Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 8680758, 12 pages
https://doi.org/10.1155/2018/8680758

Research Article

Using NearestGraph QoS Prediction Method for Service Recommendation in the Cloud

Yiqi Fu, Ding Ding, and Seid Ahmed

School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China

Correspondence should be addressed to Ding Ding; dding@bjtu.edu.cn

Received 2 March 2018; Revised 2 May 2018; Accepted 29 May 2018; Published 27 June 2018

Academic Editor: Pierre-Martin Tardif

With the advent of the mobile network, the fusion of cloud computing and fog computing is becoming feasible to promise lower latency and short-fat connection. However, there are a lot of redundant cloud-aware services with identical functionalities but a different quality of service (QoS) in the fog cloud environment. In fact, since QoS information is stored in distributed fog servers rather than remote cloud, it is hard for individuals to make recommendation and selection with sparse QoS information. Collaborative filtering is an important method for the sparsity problems and has been widely adopted on the prediction of missing QoS values. Focusing on the fact that existing researchers often ignore the QoS fluctuation in a wide range in the fog cloud environment, a novel neighbor-based QoS prediction method is proposed for service recommendation, in which a concept and calculation method is put forward to describe the stable status of services and users with quantifiable QoS values, and a NearestGraph algorithm is further designed to recognize stable or unstable candidate along with their popularity by a nearest neighbor graph structure which can help to make missing QoS values prediction in a certain order to improve final prediction accuracy. Experimental results confirm that the proposed method is effective in predicting unknown QoS values in terms of service recommendation accuracy and efficiency.

1. Introduction

As the development of mobile Internet and agility of distributed system services [1, 2], cloud computing is migrating to the fusion of cloud and fog computing since fog computing is able to better satisfy demands on lower latency and short-fat connection. At present, the composed distributed system [3] is becoming the main solution accepted by the majority [4]. However, a wide range of cloud-aware services is produced to cater for the fog cloud environment. In this situation, mobile users often feel confused to select proper cloud-aware services due to the appearance of redundant cloud-aware services with identical functionalities but a different quality of service (QoS) [5, 6]. Recommender systems are designed to address the suitable matching problem with mobile users and cloud-aware services under information overload.

The key to cloud-aware service selection and recommendation is QoS [7]. QoS is defined as a set of properties of specific cloud-aware services such as response-time, throughput, reputation, and the like, which is treated as an important criterion to distinguish among different functionally equivalent services [8]. In the fog cloud environment, QoS information is normally collected and stored in various fog servers, instead of being transferred to the remote cloud directly, due to the big volume of data and heavy transmission cost. In this situation, QoS information is always distributed but not centralized [9], which means QoS information is often sparse and unavailable for mobile users. Therefore, motivated by making an effective recommendation, it is a feasible way to complete missing QoS values by making predictions.

In fact, all roles in the fog cloud environment have the motivation to predict QoS before their assignments. A typical example of the fog cloud environment [10–12] is shown in Figure 1, which includes three roles, service user, service broker, and service designer. The same problem is happening to each role on how to manage cloud-aware services with high-quality performance. For example, service users expect more qualified services which respond more quickly while meeting basic functions. In general, it is necessary for the three roles in the fog cloud environment to predict QoS

