel similarity-enhanced CF approach to capture the time feature of similarity estimation and replenished the missing QoS value for further prediction. To address the problem of missing QoS values, Chen et al. [16] proposed a QoS prediction model by integrating the geographical neighbors into matrix factorization. By taking advantages of crowd intelligence, Wu et al. [17] proposed novel deviation-based neighborhood models for QoS prediction.

In [18], Zhu et al. proposed an adaptive matrix factorization approach to perform online QoS prediction for candidate services. To exploit the structural relationships among the multi-dimensional QoS data, Wang et al. [19] proposed an integrated QoS prediction approach which unifies the modeling of multi-dimensional QoS data via multi-linear-algebra based concepts of tensor and enables efficient Web service recommendation for mobile clients. Jirsik et al. [20] developed a Long Short-Term Memory Neural Networks model for QoS prediction. Ma et al. [11] proposed a variation-aware approach via collaborative QoS prediction to select an optimal cloud services according to users' non-functional requirements.

2.2 Time or Location aware QoS Prediction

Time or location aware QoS prediction refers to QoS prediction with consideration of time and/or location, which

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2944596, IEEE Transactions on Services Computing

are useful for improving the effect of QoS prediction. Hu et al. [21] proposed an improved time-aware collaborative filtering approach for high-quality Web service recommendation. Ding et al. [22] developed a novel similarity-enhanced collaborative filtering approach to capture the time feature of user similarity and address the data sparsity in the existing point in time. Yu et al. [23] proposed a time-aware and location-aware collaborative filtering algorithm. They divide the users set and services set into many clusters according to the location information.

Considering the influence of location, Liu et al. [24] proposed a location-aware personalized CF method for Web service recommendation. For time-aware Web service recommendation, Wang et al. [25] proposed a spatial-temporal QoS prediction approach based on MF. To make credible location-aware QoS prediction, Li et al. [26] proposed a hybrid matrix factorization method integrated location and reputation information (LRMF) to predict the unattainable QoS values. Zhou et al. [10] proposed two universal spatio-temporal context-aware collaborative neural models to make QoS prediction by considering invocation time and multiple spatial features both of service-side and user-side.

2.3 Context-aware QoS Prediction

Context-aware QoS prediction refers to QoS prediction with consideration of contextual factors. Xu et al. [27] proposed a context-aware QoS prediction method based on MF, they took the geographical information as the user context and took the affiliation information as the service context. Wu et al. [28] proposed a general context-sensitive matrix-factorization approach to make collaborative QoS prediction. Considering the impact of network on Web service QoS, Tang et al. [29] proposed a network-aware QoS prediction approach by integrating MF with the network map. By considering the influences of software, hardware and resources, Zhang et al. [30] proposed a Bayesian network model based QoS prediction method for cloud services. Wu et al. [9] proposed a universal deep neural model (DNM) for making multiple attributes QoS prediction with contexts. In our previous work [31], [32], we proposed a context-aware QoS prediction method based on improved Case-Based Reasoning with consideration of user-related context.

Wang et al. [33] proposed a QoS prediction method for MEC services based on CF, with the consideration of user mobility. Aazam et al. [34] provided a methodology for historical record-based resource estimation to mitigate resource under utilization and to enhance QoS for multimedia IoTs. For edge computing, Yin et al. proposed a QoS prediction method based on matrix factorization model and convolutional neural network (CNN). Although existing context-aware QoS prediction research has achieved good results, they do not consider user-related and service-related contextual factors simultaneously. These methods typically only consider single service scheduling scenario and are lack of adaptability.

Moreover, QoS-aware service placement (scheduling) in mobile edge computing is an important issue. Yousefpour et al. [35] proposed a framework for QoS-aware dynamic fog service provisioning. Since QoS prediction is useful for improving the effect of service placement, Xiong et al. [36] provided an autonomous deployment policy of services based on QoS prediction. However, this work does not consider important user-related and service-related contextual factors.

3 Motivation and Approach Overview

3.1 Motivation

Contextual factors (including user-related and service-related contextual factors) have significant impact on the QoS of MEC services. User-related contextual factors mainly include the type and volume of the task to be processed. Task type means a specific kind of task. For example, "uploading data", "downloading data" and "calculating" are three tasks with different types. Task volume indicates the quantity related to the task. Take the task of "downloading 1GB data" as an example, 1GB denotes the task volume. Service-related contextual factors mainly include network speed and the workload of the MEC services.

Assume that one consumer wants to download two movies using one MEC service, the data sizes of the two movies are 1.5GB and 4GB, respectively. It can easily find out that the downloading time of the smaller-size movie will be shorter than the downloading time of the larger-size movie, when contextual factors (e.g., workload of the MEC services, network speed) remain the same. It is easy to understand that any change of these contextual factors will bring some fluctuations to the QoS of MEC services. Therefore, it is necessary to consider contextual factors when predicting QoS for MEC services.

Moreover, there exist various MEC service scheduling scenarios. For example, some users may want to schedule the MEC services immediately (named as real-time MEC service scheduling) with multi-QoS constraints (or with none QoS constraints), and other users (e.g., in a service composition) may schedule the MEC services in a future time (named as future MEC service scheduling) with multi-QoS constraints (or with none QoS constraints). For different MEC service scheduling scenarios, the system should be able to predict QoS adaptively. For example, for real-time MEC service scheduling with multi-QoS constraints, the system should estimate multi-QoS of MEC services according to real-time contextual factors. For real-time MEC service scheduling without QoS constraints, the system should estimate the fitted QoS of MEC services. Similarly, for future MEC service scheduling with multi-QoS constraints, the system should firstly predict the contextual factors (e.g., workload) of MEC services, then, conduct context-aware multi-QoS prediction for MEC services. The definitions of multi-QoS and fitted QoS are presented as follows:

Definition 1: Multi-QoS. Multi-QoS represents a set of QoS attributes of an MEC service. It can be defined as:

$$MultiQoS(S) =< q_1, ..., q_i, ..., q_l > \quad (1)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2944596, IEEE Transactions on Services Computing

where q_i is the i^{th} QoS attribute of the MEC service S.

Definition 2: Fitted QoS. Fitted QoS indicates the comprehensive evaluation of the QoS attributes of an MEC service. It is defined as:

$$FittedQoS(S) = \sum_{i=1}^{k} a_i * q_i \quad (2)$$

where q_i is the i^{th} QoS attribute of the MEC service S, and a_i is the weight of the i^{th} QoS attribute.

3.2 Overview of Our Approach

To realize context-aware and adaptive QoS prediction for MEC services, we firstly take user-related and service-related contextual factors into consideration. Then, we group MEC services application scenarios into two major categories: real-time MEC services scheduling and future MEC services scheduling. Next, considering the needs of different formats of QoS data, we classify the first major category into two subclasses, namely i) real-time MEC services scheduling with need of multi-QoS and ii) real-time MEC services scheduling with need of fitted QoS. We also classify the second major category into two subclasses, i.e., i) future MEC services scheduling with need of multi-QoS and ii) future MEC services scheduling with need of fitted QoS.

For the real-time MEC service scheduling scenario, we propose the context-aware real-time QoS estimation scheme that consists of two QoS estimation methods. The first method is based on the improved case-based reasoning (CBR), which can estimate the real-time multi-QoS of MEC services. The second method is based on the optimized support vector machine (O-SVM), which can estimate the real-time fitted QoS of MEC services. Similarly, for the future MEC service scheduling scenario, we propose the context-aware QoS prediction scheme that includes two QoS prediction methods. The first method forecasts the workload of MEC services with O-SVM, and then predicts the multi-QoS of MEC services with the improved CBR. The second method forecasts the workload of MEC services with O-SVM, and then predicts the fitted QoS of MEC services.

To realize adaptive QoS prediction for MEC services, we further develop a set of adaptive QoS prediction strategies. The main idea is that, when a MEC service application requirement comes, the system firstly classifies the requirement into one major category according to the scheduling time, then classifies the requirement into a subclass according to the demand of QoS data format, and finally selects the most suitable QoS prediction method according to the adaptive QoS prediction strategies. The overview of our approach is depicted in Fig. 1.

