Received April 25, 2019, accepted June 9, 2019, date of publication June 28, 2019, date of current version July 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2925778

OPRC: An Online Personalized Reputation Calculation Model in Service-Oriented Computing Environments

XIN DU[1], JIANLONG XU[1,2], WEIHONG CAI[1,2], CHANGSHENG ZHU[1], AND YINDONG CHEN[1]

[1]Department of Computer Science, Shantou University, Shantou 515063, China
[2]Key Laboratory of Intelligent Manufacturing Technology, Ministry of Education, Shantou University, Shantou 515063, China

Corresponding authors: Jianlong Xu (xujianlong@stu.edu.cn) and Weihong Cai (whcai@stu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702318, in part by the Science and Technology Planning Project of Guangdong Province under Grant 2019B010116001 and Grant 2016B010124012, in part by the Shantou University Scientific Research Start-Up Fund Project under Grant NTF18024, and in part by the 2018 Provincial and Municipal Vertical Coordination Management Science and Technology Planning Project.

ABSTRACT Cloud applications based on service-oriented architectures usually integrate many component services to implement specific application logic. In service-oriented computing environments, many Web services are provided for users to build service-oriented systems. Since the performance of the same Web service varies according to different users' perspectives, the users have to personally select the optimal Web services according to the quality-of-service (QoS) data observed by other similar users. However, users with a low reputation provide unreliable data, which has a negative impact on service selection. Moreover, the QoS data vary over time due to changes in user reputation; and therefore, how to calculate a personalized reputation for each user at runtime remains a substantial problem. To address this critical challenge, this paper proposes an online reputation calculation method, called the OPRC, to efficiently provide a personalized reputation for each user. Based on the users' observed QoS data, the OPRC employs MF and online learning techniques to calculate personalized reputations. To validate the approach, large-scale experiments are conducted, which contain two QoS attributes from 142 reliable users and 15 unreliable users. The results show that OPRC has high accuracy and effectiveness compared to other approaches.

INDEX TERMS Service-oriented systems, Web services, quality of service (QoS), user reputation, online learning.

I. INTRODUCTION

Service-oriented computing (SOC) provides a component-based computing paradigm by dynamically integrating its component systems through service composition [1]. As a relatively new SOC paradigm, a service-oriented system is different from traditional component-based systems and is composed of different software components provided by other organizations that can be invoked remotely through the Internet [2]. As one of the most common forms of services for implementing a service-oriented system, Web services have attracted significant attention. Web services can be combined in a loosely coupled manner to implement complicated functions. In SOC environments, service consumers (users) use Web services offered by providers to accomplish their tasks. However, in open and dynamic Internet environments, there will be many services competing to offer similar functionalities. Therefore, how to select optimal Web services for different users to build high-quality service-oriented systems becomes very important.

As a nonfunctional requirement, quality-of-service (QoS) is generally employed to select optimal Web services. However, the QoS of each Web service varies for different users. For example, due to the dynamics of the network, the response time and invocation failure rate are different when different users invoke the same Web service. Therefore, selecting services with personalized QoS values for different users should be considered. In fact, personalized QoS-based approaches have been applied to Web service composition [3], Web service selection [4], QoS prediction [5], and so forth.

Specifically, personalized QoS values can be obtained after the users invoke the Web services and provide the QoS attribute values [6], and other users can invoke personalized

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

This work is licensed under a Creative Commons Attribution 3.0 License. For more information, see http://creativecommons.org/licenses/by/3.0/

services based on these QoS values. However, whether these values are reliable is unknown. Generally, the trustworthiness of the QoS data depends on the reputation of the user. Users with high reputations will provide reliable QoS data; however, other users will provide untrusted data. For example, some users may simultaneously be service providers, and they may provide good QoS values for their own services and bad QoS values for their competitors' services [7], [8]. In addition, traditional personalized Web service selection methods focus mostly on geographical similarity [9]–[11]. However, if these similar users are untrusted users with low reputations, it will seriously interfere with other users making the best service selection. Therefore, trusted and untrusted users must be identified, and how users acquire reputations is a major problem.

Since user reputation cannot be obtained directly, it is generally acquired indirectly. In the recent literature, the existing personalized reputation calculation approaches [7], [8], [12], [13] for service-oriented systems focus primarily on calculating reputations offline. However, the intentions of users may be variable; for example, users may sometimes submit reliable data and at other times submit unreliable data. In other words, user reputations may fluctuate over time. In addition, the QoS data submitted by users are very sparse, making it difficult to accurately evaluate reputations. Moreover, because the data change over time, the computational model must be sufficiently adaptive. Therefore, accurate user reputations are difficult to obtain online. The major challenge lies in providing an online and accurate reputation calculation model to enable trustworthy service selection. On the one hand, online is foundational. Reputation calculation must be performed online and adaptively to select the best services for building high-quality service-oriented systems. On the other hand, accuracy is critical. Inaccurate calculation of reputation may result in poor service selection and low-quality service-oriented systems.

To address these problems, we exploit an online personalized reputation calculation model based on the QoS data submitted by users, named OPRC, to accurately calculate user reputations. In this model, we adopt a matrix factorization (MF) framework to address the problem of sparse data, and we employ an online learning method to address the changing data.

The main contributions of this paper are summarized as follows:

1) First, we address the problem of online reputation for each user, which can be used to select reliable candidate services at runtime.
2) Second, we propose a novel reputation calculation method in service-oriented systems, named OPRC, by leveraging MF and online learning.
3) Third, we conduct extensive experiments based on a large-scale, real-world dataset to evaluate the accuracy of our method compared with other state-of-the-art approaches.

The remainder of this paper is organized as follows. Section II reviews related work. Section III details our reputation model. Section IV presents the experimental results. The conclusions and future work are summarized in Section V.

II. BACKGROUND AND RELATED WORK

In the service-oriented environment, reputation is regarded as a metric of a service's future [12] and can be measured based on user feedback ratings [14], [15]. Many notable research efforts have been made by the academic community in recent years to provide accurate reputation calculations for service-oriented systems. We review some of these efforts only to provide context to state their limitations.