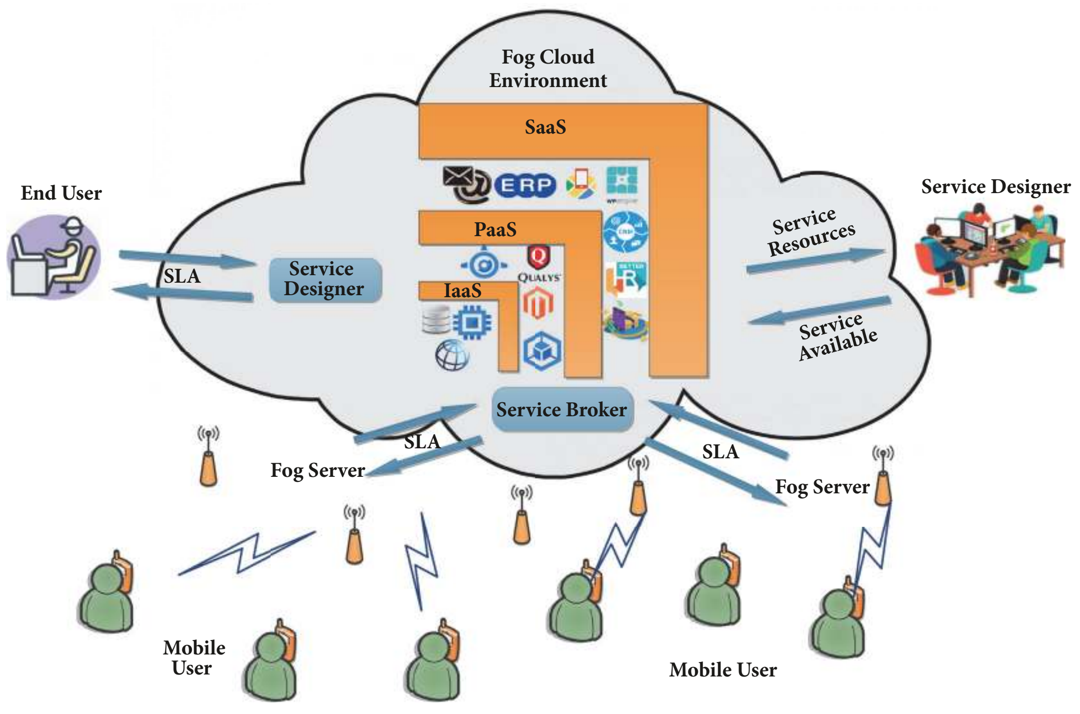


Figure 1: The architecture of fog cloud environment: each role wants to manage a cloud-aware service with "good" performance, especially QoS. However, QoS information is sparse and often varies among different roles. QoS prediction can achieve the goal of finding "good" performance through the analysis of historical QoS information.

values due to the following reasons: (1) service user can only get a limited number of QoS values caused by time-and-money-consuming QoS invocation, which makes it difficult for cloud-aware service recommender to make a decision, (2) service broker always has a strong desire to manage cloud-aware services with good performances, and (3) service designer needs to deploy cloud-aware services that satisfy QoS constraints to avoid punishment. Therefore, QoS prediction is a critical issue for cloud-aware service deployment, selection, and recommendation.

At present, the studies on QoS prediction have made certain progress in recent years. Many scholars prefer to "fill" the unknown QoS values through historical QoS information and formulate it as a matrix completion problem [13]. Chen et al. [14] take user-service geographical location into account to improve prediction accuracy. Wang et al. [15] introduce more QoS values affecting aspects such as time and location. Wu et al. [16] answer this problem by considering the relationship between similarity and candidate's consistency. Some other researchers devote them to finding solutions on how to improve the poor credibility of a fog cloud environment. Tang et al. [17] apply the trust concept for cloud-aware service QoS prediction. Su et al. [18] make a prediction for missing QoS values based on the trust relationship.

However, to the best of our knowledge, there is still a lack of research efforts explicitly targeting on the fluctuation of QoS values related to mobile users' status and services' status. In a highly dynamic Internet environment, QoS values of cloud-aware services often fluctuate in a large range due to the variety of users' mobile networking environments and physical distance between mobile users and fog servers. There is a current situation that some services perform more "unstable" according to a study on the real world QoS[19]. We select two services in random which is invoked by 339 users and draw their QoS values distribution as shown in Figure 2. We all feel service Y in blue is more unstable compared with service X in orange intuitively in Figure 2. Therefore, we can conclude that a cloud-aware service with a wide QoS range performance is not of general applicability and should not be recommended to other users since it is difficult to make an accurate prediction when candidate services with "unstable" performance are employed.

In this paper, the problem of QoS prediction is formulated to leverage historical QoS information. Inspired by the fact of QoS fluctuation, we propose a novel neighbor-based QoS prediction algorithm under the assumption that QoS values have a close relation with services and users in the fog cloud environment. In our approach, a concept and quantization method is put forward to represent the stable status of services and users in the fog cloud environment. And a graph structure is adopted to recognize stable or unstable candidate and to expose their popularity at the same time. Based on this, a NearestGraph method is used to generate an optimal prediction order to get the higher prediction accuracy.