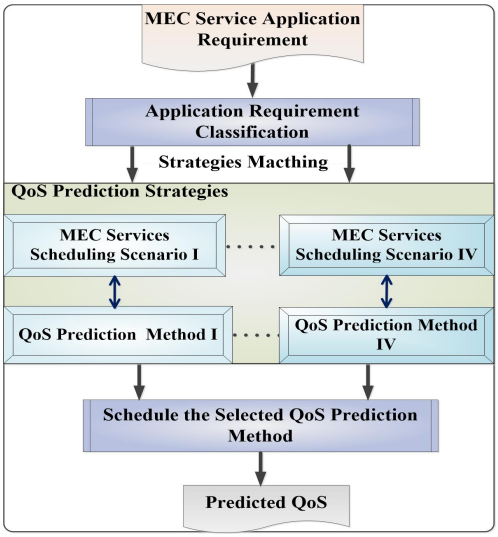

Fig. 1. Overview of our approach

4 Context-aware Real-time QoS Estimation Scheme

The context-aware, real-time QoS estimation scheme is proposed for situations when MEC services need to be scheduled in real-time. This scheme consists of two context-aware real-time QoS estimation methods. The first one is the real-time multi-QoS estimation method, and the second one is the real-time fitted QoS estimation method.

4.1 Context-aware Real-time Multi-QoS Estimation Method

Our context-aware and real-time multi-QoS estimation method is proposed based on the improved Case-Based Reasoning (CBR). CBR is a popular problem solving methodology that uses a previous data or memorized problem situations called cases [37]. The life cycle of a CBR-based system consists of four main phases: i) identifying the current problem and finding a past case similar to the new case (retrieve), ii) using the case and suggesting a solution to the current problem (reuse), iii) evaluating the proposed solution (revise), and iv) updating the system to learn from experience (retain) [37].

CBR is very suitable for context-aware, real-time multi-QoS estimation because its case model supports multi-input and multi-output computing. The main idea of QoS prediction based on improved CBR is that when an MEC service executes under the same or similar contextual factors, the QoS of the MEC service will be the same or similar. In this work, we develop a QoS case model, QoS cases retrieving phase and QoS cases reuse phase for context-aware, real-time multi-QoS estimation, which are presented as follows.

(1) QoS Case Model. In CBR, each case contains values for a set of features and a corresponding solution. For the problem of context-aware, real-time multi-QoS prediction, we construct the QoS case model by using the first part of the model to describe the contextual factors, and applying the second part of the model to present the multi-QoS values of the MEC service. The multi-QoS values indicate the QoS of the MEC service when it deals with a task under the context described by the first part. The multi-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2944596, IEEE Transactions on Services Computing

QoS values of the MEC services can be provided by services providers.

Moreover, the newer a QoS case is, the stronger impact it will play on multi-QoS estimation. So, we add a time stamp on the model to indicate when the QoS case is constructed. The QoS case model is defined as:

$$QoSC =< TT, TV, NS, WL \| q_1, q_2, ..., q_k >^{TimeTag} \quad (3)$$

where TT and TV indicate the type and volume of the task to be processed. NS indicates the network speed, WL denotes the workload of the MEC service. $q_1, ..., q_k$ indicates the multi-QoS values of the MEC service. $TimeTag$ indicates the time when this QoS case is constructed. This QoS case model is extensible in the sense that other contextual factors and QoS attributes can be added into or deleted from the model.

For creating QoS cases, values of TT and TA can be extracted from user application requirements. In this work, we use the utilization of CPU of the server where the EMC service is employed to indicate the workload WL. Values of NS and WL can be obtained by network speed monitor and CPU detector, respectively. QoS cases of an MEC service can be constructed based on the contextual factors and multi-QoS values of the MEC service when it finishes the task. In this way, multiple cases can be constructed when the service deals with different tasks, and we can construct the QoS case base (QCB) of the MEC service with these QoS cases.

(2) QoS Cases Retrieving Phase. QoS cases retrieving aims to find the top K similar historical QoS cases with the target QoS case. The target QoS case model is defined as:

$$TC =< TT_0, TV_0, NS_0, WL_0, \| ?q_1, ?q_2, ..., ?q_k > \quad (4)$$

Where TT_0 and TV_0 indicate the type and volume of the task to be processed. NS_0 indicates the current network speed. WL_0 means the current workload of the MEC service. $?q_1, ?q_2, ..., ?q_k$ indicate the multi-QoS values to be estimated. Similar historical QoS cases are retrieved according to the similarities between the target QoS case and historical QoS cases in the QCB. In this work, we calculate the similarity between the target QoS case and a historical QoS case using:

$$Sim(TC, QC_i) = w_{s1} sim_{1i} + w_{s2} sim_{2i} + w_{s3} sim_{3i} \quad (5)$$

where TC indicates the target QoS case, QC_i means the i^{th} historical QoS case. sim_{1i}, sim_{2i} and sim_{3i} are three local similarities, which denote similarities of task volume, network speed and workload between the target QoS case and the i^{th} historical QoS case. w_{s1}, w_{s2} and w_{s3} indicate the weights of the three local similarities, and $w_{s1} + w_{s2} + w_{s3} = 1$. The calculation of the three local similarities are defined as Eqn (6), (7) and (8):

$$sim_{1i} = 1 - \frac{|TV_0 - TV_i|}{TV_{max}} \quad (6)$$

where TV_0 and TV_i indicate the task volume of the target QoS case and the i^{th} historical QoS case, respectively. TV_{max} means the maximum of task volume that the MEC service can deal with.

$$sim_{2i} = 1 - \frac{|NS_0 - NS_i|}{NS_{max}} \quad (7)$$

where NS_0 and NS_i represent the network speed of the target QoS case and the i^{th} historical QoS case. NS_{max} means the maximum of network speed.

$$sim_{3i} = 1 - |WL_0 - WL_i| \quad (8)$$

where WL_0 and WL_i denote the workload of the target QoS case and the i^{th} historical QoS case. The procedure for retrieving the top K similar historical QoS cases is described as follows:

Step 1: Screening out historical QoS cases that have the same task type with the target QoS case.
Step 2: Calculating the similarities between the target QoS case and each historical QoS case.
Step 3: Selecting the top K historical QoS cases according to the similarities.

(3) QoS Cases Reusing Phase. Real-time multi-QoS estimation is realized through the reuse of the top K similar historical QoS cases. Assume that the top K similar historical QoS cases are $\{QC_1, QC_2, ..., QC_k\}$. Then, the K historical QoS cases are sorted according to their $TimeTag$, and the sequence of the sorted QoS cases is $QC_1 \succ QC_2 \succ ... \succ QC_k$. Based on these K historical QoS cases, the multi-QoS values of the target QoS case can be estimated using:

$$TC_{QoS} = w_1^{'} QC_{1QoS} + w_2^{'} QC_{2QoS} + ... + w_k^{'} QC_{kQoS} \quad (9)$$

where TC_{QoS} denotes the multi-QoS values of the target QoS case. QC_{1QoS}, QC_{2QoS}, ... and QC_{kQoS} indicate the multi-QoS values of the K similar historical QoS cases. $w_1^{'}, w_2^{'}, ..., w_k^{'}$ are the weights of the K historical QoS cases, and $w_1 + w_2^{'} + ... + w_k^{'} = 1$. To ensure that the newer QoS case acts stronger influence on multi-QoS estimation, we set the weight of the newer historical QoS case slightly larger. Our context-aware, real-time multi-QoS estimation method is described in Algorithm 1, where the values of parameters TT_0 and TV_0 can be obtained from the user service requirement, and the values of NS_0 and WL_0 can be obtained from network monitoring device and server management software.