Wang *et al.* [12] proposed a reputation evaluation approach for Web service selection, and their reputation evaluation model included feedback checking, feedback adjustment, and feedback detection. They used QoS values to calculate user reputation through a statistical average approach. Li *et al.* [16] proposed a reputation measurement approach based on a user's context by calculating and weakening the effect caused by the user's context. They employed collaborative filtering to measure the reputation of each Web service. Mehdi *et al.* [17] proposed a QoS-aware trust model that leverages the correlation information among various QoS metrics, and they also proposed a reputation aggregation algorithm that took the credibility of the feedback's senders into consideration. Li *et al.* [18] proposed six reputation calculation approaches: L1-AVG, L2-AVG, L1-MAX, L2-MAX, L1-MIN, and L2-MIN. These approaches are based on convergence algorithms. Based on [18], Li *et al.* [13] proposed a topic-biased model (TBM) to calculate user reputation that is applied in rating systems. Xu *et al.* [8] and Li et al [19] also used the L1-AVG algorithm to compute user reputation values in Web services. These approaches can be effective under offline conditions; however, most of these works did not consider online conditions.

In recent years, some researchers have also focused on online measurements for reputation. Azzeh *et al.* [20] proposed an online reputation model that aggregates ratings based on a moving window that can capture the variability in the ratings over time. Gao and Zhou [21] proposed an iterative group-based ranking method to evaluate user reputation that is based on the weighted sizes of the user rating groups in online rating systems. Fu *et al.* [22] proposed a reputation measurement mechanism for online services based on dominance relationships. They modeled reputation measurement as a problem of finding a ranking that indicates the relationships among services.

However, in the online case, data variability also requires the computational model to be adaptive. Therefore, online learning technology, which can be used in situations when the algorithm must dynamically adapt to new patterns in the data or when the data are generated as a function of time, has received considerable attention [23]. In contrast to batch learning, online learning can update the best predictor for future data in each step. Mairal *et al.* [24] proposed an online algorithm based on stochastic approximations for Ling *et al.* [25] proposed a unified framework for computing the reputation score of a user in online rating systems, and they also combined an online learning algorithm with Song *et al.* [26] proposed online learning in large-scale contextual recommender systems to improve the learning speed when the numbers of users and items are large. Based on the aforementioned previous works, in this paper, we study an online calculation method for user reputations that employs online learning algorithms.

III. REPUTATION CALCULATION MODEL

In this section, we first introduce the problem of reputation calculation and the existing models for batch computing in calculating Web service reputations. Based on these models, we ameliorate the intricate contextual problems in Web services proposed in the above sections, and we propose an online reputation calculation method. The reputation calculation model that we construct follows the unified framework presented by Ling *et al.* [25], which contains a prediction model, penalty function and link function. Although this framework was proposed for a rating system, we provide a novel online reputation calculation model by utilizing its flexibility, and we demonstrate that it works well on Web services.

A. PROBLEM FORMULATION

Given a set of N users $U = \{u_1, u_2, \ldots, u_N\}$ and a set of M services $S = \{s_1, s_2, \ldots, s_M\}$, users' real-time QoS values on services form an $N \times M$ matrix Q, where the element in the ith row, jth column q_{ij} represents a certain user-side QoS property (e.g., response time) of the Web service observed by service user u_i. In practice, the QoS matrix is notably sparse since each user generally only invokes a handful of services. If user i did not previously invoke Web service s, we initialize $q_{ij} = null$.

Our goal for personalized reputation calculation is to employ the matrix $Q \in \mathbb{Q}^{n\times m}$ to calculate the reputation r_i for each of the users u_i. To demonstrate the reputation degree more clearly and conveniently, r_i requires that $0 \leqslant r_i \leqslant 1$, with a larger value indicating that the reputation of u_i is higher. In addition, the personalized reputation approach should be performed in an online, accurate, and scalable manner.

B. TRADITIONAL ALGORITHMS AND THEIR LIMITATIONS

To compute the reputation of users in a rating system, Li *et al.* [18] proposed a class of six algorithms, including the classic algorithms L1-AVG and L2-AVG. These algorithms were proven to converge to a unique value in theory, which is the ideal attribute of a reputation evaluation algorithm. However, before Xu *et al.* [8] used these algorithms to calculate user reputation in Web services, they had been used in rating systems because of the limitations of these algorithms. Similarly, a series of models based on collaborative filtering or matrix factorization have been implemented in rating systems [27], [28].

Our personalized reputation calculation problem in Web services differs from rating systems mainly in that the QoS values are time-varying, while rating values remain unchanged over time. In addition, the coherent value range of ratings (e.g., 1∼5) in rating systems has enormous discrepancies with respect to QoS values (e.g., 0∼20s for response time and 0∼7000 *kbps* for throughput).

Due to these reasons, the limitations of traditional methods of computing user reputations of Web services are as follows:

1) The QoS data that we use to calculate reputations are updated in real time. Traditional models have to entirely retrain all the data when adapting to newly observed data, which requires considerable time and space overhead and is clearly unsuitable for real-time updates.
2) Due to having larger variances than that of the rating distribution, the distributions of QoS data and different low-density data matrices will make the calculation of user reputations more challenging.

C. LOW-RANK MATRIX FACTORIZATION AND ONLINE LEARNING

In the above subsection, we introduced the limitations and challenges of traditional methods in computing user reputations using real-time QoS data in Web services. Inspired by the framework of rating systems [25], we introduce an online personalized reputation calculation model in the SOC environment and demonstrate its applicability in terms of the response time (RT) and throughput (TP).

In contrast to the rating system framework composed of a prediction model and functions, we merge the forecasting and computing processes to form our OPRC model to achieve better effects in SOC environments. Specifically, the density of the dataset matrix is sparse; thus, it is difficult to predict the unknown QoS value accurately and calculate the user reputations through functional methods. By calculating the errors between the real value and the predicted value, we adopt the median approach to obtain the deviation between a certain user's reputation and the actual reputation. Then, the reputation values of reliable users (i.e., $r_i = 1$) are subtracted from the deviation of the upper requirement, and the reputation value of each user is obtained. In this process, our OPRC model integrates the two technologies of low-rank MF and online learning and successfully calculates online user reputation by improving these two technologies.