The remainder of this paper is organized as follows. Section 2 introduces related works of QoS prediction and existing methods. Section 3 presents our proposed QoS prediction method for cloud-aware service recommendation. Section 4 provides our experimental results and the details of our experimental implementation. Section 5 sets out our conclusion and looks forward to future works.

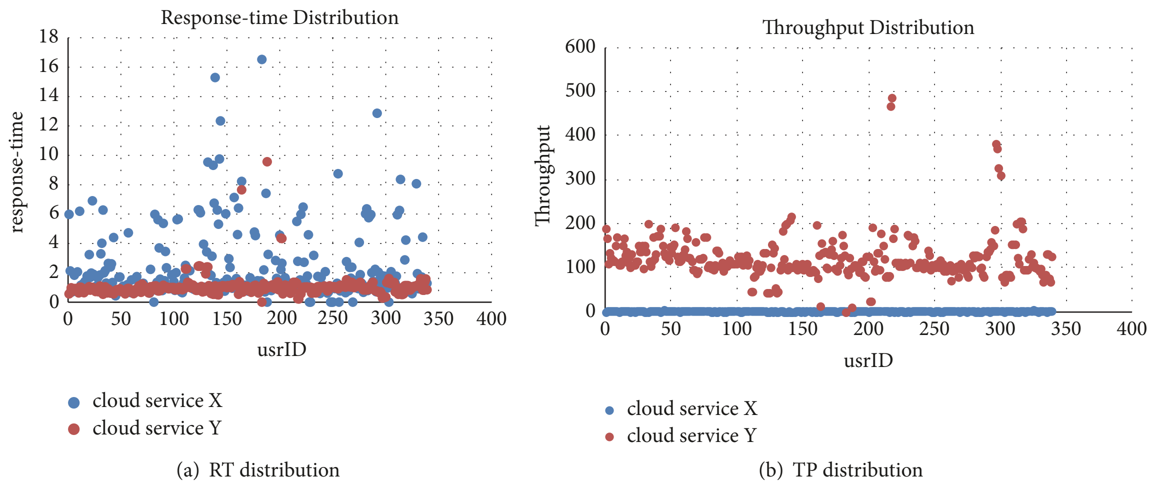


FIGURE 2: Dataset visualization.

2. Related Works

At present, there are a lot of efforts and results devoted to tackling the issue of QoS prediction. Initially, scholars adopt static methods to make a QoS prediction. Static methods use the arithmetic average value for prediction, including average QoS value from global, user, and service, respectively. These methods are simple and easy to implement, disregarding the situation-aware factors of users and services. Moreover, these static methods cannot reflect the dynamic properties of QoS values, which are leading to greater prediction error between predicted and actual values according to our experimental results in Section 4.

Motivated by the success of traditional recommender systems, existing works on QoS prediction in the fog cloud environment is usually based on collaborative filtering. Collaborative filtering (CF) methods are widely used to rate prediction in recommender systems. It exploits the similarity between users' experiences to predict user preference on unknown items. The intuitive idea is to identify "similar" users with the active user and to predict the active user's preference based on these similar users' feedback. CF can be further divided into two main categories: model-based method and neighbor-based method.

Model-based method makes a prediction from the known QoS values by learning a predictive model [20]. Observed values are used to learn two matrices which are the basis to calculate similarities among users. However, model-based method suffers from the ignorance of the low-rank structure of real world user-service matrices [21]. The main idea of the model-based method is based on matrix completion, in which the key is to exploit the low-rank structure of the user-service matrix. Lee et al. [22] present an algorithm for nonnegative matrix factorization indicating that there is only a small number of factors influencing the service performance. Some scholars think QoS has strong relation with time and put forward an online prediction. Zhu et al. [23] propose a method for running cloud-aware service to predict its QoS value.