A sensitivity analysis of the algorithm performance with respect to error in the input values is as follows. Firstly, the contextual parameter TT_0 has the greatest influence on the algorithm performance. An error of TT_0 can make the prediction result useless. The errors of other three contextual parameters (TV_0, NS_0, WL_0) can also affect the prediction accuracy of the algorithm, and the affection degree depends on parameters' weights. The lager the weights are, the more serious the affection will be. Moreover, the errors of K and the weights can also affect the prediction accuracy of the algorithm.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2944596, IEEE Transactions on Services Computing

Algorithm 1 Context-aware real-time multi-QoS estimation

Input:
TT_0: Task Type
TV_0: Task Volume
NS_0: Network Speed
WL_0: Workload
QCB: QoS Cases Base
K: The number of similar QoS cases
w_{s1}, w_{s2}, w_{s3}: Values of weights of local similarities
$w_1^{'}, w_2^{'}, ..., w_k^{'}$: Values of weights of the K similar QoS cases
Output: Estimated multi-QoS values

Begin
Step 1. Construct the target QoS case
$TC \leftarrow (TT_0, TV_0, NS_0, WL_0 | ?q_1, ?q_2, ..., ?q_k)$ /* *construct target QoS case according to contextual factors* */
Step 2. QoS Cases Retrieving Phase
for $i = 0 \rightarrow n$ **do** /* *for each historical QoS case in the QCB* */
  $Sim(TC, QC_i) \leftarrow forumla(5)$ /* *calculate the similarity between the* i^{th} *QoS case and TC* */
end for
for $i = 0 \rightarrow n$ **do**/* *for each historical QoS case in the QCB* */
  Sort historical QoS cases according to similarities
end for
for $i = 0 \rightarrow K$ **do**/* *select the top K QoS cases form the QCB* */
  Put the K^{th} *QoS case into the similar QoS case set*
end for
Step 3. QoS Cases Reusing Phase
$TC_{QoS} \leftarrow forumla(9)$ /* *calculate multi-QoS values of TC based on the K historical QoS cases* */
return *Estimated multi-QoS values*
Step 4. Revise and Retain the new QoS Case
End

4.2 Context-aware Real-time Fitted QoS Estimation Method

The context-aware real-time fitted QoS estimation method is proposed based on the optimized support vector machine (O-SVM). Support vector machine (SVM) is a machine-learning tool that has been originated from Statistical Learning Theory (SLT) developed by Vapnik [38]. SVM is the most suitable method for context-aware fitted QoS estimation because SVM supports the multi-input and single-output computing [39]. To improve the performance of SVM, we propose the improved artificial bee colony algorithm (ABC) to optimize the key parameters of SVM with radial basis function (RBF). In the following sections, we firstly introduce SVM briefly, then, we present the improvement of ABC. Finally, we describe the optimization process of SVM based on improved ABC and present the context-aware, real-time fitted QoS estimation method based on O-SVM in details.

4.2.1 Support Vector Machine

Assume that $\{(x_i, y_i)\}_{i=1}^n$ indicates a set of data, where $x_i \in R^n$ denotes the input vectors, $y_i \in \{+1, -1\}$ stands for two classes, and n is the number of sample data. It is possible to determine the hyperplane $f(x) = 0$ that separates the given data when two classes are linearly separable. $f(x)$ is defined as:

$$y = f(x) = w \cdot (x) + b = \sum_{k=1}^{n} w_k \cdot x_k + b \tag{10}$$

where w is the weight vector and b indicates the bias term. w and b are used to define the position of the separating hyperplane. By introducing the Lagrangian multipliers, the optimization problem is transformed into the dual quadratic optimization problem (defined as Eqn (11)) with the decision function which is defined as Eqn (12) [38]:

$$MaxmizeL(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i, x_j) \tag{11}$$

$$s.t. \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ 0 \leq \alpha_i, i = 1, 2, \ldots, n \end{cases}$$

$$f(x) = sign(\sum_{i,j}^{n} \alpha_i y_i K(x_i, x_j)) + b \tag{12}$$

where $K(x_i, x_j)$ is called the kernel function and $k(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$. In this work, we adopt the radial basis function (RBF) as the kernel function, because SVM constructed with RBF has excellent nonlinear classification ability [39]. RBF is defined as:

$$K(x_i, x_j) = exp(-||x_i - x_j||/2\sigma^2) \tag{13}$$

The key parameters of SVM with RBF kernel function are the punishment parameter C and the kernel parameter σ, which determine the learning and predictive capability of SVM. Some research work has applied optimization algorithms (e.g., grid search [40], genetic algorithm [41], particle swarm optimization algorithm [42]) to obtain optimal values for these two parameters. However, existing optimization algorithms are easy to fall into local optimum and with low convergence speed. To overcome these shortcomings, we propose an improved Artificial Bee Colony algorithm to optimize these two parameters.

4.2.2 Improved Artificial Bee Colony Algorithm

Artificial Bee Colony algorithm (ABC) is a new swarm intelligent algorithm and has been successfully applied in solving optimization problems [43]. The main steps of the ABC algorithm are as follows:

Initialize the parameters.
While Termination criteria is not satisfied **Do**
Step1: Employed Bees Phase for generating food sources;
Step2: Onlooker Bees Phase for updating the food sources depending on their nectar amounts;
Step3: Scout Bees Phase for discovering the new food sources in place of abandoned food sources;
Step4: Memorize the best food source found so far;
EndWhile
Output The best solution found so far.

Although ABC has better performance than other swarm intelligent algorithms [43], it still has some shortcomings. At the onlooker bees phase, food sources are selected according to the roulette wheel method. This method always decrease the diversity of the population, and results in stagnation and falling into local optimum.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2944596, IEEE Transactions on Services Computing

At the scout bees phase, some food sources are abandoned according to the parameter $Limit$ with a fixed value. This method always causes low flexibility. Moreover, scout bees randomly identify new food sources, and this method would decrease the convergence speed of ABC.

To overcome the above shortcomings, we propose an improved onlooker bees phase with a new food sources selection method based on the immune selection mechanism, as well as an improved scout bees phase with a new food source abandoning method based on the immunologic mechanism and a new food source generation method based on the escape mechanism. The improvements of ABC are presented as follows.

(1) Improved onlooker bees phase. Relating each food source to a specific antibody, the selection probability of each food source is determined by its fitness proportion and antibody concentration. The selection probability of each food source can be calculated using:

$$P(X_i) = \alpha Pf_i + (1-\alpha)Pa_i \tag{14}$$

where α is a constant number and $\alpha \in (0,1]$. Pf_i indicates the fitness proportion which is calculated using:

$$Pf_i = \frac{Fit_i}{\sum_{i=1}^{SN} Fit_i} \tag{15}$$

where Pa_i presents the selection probability which is determined by the antibody concentration that is calculated by using:

$$Pa_i = exp\left(-\frac{C_{x_i}}{k}\right) \tag{16}$$

where k is a constant number and $k \in (0,1]$. C_{x_i} indicates the concentration of antibody X_i and it is calculated by using:

$$C_{X_i} = \frac{1}{SN}\sum_{j=1}^{SN} G_{ij} \tag{17}$$

$$G_{ij} = \begin{cases} 1, & a_{X_{ij}} \geq TH \\ 0, & else \end{cases}$$

where $a_{X_{ij}}$ indicates the affinity between antibody X_i and X_j. TH indicates the determined threshold. $a_{X_{ij}}$ is calculated using:

$$a_{X_{ij}} = \frac{1}{1+ED} \tag{18}$$

where ED is the Euclidean distance between the antibody X_i and X_j.

According to the new food source selection method, when some food sources have the same fitness, the selection probability of food sources with higher antibody concentration will be smaller. Conversely, the selection probability of food sources with smaller antibody concentration will be larger. When some food sources have the same antibody concentrations, food sources with higher fitness are more likely to be selected. This new food source selection method not only can preserve the food source with high fitness, but also can maintain the diversity of the population.

(2) Improved scout bees phase. To reduce the dependence of scout bees on the parameter $Limit$, a new method for determining the food sources to be abandoned is proposed based on the roulette wheel with escape indicator. The selection possibility of each food source X_i is calculated as:

$$PA(X_i) = \frac{Esp_i - (Esp_i)_{min}}{(Esp_i)_{max} - (Esp_i)_{min}} \tag{19}$$

where Esp_i indicates the escape indicator of individual X_i. The escape indicator is determined by the times of the individual that has not been updated and the impact of individual position on the population diversity. Here $(Esp_i)_{max}$ and $(Esp_i)_{min}$ indicate the maximum and minimum values of the property, and Esp_i can be calculated by using:

$$Esp_i = limit_i - dis_i \tag{20}$$

where dis_i indicates the impact of the position of individual X_i on the population diversity, which can be calculated using:

$$dis_i = |Fit_i - aveFit| \tag{21}$$

where $aveFit$ indicates the average value of the population's fitness. This formula uses the deviation degree between the fitness of X_i and average fitness of the population to determine the effect of the position of X_i on the population variation diversity.