1) LOW-RANK MATRIX FACTORIZATION

To address the limitation of data sparsity, we need to predict the missing QoS value. With the basic low-rank MF, we can fit the factor model into the user-item matrix to solve this problem. In an MF model, the most critical step is establishing an objective function under which two independent feature spaces can be reconstructed. In an SOC environment, the matrix $Q \in \mathbb{Q}^{N\times M}$ has a low-rank structure. In other words, Q has a rank of $L \ll \min\{M, N\}$, where M and N denote sets of users and services, respectively. Let $U \in \mathbb{Q}^{L\times N}$ indicate the latent user factors and $S \in \mathbb{Q}^{L\times M}$ denote latent service factors, which are decomposed as $Q = U^T S$. The dot product $U_i^T S_j$ is used to predict the QoS value of u_i on s_j, where $q_{ij} \approx U_i^T S_j$. In addition, regularization terms that penalize the norms of the results were added to avoid overfitting; thus, our target was to minimize the following loss function:

$$\zeta = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(q_{ij}-U_i^T S_j)^2+\frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_S}{2}\|S\|_F^2, \quad (1)$$

where I_{ij} is an indicator that equals 1 if q_{ij} is observed and 0 if q_{ij} is unknown. λ_S and λ_U are both small positive decimal numbers that control the extent of regularization, and $\|\cdot\|_F$ represents the Frobenius norm. For minimizing the loss function, the gradients of the reconstructed feature space U and S are computed as

$$\frac{\partial \zeta}{\partial U_i} = \sum_{j\in S_j} I_{ij}(U_i^T S_j - q_{ij})S_j + \lambda_U U_i, \quad (2)$$

$$\frac{\partial \zeta}{\partial S_j} = \sum_{i\in U_i} I_{ij}(U_i^T S_j - q_{ij})S_j + \lambda_S S_j, \quad (3)$$

by iterating in the following formulas to alternatively update on U_i and S_j until convergence:

$$U_i \leftarrow U_i - \eta\frac{\partial \zeta}{\partial U_i}, \quad (4)$$

$$S_j \leftarrow S_j - \eta\frac{\partial \zeta}{\partial S_j}, \quad (5)$$

where η is the learning rate that controls each iteration's change. Finally, the unknown QoS value Q_{ij} can be predicted by the dot product after obtaining the latent factors U_i and S_j.

2) ONLINE LEARNING

Inspired by the reputation calculation model of rating systems based on prediction, we proposed an online reputation calculation method that employs low-rank MF technology and an online learning algorithm to achieve effective results.

The existing QoS values, which we used, will be constantly updated with newly observed values or expire after a time period without updating. To address these limits and achieve online personalized reputation calculation in an SOC environment, we not only employ an online learning algorithm to keep the sequentially observed QoS data continuous and incrementally updating but also adopt the sigmoid function to map the value $U_i^T S_j$ into the range of [0, 1]. Then, the loss function corresponds to the following:

$$\zeta = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(\frac{q_{ij}-g_{ij}}{q_{ij}})^2 + \sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}\frac{\lambda_u}{2}\|U_i\|_2^2 + \sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}\frac{\lambda_s}{2}\|S_j\|_2^2 \quad (6)$$

where g_{ij} indicates the solution of $U_i^T S_j$ processed by the sigmoid function. In contrast to Equation (1), λ_u and λ_s are two parameters that control the degree of regularization for each QoS value observed by user u_i for invoking service s_j, and $\|\cdot\|_2$ denotes the Euclidean norm. We can obtain the update equations of each data point in the matrix Q as follows:

$$U_i \leftarrow U_i - \eta((g_{ij}-q_{ij})g'_{ij}S_j/q_{ij}^2 + \lambda_u U_i), \quad (7)$$

$$S_j \leftarrow S_j - \eta((g_{ij}-q_{ij})g'_{ij}U_i/q_{ij}^2 + \lambda_s S_j), \quad (8)$$

where η is still the learning rate and g'_{ij} indicates $g'(U_i^T S_j)$. When new data are observed at each moment, online updating will be performed on its corresponding factors at each iteration. u_i, s_j and q_{ij} can undergo small changes after observing new data; thus, the feature vectors U_i and S_j will also experience small changes. Through these methods, all the QoS values in the matrix can be predicted in real time, and this is also the application of low-rank MF and online learning in our model.

D. IMPROVEMENTS IN THE PREDICTION AND COMPUTATION OF USER REPUTATION

Through the above approaches, we can not only effectively calculate the predicted value of Q_{ij} but also update the feature vectors incrementally without retraining the whole model. However, when we use the predicted value to calculate the user's reputation, the aforementioned approach may not perform well when some users or services incessantly join or leave the environment because our online personalized reputation calculation model will acquire the reputation values of users by summarizing the set of unexpectedness (i.e., the distance between the model's prediction and the real observation). Consequently, we continue improving the prediction model to obtain the best personalized reputation evaluation results.

According to Equation (6), we easily obtain the following pairwise loss function:

$$\begin{cases} \ell(U_i, S_j) = \frac{1}{2}(E_{u_i}+E_{s_j})(\frac{q_{ij}-g_{ij}}{q_{ij}})^2 + \frac{\lambda_u}{2}\|U_i\|_2^2 \\ \qquad + \frac{\lambda_s}{2}\|S_j\|_2^2. \\ E_{u_i}+E_{s_j} = e_{u_i}/(e_{u_i}+e_{s_j}) + e_{s_j}/(e_{u_i}+e_{s_j}) = 1 \end{cases} \quad (9)$$

This equation is the loss function for only one QoS value observed by user u_i for invoking service s_j. Specifically, E_{u_i} and E_{s_j}, which are contingent on the consistent user u_i or service s_j's accuracy, control the step size at each iteration. e_{u_i} and e_{s_j} represent the average error of user u_i and the average error of service s_j, respectively. This effectively alleviates the nonconvergence or high errors of new users or services. We can deduce the following function:

$$\ell_{E_{u_i}}(U_i) = \frac{1}{2}E_{u_i}(\frac{q_{ij}-g_{ij}}{q_{ij}})^2 + \frac{\lambda_u}{2}\|U_i\|_2^2, \quad (10)$$

$$\ell_{E_{s_j}}(S_j) = \frac{1}{2}E_{s_j}(\frac{q_{ij}-g_{ij}}{q_{ij}})^2 + \frac{\lambda_s}{2}\|S_j\|_2^2, \quad (11)$$