The neighbor-based method uses QoS values of similar users or services to make QoS predictions directly. Shao et al. [24]first introduce a collaborative filtering approach for similarity mining and inference based on historical QoS information in the user-service matrix. They perform positive and negative user similarity calculations separately and integrate them using a weighted mean equation. Zhu et al. [25]give a QoS prediction approach based on multidimension, which takes timing constraints, QoS, throughput, fairness, and load balancing into account. Zheng et al. [5] propose a novel QoS ranking prediction model with the consideration of different cloud users having different preferences for different QoS attributes values.

In this paper, we mainly focus on the neighbor-based collaborative filtering since it is simple to implement and the prediction results are often easy to explain. The prediction accuracy of the neighbor-based method is highly influenced by the available similar candidates. Similar candidates play an effective role in QoS prediction phase mainly since they come from similar computation and assign more or less importance to the target in the prediction. However, the sparsity of QoS information always degrades the accuracy of QoS prediction. In our proposed approach, we address this challenge by introducing graph structure to expose candidate's own popularity.

3. NearestGraph QoS Prediction

In this section, the problem of QoS prediction is described and formulated in Section 3.1. After that, both user-user similarity and service-service similarity are computed in Section 3.2 to select neighbors. Our NearestGraph algorithm is presented in Algorithms 1 and 2 to predict missing QoS values at last.

3.1. Problem Description. In this paper, the problem of QoS prediction is described as follows: considering a fog cloud environment with m users and n cloud-aware services, the QoS of n cloud-aware services rated by m users is represented as an $m \times n$ matrix R. The entry $r_{i,j} \in \mathbb{R}^k$ is a k-dimensional

```
Input: usrG: user nearest neighbor graph; R: user-service matrix
Output: $usrSet(t)$: prediction order for user-based CF
1: $usrSet(t)$=[ ];
2: for each $i \in$ [0, usrG.vertices] do //add property usrWeight, usrIndegree to a graph usrG=(usrID,usrEdge)
3:     usrG←usrWeight given by Eq.(6);
4:     usrG←usrIndegrees given by usrG.inDegrees;
5: end for
6: repeat
7:     select usrID from usrG.vertices where usrG.vertices.Weight.max in {select * from usrG.vertices where
       usrG.vertices.Indegree.min};
8:     $usrSet(t).append$(usrID);
9:     usrG←usrIndegrees given by usrG.inDegrees;
10: until usrG.vertices.count is 0
```

Algorithm 1: NearestGraph algorithm for *usrSet*.

```
Input: usrG: service nearest neighbor graph; R: user-service matrix
Output: $servSet(t)$: prediction order for service-based CF
1: $servSet(t)$=[];
2: for each $i \in$ [0, serG.vertices] do //add property serWeight, serIndegree to a graph serG=(serID,serEdge)
3:     serG←serWeight given by Eq.(7);
4:     serG←serIndegrees given by serG.inDegrees;
5: end for
6: repeat
7:     select serID from serG.vertices where serG.vertices.Weight.max in {select * from serG.vertices where
       serG.vertices.Indegree.min};
8:     $serSet(t).append$(serID);
9:     serG←serIndegrees given by serG.inDegrees;
10: until serG.vertices.count is 0
```

Algorithm 2: NearestGraph algorithm for *serSet*.

vector representing the QoS values of k^{th} criteria. Let $U = \{u_1, u_2, \ldots, u_i, \ldots, u_m\}$, $i \in \{1, 2, \ldots, m\}$ be the set of m users, let $S = \{s_1, s_2, \ldots, s_j, \ldots, s_n\}$, $j \in \{1, 2, \ldots, n\}$ be the set of n cloud-aware services, Ω is set of all tuples $\{i, j\}$, and Λ is set of all unknown tuples (i, j), $r_{i,j} = \varnothing$. Then the missing information $\{r_{i,j} \mid (i, j) \in \Lambda\}$ is filled based on the existing information $\{r_{i,j} \mid (i, j) \in \Omega - \Lambda\}$. The order of filling in the matrix R is expressed as $r^{t}{}_{i,j} \in \Lambda, t \in (1, |\Lambda|)$ and there is an optimal order t to satisfy the higher accuracy.