To overcome the randomness of food sources generation in scout bees phase, we propose a new food source generation method based on escape mechanism, which is defined as:

$$x_{ij}^{new} = x_{ij} + \varphi_{ij} \cdot \left(1 - \frac{g}{G}\right) \cdot x_{ij} \tag{22}$$

where φ_{ij} is a random parameter and $\varphi_{ij} \in [-1,1]$. G is the max iteration of the algorithm, g is the current iteration time, and is used to control the escape scale. Meanwhile, to ensure solution x_{ij}^{new} to be feasible, the reflection strategy is adopted to restrict x_{ij}^{new} into $[a_j, b_j]$, which is defined in the following, where "mod" indicates the Modulo Operation.

$$x_{ij}^{new} = \begin{cases} b_j - (x_{ij}^{new} - b_j)mod(b_j - a_j), & x_{ij}^{new} > b_j \\ a_j - (a_j - x_{ij}^{new})mod(b_j - a_j), & x_{ij}^{new} < a_j \\ x_{ij}^{new}, & else \end{cases} \tag{23}$$

From Eqn (19) we can see that an adaptive selection mechanism is adopted to determine the scout bees, and this mechanism can improve the flexibility of scout bees determination. Moreover, the variable scale escape operator defined in Eqn (22) can enable scout bees to make large scale escape in the early stage and decrease the escape scale gradually in the later stage of search. So, the improved scout bees determination method and the new food source generation method not only can maintain the diversity of population, but also can strengthen the optimization ability in the later search stage.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2944596, IEEE Transactions on Services Computing

4.2.3 Optimized SVM based on Improved ABC

For SVM to have better performance, we apply the improved ABC (I-ABC) to optimize the parameters C and σ of SVM. Here, (C, σ) is taken as the food source of I-ABC, fitness of (C, σ) indicates the performance of the SVM with parameter C and σ. When computing the fitness, it uses the training data set to train the SVM model, and then tests the trained model with testing data set. In this work, SVM is trained and tested base on the LibSVM Toolbox of Matlab. The performance evaluation value of SVM is taken as the fitness of the food source. The fitness function of I-ABC is defined as the regression accuracy of the model on the testing data set. The fitness function is defined as:

$$Fit_i = 1 - \frac{|y_i - \hat{y_i}|}{y_i} \tag{24}$$

where y_i is the actual value and $\hat{y_i}$ is the predicted value. The process of optimizing SVM based on I-ABC is described in Algorithm 2.

For context-aware and real-time fitted QoS prediction for MEC services, the sample data format is defined as:

$$sd_i =< (cf_{i1}, cf_{i2}, ..., cf_{id}), fq_i > \tag{25}$$

where $(cf_{i1}, cf_{i2}, ..., cf_{id})$ means the context factors of the i^{th} sample data. $fq_i, i = 1, .., n$ indicates the fitted QoS value in the i^{th} sample data. For O-SVM, $(cf_{i1}, cf_{i2}, ..., cf_{id})$ is the input data, fq_i is the output data for the predicted fitted QoS. Moreover, this QoS case model is extensible in the sense that other contextual factors and QoS attributes can be added into or deleted from the model.

Before applying O-SVM to predict the fitted QoS for a certain MEC service, we need to normalize each dimension of the sample data to the range $[0, 1]$ according to:

$$x_i^* = \frac{x^{'} - x_{min}}{x_{max} - x_{min}} \tag{26}$$

where x_i indicates the i^{th} dimension data, x_{min} and x_{max} are the minimum and maximum values of x_i. x_i^* indicates the normalized value of x_i. Context-aware, real-time fitted QoS estimation based on O-SVM is described in Algorithm 3.

Algorithm 2 Optimizing SVM based on I-ABC

Input:
SN: the number of the food sources
G: the maximum evolutionary algebra
C: the punishment parameter and $C \in [0.1, 1000]$
σ: the kernel parameter and $\sigma \in [0.1, 1000]$
$TrDS$: the training data set for SVM
$TeDS$: the testing data set for SVM
Output: Optimal values of (C, σ)

Begin
Step 1. Generate initial food sources
for $i = 1 \rightarrow SN$ **do** /* SN *is the number of food sources*
  $X_i \leftarrow (C, \sigma,$ *food source generation formula*)
    /* *generate the* i^{th} *food source*
  $Fit_i \leftarrow$ (*SVM training, SVM testing,* $formula(24)$)
    /* *compute the fitness of the* i^{th} *food source*
end for
Asign food sources to the employed bees randomly
Repeat
Step 2. Employed Bees Phase
for $i = 1 \rightarrow SN$ **do**
  The i-th employed bee generates a new food source
  $Fit_i^{new} \leftarrow$ (*SVM training, SVM testing,* $formula(24)$)
    /* *compute the fitness of the new food source*
  Compare the old and the new food source; keep the better one
end for
Step 3. Improved Onlooker Bees Phase
for $i = 1 \rightarrow SN$ **do**
  $P(X_i) \leftarrow formula(14)$ /* *calculate the selection probability of the* i^{th} *food source*
end for
for $i = 1 \rightarrow SN$ **do**
  Select a food source for the i-th onlooker bee based on immune selection mechanism
end for
Carry out Step3
Improved Scout Bees Phase
for $i = 1 \rightarrow SN$ **do**
  $PA(X_i) \leftarrow formula(19)$ /* *calculate the selection probability of the* i^{th} *food source*
end for
for $i = 1 \rightarrow SN * 20\%$ **do**
  Select a scout bee according to the immunologic mechanism
  Generate a new food source according to formula (22)
  $Fit_{new} \leftarrow$ (*SVM training, SVM testing,* $formula(24)$)
    /* *compute the fitness of the new food source*
end for
Memorize the best solution achieved so far
UNTIL(The termination condition is satisfied)

5 Context-aware QoS Prediction Scheme

Our context-aware QoS prediction scheme is proposed for situations when MEC services need to be scheduled in the future. This scheme consists of two context-aware QoS prediction methods. The first one is the multi-QoS prediction method, and the second one is the fitted QoS prediction method. Since the workload of MEC services is one important contextual factor and changes dynamically, we first predict MEC services' workload at the future time ft, and then, predict multi-QoS or fitted QoS of MEC services based on the predicted workload and other contextual factors. For realizing context-aware QoS prediction for MEC services, we propose a workload prediction method based on O-SVM.

5.1 Workload Prediction Based on O-SVM

The workload of MEC services is a time series data and has the characteristic of self-similarity, so, it is feasible to predict the future workload of MEC services based on historical workload data. When predicting the workload at time i, we take m workload values prior to the time i as the input data, and the output of O-SVM is the predicted workload at i. For the workload prediction, the sample data format is defined as:

$$wl_i =< (wl_{i,t-m}, wl_{i,t-(m-1)}, ..., wl_{i,t-1}), wl_{i,t} > \tag{27}$$

where $(wl_{i,t-m}, wl_{i,t-(m-1)}, ..., wl_{i,t-1})$ means the input data of the i^{th} sample data. $wl_{i,t}$ indicates the output of the i^{th} sample data. Workload prediction based on O-SVM is described in Algorithm 4.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2944596, IEEE Transactions on Services Computing

Algorithm 3 Context-aware real-time fitted QoS estimation

Input:
$HistoricalContext - fittedQoS$: historical data
$Currentcontext$: current contextual factors
Output: Predicted fitted QoS value fq_0.

Begin
Step 1. Construct training sample data
for $i = 1 \rightarrow n$ **do** /* n *is the number of sample data*
  $sd_i \leftarrow (< (cf_{i1}, cf_{i2}, ..., cf_{id}), fq_i >)$
    /* *construct the* i^{th} *fitted QoS sample data*
end for
Step 2. Data normalizing
for $i = 1 \rightarrow n$ **do** /* n *is the number of sample data*
  for $j = 1 \rightarrow d + 1$ **do**/* d *is the dimension number*
    Normalize the j^{th} *dimensional data according to formula (26)*
  end for
end for
Step 3. Optimize the Parameters of SVM
  Get the optimal values for parameters (C, σ) *with Alg.2*
Step 4. Establish the fitted QoS prediction model
  Train the O-SVM on the training sample data with the Libsvm tool
  Obtain the decision function
  Establish the QoS prediction model
Step 5. Predict the fitted QoS
  $sd_0 \leftarrow (< (cf_{01}, cf_{02}, ..., cf_{0d}), fq_0 >)$
  /* *construct the target workload prediction data*
  Input $(cf_{01}, cf_{02}, ..., cf_{0d})$ *and start O-SVM*
  Return fq_0 /* fq_0 *is the predicted fitted QoS*
End

Algorithm 4 Workload prediction based on O-SVM

Input:
HWD: historical workload data.
ft: the future time.
Output:
wl_{ft}: predicted workload at ft.