where $\ell(U_i, S_j) = \ell_{E_{u_i}}(U_i) + \ell_{E_{s_j}}(S_j)$, and these are the loss functions corresponding to U_i and S_j for each pairwise data sample. When collecting newly observed QoS data, we calculate the moving average of each exponent for each iteration to update the parameters (i.e., e_{u_i} and e_{s_j}). The specific formulas are as follows:

$$e_{u_i} = \theta E_{u_i}|q_{ij} - g_{ij}|/r_{ij} + (1 - \theta E_{u_i})e_{u_i} \tag{12}$$

$$e_{s_j} = \theta E_{s_j}|q_{ij} - g_{ij}|/r_{ij} + (1 - \theta E_{s_j})e_{s_j} \tag{13}$$

where θ controls the impact of an average, with more weight given to the latest data. As the parameters E_{u_i} and E_{s_j} are updated in each iteration, we can compute the gradients in Equation (10) and Equation (11) to derive the following update equations:

$$U_i \leftarrow U_i - \eta((g_{ij} - q_{ij})g'_{ij}S_j/q_{ij}^2 + \lambda_u U_i)E_{u_i} \tag{14}$$

$$S_j \leftarrow S_j - \eta((g_{ij} - q_{ij})g'_{ij}U_i/q_{ij}^2 + \lambda_s S_j)E_{s_j} \tag{15}$$

We predict many online and accurate QoS values $\widetilde{Q}_{ij}$ assuming that user u_i invokes service s_j by U_i and S_j. The distance between the model's prediction and the real observation indicates how well the QoS value q_{ij} can be trained using our prediction model. We call this distance the unexpectedness for this value. The above model has predicted every entry of $\widetilde{Q}$ at each moment; thus, we can easily calculate the unexpectedness of all known QoS observations. The unexpectedness matrices are formed by subtracting the absolute value from the predictive matrix $\widetilde{Q}$ and the observed matrix Q. For addressing the highly skewed values in Q (i.e., shown in Sect. B) and obtaining the required reputation value in the range of [0,1], we adopt the Box-Cox transformation [29] and sigmoid function [30], respectively. The member value, which is denoted C_{ij}, in the unexpectedness matrix is defined as follows:

$$C_{ij} = \left|\frac{(Q_{ij})^\alpha - (Q_{min})^\alpha}{(Q_{max})^\alpha - (Q_{min})^\alpha} - \frac{e^{\widetilde{Q}_{ij}}}{1 + e^{\widetilde{Q}_{ij}}}\right|, \tag{16}$$

where the parameter α controls the extent of the transformation.

For service s_j, we summarize the set of unexpectedness by its performance into one quantity C_{s_j}, which represents the set of unexpectedness of the overall users. We take C_{s_j} to be the median, which can indicate the reasonable prediction error of reliable users for the service. The absolute value of the distance between this median and a user invoking this service is inversely related to the reputation of a user. Then, to calculate a user's reputation more accurately, we averaged the user's value invoking each service. The final reputation value of this user is as follows:

$$r_i = 1 - \frac{1}{|O_i|} \sum_{s_j \in O_i} |C_{ij} - \underset{u_i \in H_j}{med}(C_{s_j})| \tag{17}$$

where O_i denotes the set of services invoked by user i, H_j represents the set of users that invoke service j, and $|O_i|$ is the number of services invoked by user i. r_i is the ith user reputation, which is the result of our OPRC model.

Algorithm 1 OPRC Algorithm

```
Require: Observed QoS data samples: $(t_{ij}, u_i, s_j, Q_{ij})$, and all
    the model parameters.
Ensure: user's reputation $r_i$.
 1: while Collect newly observed QoS data do
 2:   Randomly initialize $U_i \in \mathbb{Q}^L$ or $S_j \in \mathbb{Q}^L$, $I_{ij} \leftarrow 1$;
 3:   if pick an existing data sample then
 4:       if $t_{now} - t_{ij} < 15min$ then
 5:       Update $U_i$, $S_j$ synchronously by:
 6:           $U_i \leftarrow U_i - \eta((g_{ij} - q_{ij})g'_{ij}S_j/q_{ij}^2 + \lambda_u U_i)$
 7:           $S_j \leftarrow S_j - \eta((g_{ij} - q_{ij})g'_{ij}U_i/q_{ij}^2 + \lambda_s S_j)$
 8:       Update $\widetilde{Q}$ by $\widetilde{Q} = U^T S$: $\widetilde{Q}_{ij} \leftarrow (U_i, S_j)$
 9:       Update $r_i$ by Equation 16 and Equation 17:
10:                                   $r_i \leftarrow (Q_{ij}, \widetilde{Q}_{ij})$
11:       else the data sample is obsolete: set $I_{ij} \leftarrow 0$;
12:   if $u_i$ is a new user or $s_j$ is a new service then
13:       Initialize $e_{u_i} = 1$ or $e_{s_j} = 1$;
14:   Update $t_{ij}$, $Q_{ij}$ corresponding to $u_i$, $s_j$
15:   Update $E_{u_i}$, $E_{s_j}$ by:
16:       $E_{u_i} \leftarrow e_{u_i}/(e_{u_i} + e_{s_j})$
17:       $E_{s_j} \leftarrow e_{s_j}/(e_{u_i} + e_{s_j})$
18:   Update $e_{u_i}$, $e_{s_j}$ by Equations 12 and 13:
19:       $e_{u_i} \leftarrow (e_{u_i}, E_{u_i}, q_{ij}, g_{ij})$
20:       $e_{s_j} \leftarrow (e_{s_j}, E_{s_j}, q_{ij}, g_{ij})$
21:   Update $U_i$, $S_j$ synchronously by:
22:       $U_i \leftarrow U_i - \eta((g_{ij} - q_{ij})g'_{ij}S_j/q_{ij}^2 + \lambda_u U_i)E_{u_i}$
23:       $S_j \leftarrow S_j - \eta((g_{ij} - q_{ij})g'_{ij}U_i/q_{ij}^2 + \lambda_s S_j)E_{s_j}$
24:       Update $\widetilde{Q}$ by $\widetilde{Q} = U^T S$: $\widetilde{Q}_{ij} \leftarrow (U_i, S_j)$
25:       Update $r_i$ by Equation 16 and Equation 17:
26:                                   $r_i \leftarrow (Q_{ij}, \widetilde{Q}_{ij})$
27:   if converged
28:   then break;
29: end while
```

E. OPRC ALGORITHM

After a detailed analysis of the components of the OPRC model, we can obtain an approximate understanding of the algorithm. The pseudocode of our online update algorithm for OPRC is described in Algorithm 1. Specifically, at each iteration, the newly observed QoS data, which are collected online, can be divided into an existing data sample and a newly received data sample. According to the steps described in 1)∼3), when collecting the newly observed data sample $(t_{ij}, u_i, s_j, Q_{ij})$, we first determine whether the data sample exists or has a corresponding new user or service such that we can add it to our model (lines 4 ∼ 11) or model (lines 13 ∼ 26). We next determine whether an existing QoS value has expired (line 4), and if so, we discard this value. For example, in our experiment, we set the expiration time interval to 15 minutes. Another important point is that our

TABLE 1. Accuracy comparison on response time.