Figure 3 shows a matrix R formed by m users and n cloud-aware services. The shaded part of the matrix indicates the user has invoked the cloud-aware service and has rated the corresponding QoS value. The blank part indicates the user has not invoked the cloud-aware service and the QoS values need to be predicted. The objective of the missing QoS value prediction is to make the user-service matrix denser within certain iteration phases [26].

Due to analysis of real world QoS datasets, QoS values can vary widely and are highly skewed with large variances that degrade the accuracy of prediction. Without loss of generality, we apply the following function to QoS data in order to map QoS values onto the interval $(0, 1)$.

$$r_{i,j} = \frac{r_{i,j} - r_{min}}{r_{max} - r_{min}} \tag{1}$$

where r_{min} and r_{max} are the minimum and maximum QoS values, respectively.

3.2. Neighbors Selection. We can find the neighborhood similarities of users and services by employing Pearson Correlation Coefficient (PCC). PCC is widely used in neighbor-based recommendation systems for similarity computation and proved to have high accuracy. In this paper, we adopt an enhanced-PCC method proposed by Zheng for the neighborhood similarity computation on both sets of users and services. The similarity between two users a and b is defined by the following equation:

$$\begin{aligned} \operatorname{sim}(a, b) &= \frac{2 \times |S_a \cap S_b|}{|S_a| + |S_b|} \\ &\times \frac{\sum_{i \in (S_a \cap S_b)} (r_{a,i} - \overline{r}_a)(r_{b,i} - \overline{r}_b)}{\sqrt{\sum_{i \in (S_a \cap S_b)} (r_{a,i} - \overline{r}_a)^2} \sqrt{\sum_{i \in (S_a \cap S_b)} (r_{b,i} - \overline{r}_b)^2}} \end{aligned} \tag{2}$$

where $\operatorname{sim}(a, b)$ falls into the interval $[-1, 1]$, $|S_a \cap S_b|$ is the number of cloud-aware services that are invoked by the two users, and $|S_a|$ and $|S_b|$ are the number of cloud-aware services

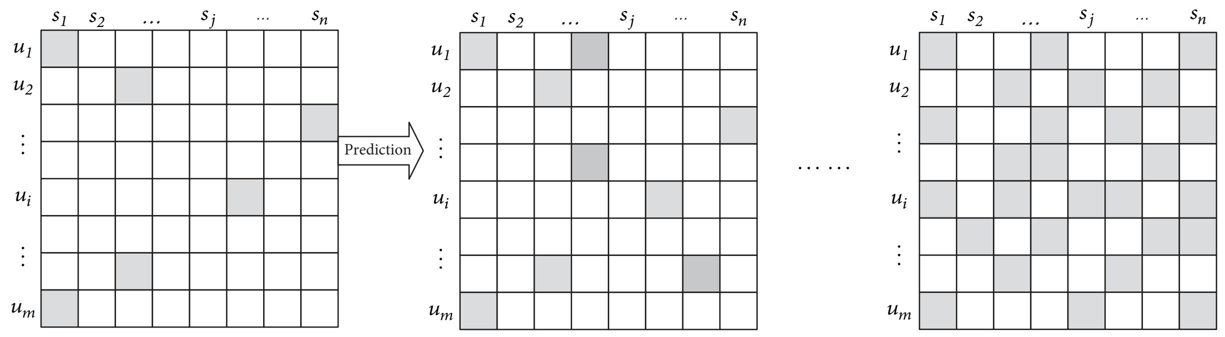


Figure 3: Prediction problem formulation.

invoked by user a and user b, respectively. $r_{a,i}$ is a vector of QoS values of cloud-aware service i observed by user a and $\overline{r}_a$ represents the average QoS values of different cloud-aware services observed by user a.