Begin
Step 1. Construct training sample data
for $i = 1 \rightarrow n$ **do** /* n *is the number of sample data*
  $wl_i \leftarrow (wl_i =< (wl_{i,t-m}, wl_{i,t-(m-1)}, ..., wl_{i,t-1}), wl_{i,t} >)$
    /* *construct the* i^{th} *sample data for workload prediction*
end for
Step 2. Data normalizing
for $i = 1 \rightarrow n$ **do** /* n *is the number of sample data*
  for $j = 1 \rightarrow d + 1$ **do** /* d *is the dimension number*
    Normalize the j^{th} *dimensional data according to formula (24)*
  end for
end for
Step 3. Optimize the Parameters of SVM
  Get the optimal values of parameters (C, σ) *with Alg.2.*
Step 4. Establish the fitted QoS prediction model
  Train the O-SVM on the training sample data with the Libsvm tool.
  Obtain the decision function.
  Establish the workload prediction model.
Step 5. Predict the workload
  $wl_0 \leftarrow ((< (wl_{t-m}, wl_{t-(m-1)}, ..., wl_{t-1}), wl_t >)$
  /* *construct the target workload prediction data*
  Input $(wl_{t-m}, wl_{t-(m-1)}, ..., wl_{t-1})$ *and start O-SVM*
  Return wl_t /* wl_t *is the predicted workload*
End

5.2 Context-aware Multi-QoS Prediction Method

To realize the context-aware multi-QoS prediction for MEC services at future time ft, it is necessary to predict the workload of MEC services at time ft. Once the workload is known, the procedure of context-aware multi-QoS prediction is the same to the procedure of context-aware real-time multi-QoS estimation method. We firstly apply Algorithm 4 to predict the workload of MEC services, then, apply Algorithm 1 to predict the multi-QoS of MEC services. The main procedure of the context-aware multi-QoS prediction for MEC services is described as follows:

Step 1 Execute Algorithm 4 to predict the workload of MEC services at future time ft.
Step 2 Take the predicted workload and other contextual factors as input data for Algorithm 1.
Step 3 Apply Algorithm 1 to predict the multi-QoS of MEC services at future time ft.

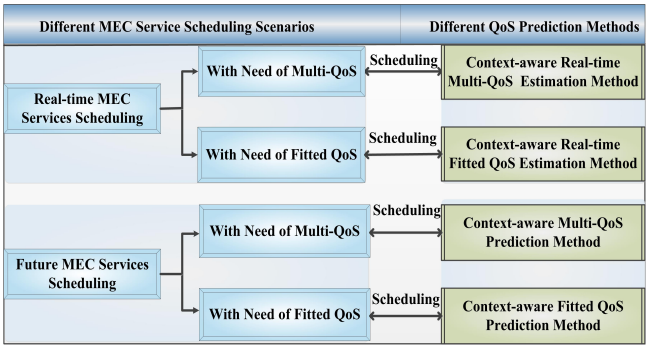

Fig. 2. Adaptive QoS Prediction Strategies

5.3 Context-aware Fitted QoS Prediction Method

Similar to context-aware multi-QoS prediction, to predict the fitted QoS of MEC services at ft, we firstly predict the workload of MEC services at ft, and then follow the procedure similar to the context-aware real-time fitted QoS prediction:

Step 1 Execute Algorithm 4 to predict the workload of MEC services at future time ft.
Step 2 Take the predicted workload and other contextual factors as input data for Algorithm 3.
Step 3 Apply Algorithm 3 to predict the fitted QoS of MEC services at ft.

6 Adaptive QoS Prediction Strategies

More and more MEC services have been increasingly used in a wide range of applications. Such MEC service application scenarios can be grouped into four categories: i) real-time MEC services scheduling with need of multi-QoS, ii) real-time MEC services scheduling with need of fitted QoS, iii) future MEC services scheduling with need of multi-QoS, and iv) future MEC services scheduling with need of fitted QoS. Accordingly, we propose four QoS prediction methods to satisfy the needs of QoS prediction for the four MEC services scheduling scenarios.

For the system to adaptively select the most suitable method to predict appropriate QoS for MEC service applications, we propose strategies in light of the characteristics of our proposed QoS prediction methods. The adaptive QoS prediction strategies are illustrated in Fig. 2.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2944596, IEEE Transactions on Services Computing

According to these strategies, the system can select the most suitable QoS prediction method for an application.

7 Experimental Studies

7.1 Experiments Design

In our experimental study, an MEC service with two computing functions is developed and deployed on an edge cloud server with the following configuration, CPU2vCPU: 1674MHz, Memory: 1024Mbytes, OS: Windows XP. One function of the MEC service is calculating the rank of the large matrix, and the other function of the MEC service is calculating the eigenvalues of large matrix. In this experiment, we take task of calculating matrix rank as task type I, calculating matrix eigenvalues as task type II, and the order of large matrix as the task volume.

In this experiment, the client is simulated by a smartphone with the following configuration, CPU: 2.2GHz, Memory: 2GB, OS: Android OS 5.0. During this experiment, the client randomly submits computing requirements with different task types and task volumes. The types of the tasks are generated from task type I and II, the volumes of tasks are randomly generated in the range of [500, 2,000]. After the MEC service finishes a task, we record the task type, task volume, workload, network speed and response time of the EMC service.

In the experiment, CPU utilization is used to indicate the workload of the MEC service. During the execution of the MEC services, CPU utilization is recorded every 50ms. After a task is completed, the average value of CPU utilization is taken as the workload of the MEC service. In this experiment, since the user just submits the task requirements and receives the results through the wireless network, the time used for these two operations is very short and the network only slightly influences the performance of the MEC service. We therefore take the network speed as a determined value. In our experiment, we collect 1,750 QoS cases with task type I and 1,500 QoS cases with task type II. Based on these collected data, we construct the QoS case base.

7.2 Performance Verification of the Improved ABC

To verify the effect of the improved ABC, we apply the improved ABC and other four optimization algorithms, namely grid search (GS), partical swarm optimization (PSO), genetic algorithm (GA), and ABC, to optimize the key parameters of SVM, and then use the five optimized SVMs to complete QoS prediction. In this experiment, the first 100 QoS data are extracted and used as the training data set, then, four groups of testing data are randomly selected from the rest data. Experimental results are presented in Table 1. From Table 1 we can find that the I-ABC_SVM achieves the best performance, indicating that the improved ABC is effective.

7.3 Performance Verification of Context-aware Real-time QoS Estimation Methods

This experiment aims to verify the performance of the two context-aware real-time QoS estimation methods in the first scheme. To the best of our knowledge, there are no works about context-aware real-time QoS estimation for MEC services to compare with our methods. Therefore, we focus on studying the prediction accuracy and effectiveness of our proposed methods.

TABLE 1
Prediction accuracy of the five optimized SVMs

Group	Predication Accuracy (%)				
	GS	PSO	GA	ABC	I-ABC
G1	89.514	81.776	93.556	93.478	93.619
G2	92.738	92.033	93.965	93.451	94.195
G3	85.232	86.184	89.471	88.623	89.523
G4	89.265	88.980	89.648	90.104	92.840
Average	89.187	87.243	91.660	91.387	93.544

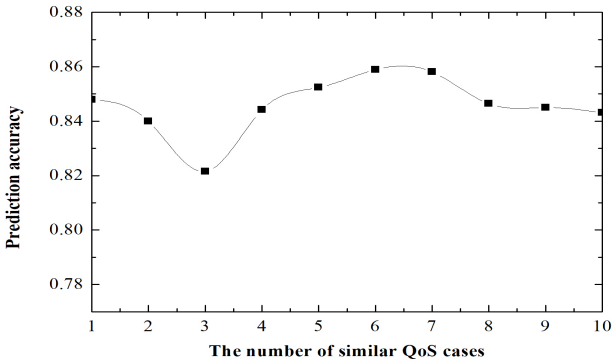

Fig. 3. Prediction accuracy with different values of K

7.3.1 Determining the Optimal Value for K

For Case-based Reasoning (CBR), the quality of the top K similar historical cases has great influence on the performance. Therefore, it is necessary to determine the most suitable value for K. In this experiment, the context-aware real-time multi-QoS estimation method based on improved CBR is implemented in Java. The experimental environment is a PC with the following configuration, CPU: Intel(R)4 2.40 GHz, Memory: 1 GB, OS: Microsoft Windows 2,000. Here, the 10-fold cross-validation method is adopted. Each data fold includes 30 QoS cases. For each data fold, we set value for K as $1, 2, \ldots, 10$, respectively. Then, we predict the QoS with the testing QoS cases. In this experiment, we set $w_{s1} = 0.6$, $w_{s2} = 0.0$, $w_{s3} = 0.4$. To simplify the experiment, we set $w_1^{'} = w_2^{'} = ... = w_K^{'}$ and $w_1^{'} + w_2^{'} + ... + w_K^{'} = 1$. The experimental results are presented in Fig. 3.