Method	density=5%		density=10%		density=15%		density=20%		density=25%		density=30%	
	RVAE	RVME	RVAE	RVME	RVAE	RVME	RVAE	RVME	RVAE	RVME	RVAE	RVME
L1-MAX	0.4928	0.4926	0.4937	0.4933	0.4941	0.4937	0.4944	0.4943	0.4947	0.4944	0.4949	0.4946
L2-MAX	0.4491	0.4476	0.4578	0.4575	0.4627	0.4640	0.4655	0.4675	0.4677	0.4689	0.4697	0.4721
L1-MIN	0.4967	0.4968	0.4957	0.4959	0.4956	0.4957	0.4946	0.4945	0.4942	0.4942	0.4935	0.4935
L2-MIN	0.4639	0.4657	0.4247	0.4256	0.3941	0.3945	0.3592	0.3585	0.3329	0.3346	0.3044	0.3020
L1-AVG	0.3077	0.3066	0.2963	0.2953	0.2927	0.2918	0.2909	0.2901	0.2898	0.2890	0.2891	0.2883
L2-AVG	0.3212	0.3204	0.2945	0.2834	0.2859	0.2845	0.2816	0.2803	0.2791	0.2778	0.2775	0.2762
OPRC model	**0.2496**	**0.2686**	**0.2489**	**0.2671**	**0.2477**	**0.2656**	**0.2465**	**0.2631**	**0.2446**	**0.2602**	**0.2420**	**0.2523**
Improve.(%)	18.88%	12.39%	15.48%	5.75%	13.38%	6.64%	12.46%	6.14%	12.36%	6.34%	12.79%	8.65%

model can scale to new users and services without retraining the whole model.

IV. EXPERIMENTAL SETUP

In this section, we conduct experiments to verify our OPRC model, and we discuss the parameters in the model. By comparing the results with those from other methods and different parameters in our model, we evaluate our OPRC model's accuracy and impact of parameters. All the experiments were conducted with a computer equipped with an Intel(R) Core(TM) i7-4790 CPU @ 3.40 GHz, 8 GB of RAM, Windows 8.1 (64 bit), and MATLAB 2016b.

A. DATASETS

For demonstrating the widespread applicability of our approach, we applied the method to two QoS attributes: RT and TP. RT denotes the time from when a user sends and receives requests, while TP indicates the data transfer rate (e.g., kbps) of a user invoking a service.

Specifically, in our experiments, we used the real-world dataset released by Zheng *et al.* [31] and Zhang [32]. Through uncomplicated treatment, this dataset includes both RT and TP values collected by 145 users invoking 4,500 Web services for 64 consecutive time slices at an interval of 15 minutes. These users include authentic and reliable users, which are acquired by machines (PlanetLab nodes) located in 22 countries. The services comprise 4,500 publicly available real-world Web services from 57 countries. To make the experiments more realistic, we mixed many randomly generated unreliable users with these realistic users. The number of added unreliable users may also impact the algorithms' performance; thus, we adjusted the proportion of unreliable users in the dataset to different levels: 2.07%, 4.14%, 6.21%, 8.28% and 10.34%. In our datasets, the response time is $0 \sim 20s$, and the throughput is $0 \sim 7000 kbps$.

B. EVALUATION METRICS

In this paper, by drawing on existing evaluation methods [33], we define the calculation average error (RVAE) and the median error of the reputation value (RVME) as follows:

$$\mathrm{RVAE} = \Big(\frac{\sum_{u_i=1}^{N_{re}}(1-r_i)}{N_{re}} + \frac{\sum_{u_i=1}^{N_{ur}}(r_i-0)}{N_{ur}}\Big)/2, \tag{18}$$

$$\mathrm{RVME} = \frac{\underset{u_i\in N_{re}}{median}(1-r_i) + \underset{u_i\in N_{ur}}{median}(r_i-0)}{2}. \tag{19}$$

RVAE is the mean of the reputation values' average error for reliable users and the reputation values' average error for unreliable users. RVME is the mean of the reputation values' median error for reliable users and the reputation values' median error for unreliable users. One property of our method is mapping the user's reputation r_i into the range [0,1]. The closer the value of a user's reputation is to 1, the more reliable the user is; the closer the value of a user's reputation is to 0, the less reliable the user is. According to this property, a smaller RVAE or RVME value means better accuracy. N_{re} and N_{ur} are the numbers of reliable users and unreliable users, respectively.

C. PERFORMANCE COMPARISON

We provide a comparison of our metrics obtained using different methods in Table 1 and Table 2. Concretely, in the two attributes (i.e., RT and TP), we compare our OPRC model with six algorithms that have been introduced for calculating reputation [18] to verify the superiority of the OPRC model in terms of accuracy. Note that although these methods can be used for accuracy comparisons, they cannot be used directly for runtime service adaptation in practice because of the aforementioned limitations of traditional algorithms.

As previously mentioned, in practical applications, since it is impossible for users to use all services, the available QoS data matrix is sparse. In this experiment, we randomly remove entries from the data matrix at each time slice to simulate the sparse situation. The approaches that contain our OPRC model are run on 6 different matrices, whose densities are 5%, 10%, 15%, 20%, 25%, and 30%, respectively. When we set the percentage of unreliable users to $\approx 10.37\%$ (15 unreliable users in all 145 users) for all the methods, we obtain the results in Table 1 for RT and in Table 2 for TP.

TABLE 2. Accuracy comparison on throughput.

Method	density=5%		density=10%		density=15%		density=20%		density=25%		density=30%	
	RVAE	RVME	RVAE	RVME	RVAE	RVME	RVAE	RVME	RVAE	RVME	RVAE	RVME
L1-MAX	0.4702	0.4692	0.4801	0.4805	0.4806	0.4802	0.4825	0.4807	0.4805	0.4805	0.4781	0.4768
L2-MAX	0.4053	0.4049	0.3979	0.3976	0.3955	0.3952	0.3942	0.3942	0.3935	0.3936	0.3931	0.3930
L1-MIN	0.4495	0.4477	0.4406	0.4434	0.4426	0.4380	0.4370	0.4315	0.4330	0.4274	0.4321	0.4227
L2-MIN	0.3760	0.3587	0.3672	0.3400	0.3531	0.3349	0.3535	0.3320	0.3509	0.3292	0.3495	0.3284
L1-AVG	0.2551	0.2543	0.2443	0.2430	0.2408	0.2401	0.2392	0.2382	0.2383	0.2377	0.2375	0.2368
L2-AVG	0.2559	0.2167	0.2217	0.1865	0.2176	0.1775	0.2175	0.1741	0.2163	0.1748	0.2166	0.1724
OPRC model	**0.1828**	**0.1569**	**0.1828**	**0.1546**	**0.1808**	**0.1527**	**0.1760**	**0.1496**	**0.1683**	**0.1416**	**0.1527**	**0.1285**
Improve.(%)	28.34%	27.60%	17.55%	17.10%	16.91%	13.97%	19.08%	14.07%	22.19%	18.99%	29.50%	25.46%

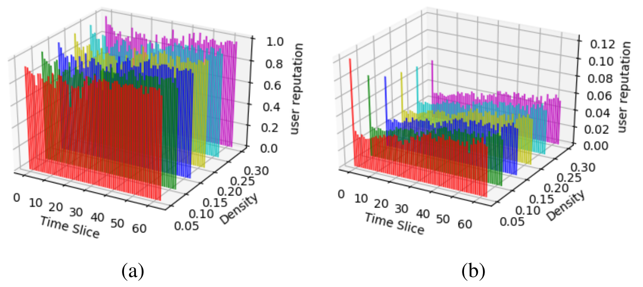

FIGURE 1. **Users on different time slices in diverse density. (a) Reliable user. (b) Unreliable user.**

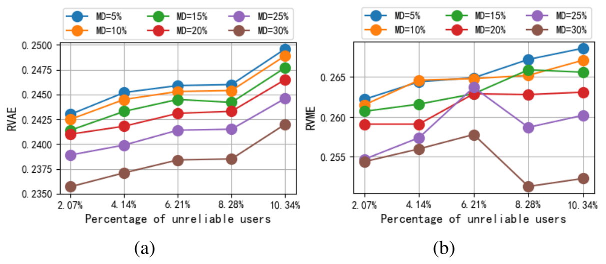

FIGURE 2. **Impacts of the percentage of unreliable users. (a) RVAE. (b) RVME.**

Specifically, for our OPRC method in this experiment, we set dimensionality $= 10$, $\lambda_U = \lambda_S = \lambda = 0.001$, $\theta = 0.3$, $\eta = 0.8$, $\alpha = -0.007$ for RT, and $\alpha = -0.05$ for TP. In addition, the other approaches' parameters are also optimized to achieve their optimal accuracy. These experimental results yield several important conclusions.

1) In terms of both RT and TP, our OPRC approach achieves smaller RVAE and RVRE values than those of other methods with different matrix densities, which verifies their high accuracy compared to existing approaches and demonstrates the effectiveness of our approach.
2) Clearly, we find that TP is a better indicator than RT, which is why we discuss the parameters that used the RT in the paper. Due to page limitations, we are convinced that the poorest performance indicator is more convincing, which is consistent with the reason why we present the results with the largest percentage of unreliable users.

For RT data, our OPRC model achieves an accuracy improvement of 12.36%∼18.88% on RVAE and 5.57%∼12.39% on RVME over the optimal value of other methods at different matrix densities. For TP data, the accuracy increased to 16.91%∼29.50% on RVAE and 13.97%∼27.60% at different matrix densities.

In addition, to further analyze the benefit of our OPRC model, Fig. 1 shows the reputations of a reliable user and another unreliable user on different time slices. We can observe that the value of the reliable user is much larger than that of unreliable users, which further illustrates the accuracy of our model. At the same density, in different time slices, the user reputations will change in real time, which also illustrates the real-time update ability of our model.

In the following experiments, we will study the impacts of the parameters on the performance of our models, including the percentage of unreliable users, λ_U, λ_S, the dimensionality, and the matrix density.

D. IMPACT OF THE PERCENTAGE OF UNRELIABLE USERS

To demonstrate the impact of the percentage of untrusted users on our reputation calculation model, we set the number of unreliable users as 3, 6, 9, 12, and 15 in our dataset. In other words, we keep 145 users unchanged, making the percentages of unreliable users 2.07%, 4.14%, 6.21%, 8.28% and 10.34% of the total, respectively. In these experiments, we set the parameters as follows: dimensionality $= 10$, λ_U and $\lambda_S = 0.001$, and matrix density varying from 5% to 30% with a step size of 5%. The experimental results are illustrated in Fig. 2.

Fig. 2 shows that the values of the evaluation metrics that contain both RVAE and RVME increase as the matrix density increases. For different percentages of unreliable users, when we evaluate the reputations of users, a denser matrix provides more information, which is beneficial for improving the accuracy. Another important point is that we obtain increasingly larger values of RVAE as the unreliable proportion increases, and the RVME values exhibit the same tendency when the matrix density (MD) is 25% or 30%. This result indicates that an increase in the number of unreliable users is negatively

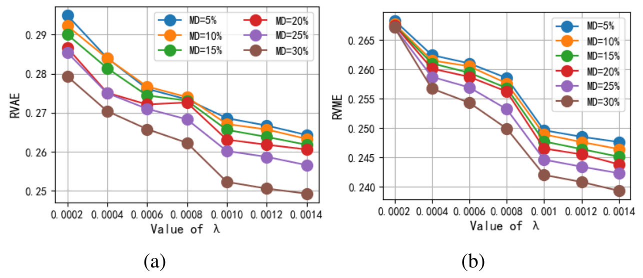
FIGURE 3. Impacts of λ_U and λ_S. (a) RVAE. (b) RVME.

correlated with the effectiveness of the reputation calculation for users.