From Fig. 3 we can find that, the prediction accuracy of context-aware real-time multi-QoS estimation method is fluctuating with the increase of K. Based on these experimental results, we can find that the prediction accuracy is the best when K is set as 6. So, we set the value of K as 6 in our work.

7.3.2 Performance Verification

When applying the improved CBR to predict the real-time multi-QoS for MEC services, we randomly select 896 historical QoS cases, then, take the first 800 QoS cases as the training set and take the last 96 QoS cases as the testing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2944596, IEEE Transactions on Services Computing

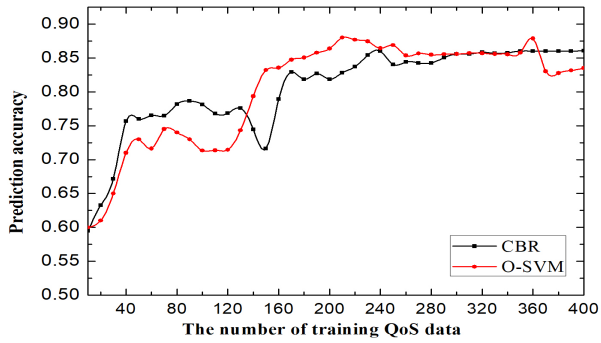

Fig. 4. Prediction accuracy of real-time QoS estimation methods

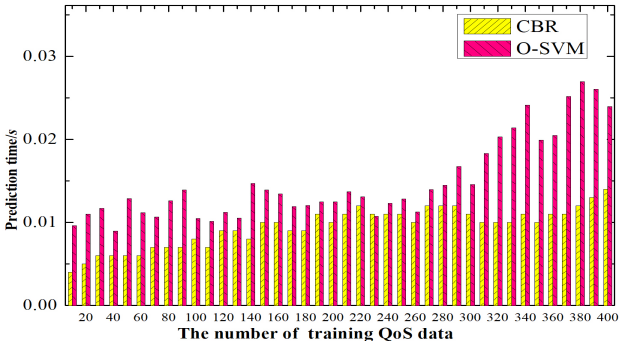

Fig. 5. Running time of two real-time QoS estimation methods

set. For each experiment, we randomly select 20 QoS cases from the testing set and take them as the testing cases.

For O-SVM, we set the loss function parameter $\varepsilon = 0.1$, the value ranges of the penalty parameter C and the kernel function parameter σ are set as $[0.1, 1000]$. The values of C and σ are determined by Algorithm 2 dynamically. When applying O-SVM to predict the fitted QoS, we divide the training data set into groups with a 50 interval, and conduct the parameter optimization and SVM training for each data group. The model derived based on the current data group will be used for QoS prediction for the next data group.

Context-aware real-time multi-QoS estimation method and context-aware real-time fitted QoS estimation method are executed when the number of training data reaches 10, 20, 30, ..., 380, 390 and 400. For each experiment, we iterate each method for 20 times and compute the average accuracy. The prediction accuracy of the two methods are shown in Fig. 4, where the vertical coordinate indicates the prediction accuracy and the horizontal coordinate indicates the scale of the training data. In Fig. 4, we use CBR to indicate the context-aware real-time multi-QoS estimation method and O-SVM to denote the context-aware real-time fitted QoS estimation method.

From the figure we can find that the prediction accuracy of CBR and O-SVM are gradually improved with the increase of the training QoS data. When the number of the training QoS data is in $[10, 140]$, the prediction accuracy of CBR is higher than that of O-SVM. The reason is that, when the number of training QoS data is small, O-SVM cannot be trained adequately, and therefore, the prediction accuracy of O-SVM is lower. When the number of the training data is in the range of $[140, 240]$, the prediction accuracy of O-SVM becomes better than that of CBR. This is because that when the number of the training QoS data becomes larger, O-SVM can be trained adequately, so, it can make full use of its advantage of nonlinear simulation and get better prediction accuracy. When the number of the training QoS data is larger than 300, the prediction accuracy of O-SVM and improved CBR are approximately the same. In this experiment, we find that the best prediction accuracy of CBR and O-SVM are 86.07% and 87.86%, respectively.

The prediction time of the two methods is depicted in Fig. 5, where the vertical coordinate indicates the running time and the horizontal coordinate indicates the number of the training QoS data. From the figure we can find that the prediction speed of CBR is faster than that of O-SVM, particularly when the number of the training QoS data becomes larger. From the above experimental results we can see that the prediction accuracy and the effectiveness of the two context-aware real-time QoS estimation methods are satisfactory. We also notice that the training time of O-SVM grows steadily with the increase of the number of training QoS data. Fortunately, O-SVM is trained offline, and the training time does not affect its prediction effectiveness.

7.4 Performance Verification of the Context-aware QoS Prediction Methods

This experiment aims to verify the performance of the two context-aware QoS prediction methods in the second scheme.

7.4.1 Verification of Workload Prediction Method

In this experiment, we set the number of time points before the prediction time ft is 5, then, apply O-SVM to predict the workload for 12 different testing data sets. The experimental results are shown in Fig. 6.

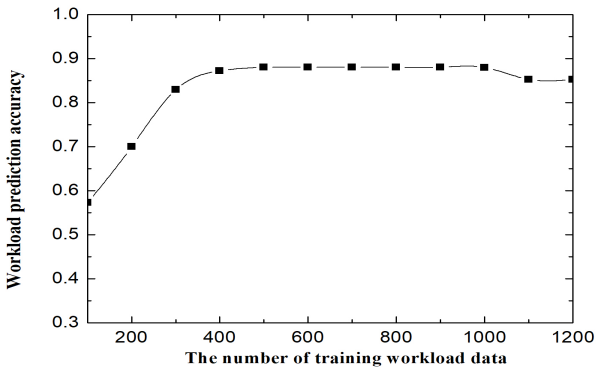

Fig. 6. Workload prediction accuracy

From Fig. 6 we can be find that, the accuracy of workload prediction increases with the increase of the training workload data, and the growth rate is faster in the early stage. When the number of training workload

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2944596, IEEE Transactions on Services Computing

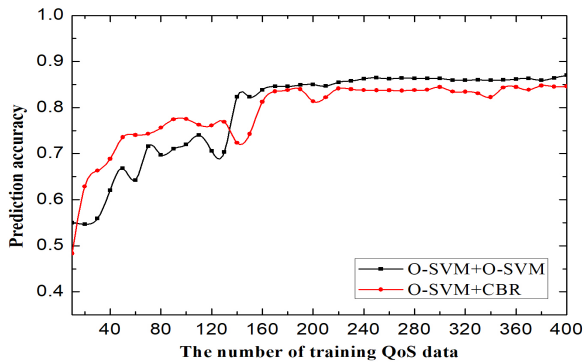

Fig. 7. Prediction accuracy of the two QoS prediction methods

data reaches 500, the prediction accuracy begins to stay stable. The reason is that when the training data set is too large, it leads to extra learning which may decrease the prediction accuracy. From the experimental results we find that when the training set reaches 500, a better workload prediction accuracy (88%) can be obtained. In the following experiment, we set the number of the workload training data as 500.

7.4.2 *Performance Verification*

In this experiment, we apply Algorithm 4 to predict the workload, and then, apply Algorithm 1 and Algorithm 3 to predict the multi-QoS and fitted QoS, respectively. During the experiment, prediction accuracy and prediction time of the two methods are recorded. The prediction accuracies of the two methods are illustrated in Fig. 7. In the figure, we use (O-SVM+CBR) to indicate the context-aware multi-QoS prediction method and (O-SVM+O-SVM) to denote the context-aware fitted QoS prediction method.