E. IMPACTS OF λ_U and λ_S

The parameters λ_U and λ_S control the percentage of regularization, and they are used to adjust the accuracy and avoid overfitting. In this experiment, we set $\lambda_U = \lambda_S = \lambda$ for our OPRC model. If λ_U and λ_S are too small, the accuracy of the reputation evaluation will be unsatisfactory. In contrast, parameters that are too large result in overfitting. To demonstrate the impacts of λ_U and λ_S, we set the numbers to vary from 0.0002 to 0.0014 with a step size of 0.0002. In addition, we set the other parameters as follows: dimensionality = 10, matrix density varying from 5% to 30% with a step size of 5%, and percentage of unreliable users of 10.37%.

As shown in Fig. 3, for either the RVAE values or the RVME values with different densities, when the parameter λ gradually increases, the results will become increasingly smaller. However, note that before $\lambda = 0.001$, their values appear to be significantly smaller. After this point, although it has also decreased slightly, it has tended to stabilize very gently. More importantly, the larger λ is, the larger the risk of overfitting. Overall, we choose $\lambda_U = \lambda_S = \lambda = 0.001$ in all experiments.

F. IMPACT OF DIMENSIONALITY

The dimension denotes the rank of the low-rank assumption of MF, which is the number of latent features used to factorize the user-service matrix. Different dimension values will affect the prediction of the QoS value, the accuracy of our calculation and the efficiency of user reputation. To assess the impact of dimensionality, we vary the values of dimensionality from 5 to 30 with a step size of 5. For the other parameters, we set the percentage of unreliable users to 10.37%, select $\lambda_U = \lambda_S = 0.001$, and set the percentage of density to 5%, 10%, 15%, 20%, 25%, and 30%. Fig. 4 presents the experimental results.

Fig. 4 shows the impact of dimensionality on RVAM and RVME in our model. When the dimensionality is 10, the value of RVAE is the smallest and then slowly increases. Moreover, with respect to the values of RVME at different densities, we cannot find a distinct trend or numerical fluctuation in the dimensionality. Thus, we conclude that the effect of dimensionality is negligible, and we set dimensionality equal to 10 in the experiments.

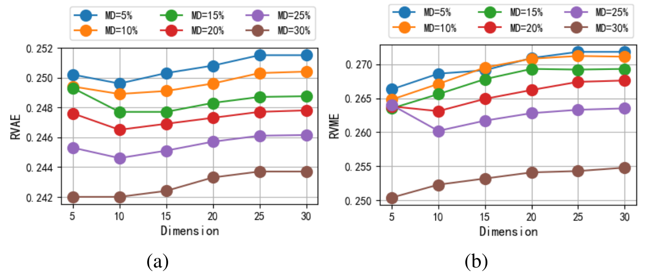
FIGURE 4. Impact of dimensionality. (a) RVAE. (b) RVME.

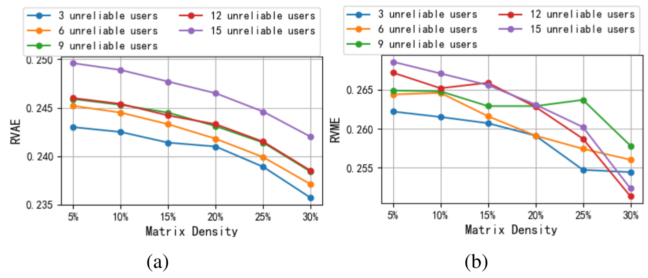
FIGURE 5. Impact of matrix density. (a) RVAE. (b) RVME.

G. IMPACT OF MATRIX DENSITY

In Web services, low-density data matrices make the calculation of user reputation more difficult. The matrix density is the proportion of unremoved entries in the user-service matrix. We change the density of the matrix from 5% to 30% with a step size of 5% to assess the impact of the matrix density. For the other parameters, we still select $\lambda_U = \lambda_S = 0.001$ and set the dimensionality to 10. Fig. 5 presents the experimental results.

Fig. 5 shows that irrespective of the percentage of unreliable users, as the matrix density increases, both RVAE and RVME first decrease. Then, as the matrix density increases, the RVAE decreases, while the rate of decrease in RVME slows. Therefore, more accurate user reputation results can be achieved by obtaining more QoS values.

V. CONCLUSION AND FUTURE WORK

In this paper, we aim to provide an online and accurate reputation calculation model for users to select highly reliable services in SOC environments. For this purpose, we propose an online reputation calculation model, called OPRC, for personalized reputation calculation at runtime. In OPRC, to accurately and effectively calculate reputation, we employ an MF model with an online learning technique. The extensive experimental results show that our proposed approach outperforms state-of-the-art approaches.

In the future, we plan to take additional information into account (e.g., geographical information) in order to achieve better performance.

REFERENCES

[1] M. P. Papazoglou, "Service-oriented computing: Concepts, characteristics and directions," in *Proc. 4th Int. Conf. Web Inf. Syst. Eng.*, Dec. 2003, pp. 3–12.
[2] Z. Zheng and M. R. Lyu, "Personalized reliability prediction of Web services," *ACM Trans. Softw. Eng. Methodol.*, vol. 22, no. 2, p. 12, 2013.