From Fig. 7 we can find that, when the number of the training QoS data is less than 140, the prediction accuracy of (O-SVM+CBR) is better than that of (O-SVM+O-SVM). When the number of QoS data is more than 140, the prediction accuracy of (O-SVM+O-SVM) becomes slightly better than that of (O-SVM+CBR). In this experiment, we find the best prediction accuracy of (O-SVM+CBR) and (O-SVM+ O-SVM) are 84.5% and 87%, respectively.

The prediction time of the two context-aware QoS prediction methods with different number of training QoS data is described in Fig. 8. From the figure we can find that, the prediction time of (O-SVM+CBR) and (O-SVM+O-SVM) increases with the growth of the number of the training QoS data. When the number of training QoS data is less than 140, the prediction time of (O-SVM+CBR) and (O-SVM+O-SVM) remains nearly the same. However, when the number of the training QoS data is larger than 140, the prediction time of (O-SVM+CBR) is less than that of (O-SVM+O-SVM). From the above experimental results we can conclude that the prediction accuracy and the effectiveness of the two future QoS prediction methods are acceptable.

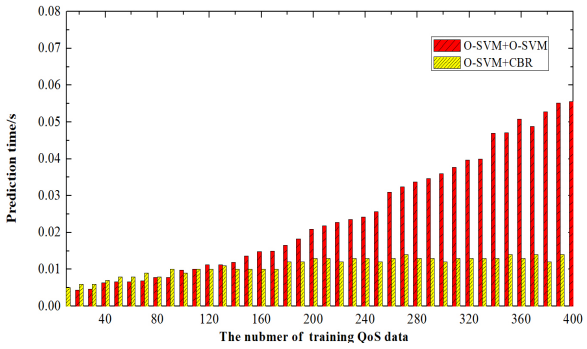

Fig. 8. Running times of the two QoS prediction methods

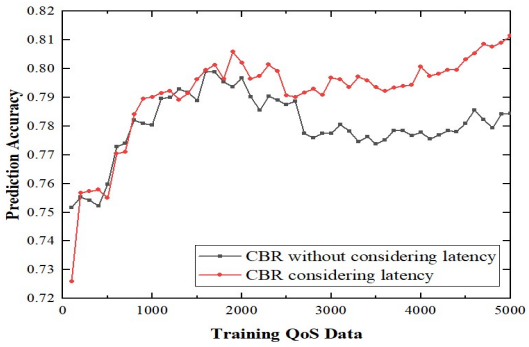

Fig. 9. Latency influence on context-aware real time QoS estimation

7.5 Influence Verification of Networking Latency

To verify the influence of the networking latency on QoS prediction for MEC services, we conduct an additional experiment on Edgecloudsim [44], and collect 60,000 QoS data. Contextual factors and QoS attribute evolve in each QoS data are *task type, task volume, workload and network latency* and *response time*, respectively. We apply the two context-aware real-time QoS estimation methods to conduct QoS prediction in this experiment. For the first method, 5,000 QoS data are randomly selected as the QoS case base, and the rest QoS data are randomly selected as the testing QoS data in the size of 100. Similarly, for the second method, 500 QoS data are randomly selected as the training QoS data, and the remaining QoS data are randomly selected as the testing QoS data in the group of 100. The experimental results are presented in Fig. 9 and Fig. 10. From the figures we can see that, considering the contextual networking latency is useful for improving the QoS prediction accuracy.

8 Conclusion

In the dynamic and complex environment of mobile edge computing (MEC), accurately predicting the quality of service (QoS) of MEC services has become a challenging task. Aiming at addressing the problem of context-aware and adaptive QoS prediction for MEC services, we firstly take into account contextual factors of both users and MEC services, and group different MEC service scheduling scenarios into four categories. Then, we propose two context-aware QoS prediction schemes. The first scheme features a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2019.2944596, IEEE Transactions on Services Computing

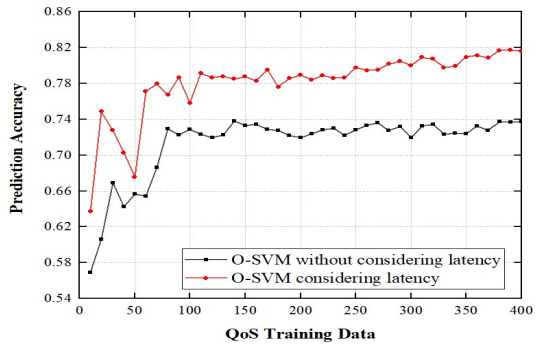

Fig. 10. Latency influence on context-aware real time QoS estimation

context-aware real-time multi-QoS estimation method and a context-aware real-time fitted QoS estimation method. The second scheme consists of a context-aware multi-QoS prediction method and a context-aware fitted QoS prediction method. Moreover, we develop strategies for adaptive QoS prediction. According to these strategies, the most suitable QoS prediction method will be selected to satisfy the needs of different MEC service scheduling scenarios. Our intensive experimental analysis indicates the effectiveness of the proposed approach.

Unlike existing QoS prediction research that only focuses on single service scheduling scenario, our approach is able to adaptively select the most suitable QoS prediction method to predict appropriate QoS data format for different MEC services scheduling scenarios. In our future work, we will study how to forecast the network speed in the dynamic and wireless environment, and take the predicted network speed as one contextual factor, thereby further improving the accuracy of QoS prediction. Moreover, we will investigate the impact of user mobility on QoS prediction for MEC services and develop mobility-aware QoS prediction method for MEC services.

Acknowledgements

This work is supported by: The National Key Research and Development Program of China (Grant no. 2018YFB1402500), National Natural Science Foundation of China (Grant nos. 61872126, 61832004 and 61772159), and Australian Research Council (ARC) Future Fellowship FT140101247.

References

[1] D. Weerasiri, M. C. Barukh, B. Benatallah, Q. Z. Sheng, and R. Ranjan, "A taxonomy and survey of cloud resource orchestration techniques," *ACM Computing Surveys*, vol. 50, no. 2, pp. 26:1–26:41, 2017.

[2] T. Tran, M. Hosseini, and D. Pompili, "Mobile edge computing: Recent efforts and five key research directions," *IEEE COMSOC MMTC Commun.-Frontiers*, vol. 12, no. 4, pp. 29–33, 2017.

[3] A. H. Ngu, M. A. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "Iot middleware: A survey on issues and enabling technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2017.

[4] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[5] A. T. N. Abbas, Y. Zhang and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[6] W. Yu, F. Liang, X. He, W. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the Internet of Things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[7] S. Wang, Y. Zhao, L. Huang, J. Xu, and C. Hsu, "QoS prediction for service recommendations in mobile edge computing," *Journal of Parallel and Distributed Computing*, 2017.

[8] R. Zhang, C. Li, H. Sun, Y. Wang, and J. Huai, "Quality of web service prediction by collective matrix factorization," in *Proc. of the IEEE International Conference on Services Computing*, (Anchorage, USA), pp. 432–439, IEEE, June 2014.

[9] H. Wu, Z. Zhang, J. Luo, K. Yue, and C. Hsu, "Multiple attributes qos prediction via deep neural model with contexts," *IEEE Transactions on Services Computing*, 2018 (early access).

[10] Q. Zhou, H. Wu, K. Yue, and C. Hsu, "Spatio-temporal context-aware collaborative qos prediction," *Future Generation Computer Systems*, vol. 100, pp. 46–57, 2019.

[11] H. Ma, Z. Hu, K. Li, and H. Zhu, "Variation-aware cloud service selection via collaborative qos prediction," *IEEE Transactions on Services Computing*, 2019 (early access).

[12] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized qos prediction for web services via collaborative filtering," in *Proc. of the IEEE International Conference on Web Services*, (Salt Lake City, USA), pp. 439–446, IEEE, July 2007.

[13] Z. Zheng, H. Ma, M. Lyu, and I. King, "Wsrec: A collaborative ltering based web service recommender system," in *Proc. of the IEEE International Conference on Services Computing*, (Los Angeles, USA), pp. 437–444, IEEE, July 2009.

[14] X. Wu, B. Cheng, and J. Chen, "Collaborative filtering service recommendation based on a novel similarity computation method," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 352–365, 2017.

[15] K. Chen, H. Mao, X. Shi, Y. Xu, and A. Liu, "Trust-aware and location-based collaborative filtering for Web service QoS prediction," in *Proc. of the 41st IEEE Annual Computer Software and Applications Conference (COMPSAC)*, (Turin, Italy), July 2017.