[3] A. G. Neiat, A. Bouguettaya, T. Sellis, and S. Mistry, "Crowdsourced coverage as a service: Two-level composition of sensor cloud services," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 7, pp. 1384–1397, Jul. 2017.
[4] M. B. Karimi, A. Isazadeh, and A. M. Rahmani, "QoS-aware service composition in cloud computing using data mining techniques and genetic algorithm," *J. Supercomput.*, vol. 73, no. 4, pp. 1387–1415, Apr. 2017.
[5] J. Zhu, P. He, Z. Zheng, and M. R. Lyu, "Online QoS prediction for runtime service adaptation via adaptive matrix factorization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 2911–2924, Oct. 2017.
[6] X. Zhu, X. Qin, and M. Qiu, "QoS-aware fault-tolerant scheduling for real-time tasks on heterogeneous clusters," *IEEE Trans. Comput.*, vol. 60, no. 6, pp. 800–812, Jun. 2011.
[7] W. Qiu, Z. Zheng, X. Wang, X. Yang, and M. R. Lyu, "Reputation-aware qos value prediction of Web services," in *Proc. IEEE Int. Conf. Services Comput.*, Jun./Jul. 2013, pp. 41–48.
[8] J. Xu, Z. Zheng, and M. R. Lyu, "Web service personalized quality of service prediction via reputation-based matrix factorization," *IEEE Trans. Rel.*, vol. 65, no. 1, pp. 28–37, Mar. 2016.
[9] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun, "Personalized QoS-aware Web service recommendation and visualization," *IEEE Trans. Serv. Comput.*, vol. 6, no. 1, pp. 35–47, 1st Quart., 2013.
[10] Y. Shen, J. Zhu, X. Wang, L. Cai, X. Yang, and B. Zhou, "Geographic location-based network-aware qos prediction for service composition," in *Proc. IEEE 20th Int. Conf. Web Services*, Jun./Jul. 2013, pp. 66–74.
[11] P. He, J. Zhu, Z. Zheng, J. Xu, and M. R. Lyu, "Location-based hierarchical matrix factorization for Web service recommendation," in *Proc. IEEE Int. Conf. Web Services*, Jun./Jul. 2014, pp. 297–304.
[12] S. Wang, Z. Zheng, Z. Wu, M. R. Lyu, and F. Yang, "Reputation measurement and malicious feedback rating prevention in Web service recommendation systems," *IEEE Trans. Serv. Comput.*, vol. 8, no. 5, pp. 755–767, Sep./Oct. 2014.
[13] B. Li, R.-H. Li, I. King, M. R. Lyu, and J. X. Yu, "A topic-biased user reputation model in rating systems," *Knowl. Inf. Syst.*, vol. 44, no. 3, pp. 581–607, 2015.
[14] W. Conner, A. Iyengar, T. Mikalsen, I. Rouvellou, and K. Nahrstedt, "A trust management framework for service-oriented environments," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 891–900.
[15] N. Limam and R. Boutaba, "Assessing software service quality and trustworthiness at selection time," *IEEE Trans. Softw. Eng.*, vol. 36, no. 4, pp. 559–574, Jul./Aug. 2010.
[16] W. Li, Q. Sun, and S. Wang, "Context-based Web service reputation measurement," in *Proc. IEEE 17th Int. Conf. Comput. Sci. Eng.*, Dec. 2014, pp. 1489–1496.
[17] M. Mehdi, N. Bouguila, and J. Bentahar, "Trust and reputation of Web services through qos correlation lens," *IEEE Trans. Serv. Comput.*, vol. 9, no. 6, pp. 968–981, Nov./Dec. 2016.
[18] R.-H. Li, J. X. Yu, X. Huang, and H. Cheng, "Robust reputation-based ranking on bipartite rating networks," in *Proc. SIAM Int. Conf. Data Mining*, 2012, pp. 612–623.
[19] S. Li, J. Wen, F. Luo, T. Cheng, and Q. Xiong, "A location and reputation aware matrix factorization approach for personalized quality of service prediction," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2017, pp. 652–659.
[20] M. Azzeh, "Online reputation model using moving window," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 4, pp. 508–512, 2017.
[21] J. Gao and T. Zhou, "Evaluating user reputation in online rating systems via an iterative group-based ranking method," *Phys. A, Stat. Mech. Appl.*, vol. 473, pp. 546–560, May 2017.
[22] X. Fu, K. Yue, L. Liu, Y. Feng, and L. Liu, "Reputation measurement for online services based on dominance relationships," *IEEE Trans. Serv. Comput.*, to be published.
[23] H. J. Kushner and G. G. Yin, *Stochastic Approximation and Recursive Algorithms and Applications*, vol. 35. Springer, 2003.
[24] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, Mar. 2010.
[25] G. Ling, I. King, and M. R. Lyu, "A unified framework for reputation estimation in online rating systems," in *Proc. IJCAI*, 2013, pp. 2670–2676.
[26] L. Song, C. Tekin, and M. van der Schaar, "Online learning in large-scale contextual recommender systems," *IEEE Trans. Serv. Comput.*, vol. 9, no. 3, pp. 433–445, May/Jun. 2016.
[27] J. Liu, M. Tang, Z. Zheng, X. F. Liu, and S. Lyu, "Location-aware and personalized collaborative filtering for Web service recommendation," *IEEE Trans. Services Comput.*, vol. 9, no. 5, pp. 686–699, Sep. 2016.
[28] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 8, pp. 30–37, Aug. 2009.
[29] R. M. Sakia, "The box-cox transformation technique: A review," *Statistician*, vol. 41, no. 2, pp. 169–178, 1992.
[30] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1257–1264.
[31] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed qos evaluation for real-world Web services," in *Proc. IEEE Int. Conf. Web Services*, Jul. 2010, pp. 83–90.
[32] Y. Zhang, Z. Zheng, and M. R. Lyu, "WSPred: A time-aware personalized QoS prediction framework for Web services," in *Proc. IEEE 22nd Int. Symp. Softw. Rel. Eng.*, Nov./Dec. 2011, pp. 210–219.
[33] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative Web service QoS prediction via neighborhood integrated matrix factorization," *IEEE Trans. Services Comput.*, vol. 6, no. 3, pp. 289–299, Jul. 2012.

XIN DU is currently pursuing the master's degree with the Department of Computer Science and Technology, College of Engineering, Shantou University, China. His research interests include service computing and machine learning.

JIANLONG XU received the Ph.D. degree from the South China University of Technology, in 2013. He was a Postdoctoral Fellow of the Shenzhen Research Institute, The Chinese University of Hong Kong. He is currently a Researcher with the College of Engineering, Shantou University, China. His research interests include service computing, the Internet of Things, and distributed computing.

WEIHONG CAI received the Ph.D. degree from the South China University of Technology, in 2012. He is the Head of the Institute for Modern Network Technology and Information Security Studies, Shantou University, and a Representative of the 10th Member Congress of China Computer Federation, and the Director General of the Shantou Computer Society. His research interests include network and communication technologies, information security and network management, and computer application systems.

CHANGSHENG ZHU graduated from Shantou University, China, where he is currently an Engineer with the Research Division. His research interests include big data and information security.

YINDONG CHEN received the Ph.D. degree in computer science from Fudan University, in 2010. He is currently a Faculty Member of the Department of Computer Science with Shantou University. His research interests include cryptology and information security.