[16] Z. Chen, L. Shen, and F. Li, "Exploiting Web service geographical neighborhood for collaborative QoS prediction," *Future Generation Computer Systems*, vol. 68, pp. 248–259, 2017.

[17] H. Wu, K. Yue, C. Hsu, Y. Zhao, B. Zhang, and G. Zhang, "Deviation-based neighborhood model for context-aware QoS prediction of cloud and IoT services," *Future Generation Computer Systems*, vol. 76, pp. 550–560, 2017.

[18] J. Zhu, P. He, Z. Zheng, and M. Lyu, "Online QoS prediction for runtime service adaptation via adaptive matrix factorization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 10, pp. 2911–2924, 2017.

[19] S. Wang, Y. Ma, B. Cheng, F. Yang, and R. Chang, "Multi-dimensional QoS prediction for service recommendations," *IEEE Transactions on Services Computing*, vol. 12, no. 1, pp. 47–57, 2019.

[20] T. Jirsik, S. Trcka, and P. Celeda, "Quality of service forecasting with lstm neural network," in *Proc. of the IEEE Symposium on Integrated Network and Service Management*, (Arlington, USA), pp. 8–12, IEEE, April 2019.

[21] Y. Hu, Q. Peng, X. Hu, and R. Yang, "Time aware and data sparsity tolerant Web service recommendation based on improved collaborative filtering," *IEEE Transactions on Services Computing*, vol. 8, no. 5, pp. 782–794, 2015.

[22] S. Ding, Y. Li, D. Wu, Y. Zhang, and S. Yang, "Time-aware cloud service recommendation using similarity-enhanced collaborative filtering and ARIMA model," *Decision Support Systems*, vol. 107, pp. 103–115, 2018.

[23] C. Yu and L. Huang, "A Web service QoS prediction approach based on time-and location-aware collaborative filtering," *Service Oriented Computing and Applications*, vol. 10, no. 2, pp. 135–149, 2016.

[24] J. Liu, M. Tang, Z. Zheng, X. Liu, and S. Lyu, "Location-aware and personalized collaborative filtering for web service recommendation," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 686–699, 2016.

[25] X. Wang, J. Zhu, Z. Zheng, W. Song, Y. Shen, and M. Lyu, "A spatial-temporal QoS prediction approach for time-aware web

service recommendation," *ACM Transactions on the Web*, vol. 10, no. 1, p. 7, 2016.
[26] S. Li, J. Wen, and X. Wang, "From reputation perspective: A hybrid matrix factorization for qos prediction in location-aware mobile service recommendation system.," *Mobile Information Systems*, 2019.
[27] 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.
[28] H. Wu, K. Yue, B. Li, B. Zhang, and C. Hsu, "Collaborative QoS prediction with context-sensitive matrix factorization," *Future Generation Computer Systems*, vol. 82, pp. 669–678, 2018.
[29] M. Tang, Z. Zheng, G. Kang, J. Liu, Y. Yang, and T. Zhang, "Collaborative web service quality prediction via exploiting matrix factorization and network map," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 126–137, 2016.
[30] P. Zhang, Q. Han, W. Li, H. Leung, and W. Song, "A novel QoS prediction approach for cloud service based on bayesian networks model," in *Proc. of the IEEE International Conference on Mobile Services (MS)*, (SanFrancisco, USA), June 2016.
[31] Z. Liu, D. Chu, Z. Jia, J. Shen, and L. Wang, "Two-stage approach for reliable dynamic Web service composition," *Knowledge-Based Systems*, vol. 97, pp. 123–143, 2016.
[32] Z. Liu, Z. Jia, X. Xue, and J. An, "Reliable Web service composition based on qos dynamic prediction," *Soft Computing*, vol. 19, no. 5, pp. 1409–1425, 2015.
[33] S. Wang, Y. Zhao, L. Huang, J. Xu, and C. Hsu, "QoS prediction for service recommendations in mobile edge computing," *Journal of Parallel and Distributed Computing*, 2017.
[34] M. Aazam, M. St-Hilaire, C. Lung, and I. Lambadaris, "Mefore: Qoe based resource estimation at fog to enhance qos in iot," in *Proc. of the IEEE International Conference on Telecommunications*, (Thessaloniki, Greece), pp. 1–5, IEEE, May 2016.
[35] A. Yousefpour, A. Patil, G. Ishigaki, I. Kim, X. Wang, H. C. Cankaya, Q. Zhang, W. Xie, and J. P. Jue, "Fogplan: A lightweight qos-aware dynamic fog service provisioning framework," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5080–5096, 2019.
[36] W. Xiong, Z. Lu, B. Li, Z. Wu, B. Hang, J. Wu, and X. Xuan, "A self-adaptive approach to service deployment under mobile edge computing for autonomous driving," *Engineering Applications of Artificial Intelligence*, vol. 81, pp. 397–407, 2019.
[37] S. Shiu and S. Pal, "Case-based reasoning: concepts, features and soft computing," *Applied Intelligence*, vol. 21, no. 3, pp. 233–238, 2004.
[38] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
[39] S. Gupta, R. Kambli, S. Wagh, and F. Kazi, "Support-vector-machine-based proactive cascade prediction in smart grid using probabilistic framework," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2478–2486, 2015.
[40] T. Wang, X. Ye, L. Wang, and H. Li, "Grid search optimized SVM method for dish-like underwater robot attitude prediction," in *Proc. of the Fifth International Joint Conference on Computational Sciences and Optimization*, (Harbin, China), June 2012.
[41] C. Wang, L. Sun, S. Yu, J. Huang, X. Wang, and H. Guo, "Method of detecting in coal mine disaster warning internet of things based on svm intruders optimized by genetic algorithm," *Progress in Mine Safety Science and Engineering II*, p. 171, 2017.
[42] S. Fei, M. Wang, Y. Miao, J. Tu, and C. Liu, "Particle swarm optimization-based support vector machine for forecasting dissolved gases content in power transformer oil," *Energy Conversion and Management*, vol. 50, no. 6, pp. 1604–1609, 2009.
[43] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied mathematics and computation*, vol. 214, no. 1, pp. 108–132, 2009.
[44] C. E. C. Sonmez, A. Ozgovde, "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3493, 2018.

Zhizhong Liu received his PhD degree in Computer Science from Hohai University in 2011. He did his post-doc at Harbin Institute of Technology from 2013 to 2017. He is currently an associate professor in School of Computer and Control Engineering, Yantai University and was a visiting scholar at Department of Computing, Macquarie University, Sydney, Australia from December 2017 to December 2018. He is the author of more than 30 papers. His research interests include Intelligent Services, Service Composition and QoS Prediction.

Quan Z. Sheng is a professor and Head of Department of Computing at Macquarie University, Sydney, Australia. His research interests include service oriented computing, distributed computing, Internet computing, and Internet of Things. He holds a PhD degree in computer science from the University of New South Wales (UNSW) and did his post-doc as a research scientist at CSIRO ICT Centre. He has more than 360 publications. Prof Quan Z. Sheng is the recipient of AMiner Most Influential Scholar Award on IoT (2019), ARC Future Fellowship (2014), Chris Wallace Award for Outstanding Research Contribution (2012), and Microsoft Fellowship (2003).

Xiaofei Xu is a professor at School of Computer Science and Technology, and Vice President of Harbin Institute of Technology (HIT). He received the PhD degree in computer science from HIT in 1988. His research interests include enterprise computing, services computing, Internet of services, and data mining. He is the associate chair of IFIP TC5 WG5.8, chair of INTEROP-VLab China Pole, Fellow of China Computer Federation (CCF), and vice director of the technical committee of service computing of CCF. He is the author of over 300 publications. He is a member of the IEEE and ACM.

Dianhui Chu is a professor and Head of Computer Science and Technology College of Harbin Institute of Technology (Weihai). He received his PhD degree in computer science from Harbin Institute of Technology in 2014. He is the author of more than 100 papers. His research interests include Service Computing, Cloud Computing and Internet of Things.

Wei Emma Zhang is currently a lecturer at School of Computer Science, the University of Adelaide. Her research interests include Internet of Things, text mining, data mining and knowledge base. She received her PhD degree in computer science from the University of Adelaide in 2017. She is the author more than 50 papers. She is the member of the IEEE and ACM.