Similar to the user similarity computation, we also employ enhanced-PCC to compute the similarity between cloud-aware service x and y as follows:

$$\operatorname{sim}(x,y) = \frac{2\times\left|U_x\cap U_y\right|}{\left|U_x\right|+\left|U_y\right|} \times \frac{\sum_{u\in(U_x\cap U_y)}\left(r_{u,x}-\overline{r}_x\right)\left(r_{u,y}-\overline{r}_y\right)}{\sqrt{\sum_{u\in(U_x\cap U_y)}\left(r_{u,x}-\overline{r}_x\right)^2}\sqrt{\sum_{u\in(U_x\cap U_y)}\left(r_{u,y}-\overline{r}_y\right)^2}} \quad (3)$$

where sim(x,y) falls into the interval $[-1,1]$, $|U_x\cap U_y|$ is the number of users who invoked both cloud-aware services x and y previously, and $|U_x|$ and $|U_y|$ are the number of users who invoked cloud-aware services x and y, respectively. $r_{u,x}$ is a vector of QoS values of user u when he invokes cloud-aware service x and $\overline{r}_x$ represents average QoS values of different users when they invoke cloud-aware service x.

After the similarity computations, we can get the user similarity matrix and the service similarity matrix. At the same time, we can also identify their neighbors by similarity values in the ascending order. Traditional top-K algorithms select the top k most similar neighbors for making missing value prediction. In practice, some neighbors with negative similarity values could greatly decrease the prediction accuracy. In this paper, we exclude dissimilar neighbors with negative enhanced-PCC values. We employ the following equation to find a set of proper similar users for user i as Ψ_i:

$$\Psi_i = \left\{u_k \mid \operatorname{sim}(i,k)>0,\ \operatorname{rank}_i(k)\leqslant K,\ k\neq i\right\} \quad (4)$$

where rank$_i(k)$ is the ranking position of user k in the similarity neighbors of user i and K indicates the lowest ranking position manually.

In the same way, we can get the set of proper similar cloud-aware services for cloud-aware service j as Φ_j:

$$\Phi_j = \left\{s_k \mid \operatorname{sim}(j,k)>0,\ \operatorname{rank}_j(k)\leqslant K,\ k\neq j\right\} \quad (5)$$

where rank$_j(k)$ is the ranking position of cloud-aware service k in the similarity neighbors of cloud-aware service j and K indicates the lowest ranking position manually.

3.3. Predicting Missing QoS Values with NearestGraph. After user neighbors selection, we find an interesting fact that some users or services are relatively "popular" to others. For example, user E is on the top-1 similarity ranking position of user A. It also happens to user C when user E ranks top-1 in user C's similar neighbors. User B and User D may be confronted with the same situation. This is not an occasional case but happens for most similar neighbors. In order to expose this kind of popularity of users or services, we construct a directed graph by nearest neighbor graph as shown in Figure 4.

In Figure 4, a user is represented by `usrG=(usrID, usrEdge)`, in which `usrID` labels a user and `usrEdge` shows the relationship between the user and his most similar neighbor-a directed edge will line from a user to his most similar neighbor. Therefore, the indegree of a vertex in our nearest neighbor graph indicates the degree to which other vertices are in favor of this vertex. A vertex with larger indegree means it is very "popular" and will have a higher influence on other vertices. It can be understood as the relationship between celebrities and fans in social networking sites. A celebrity who has more fans means greater appeal, which reveals the greater influence at the same time. Similar method can be used to represent cloud-aware service by `servG=(servID, servEdge)`, in which `servID` labels a service and `servEdge` shows the relationship of a service and his most similar neighbor.

Furthermore, to reflect the stability of different users and cloud-aware services as shown in Figure 2, a concept of candidate stability is also proposed. We employ the following equation to describe the stability of user's status.

$$\operatorname{stability}(a) = \frac{\sum_{x\in S_a}\left(r_{a,x}-\overline{r}_a\right)}{\left|S_a\right|\times\overline{r}_a} \quad (6)$$

Similarly, we can describe the stability of cloud-aware service's status as follows.

$$\operatorname{stability}(y) = \frac{\sum_{b\in U_y}\left(r_{b,y}-\overline{r}_y\right)}{\left|U_y\right|\times\overline{r}_y} \quad (7)$$

where a smaller value of stability will indicate a more stable status.

In order to introduce the stability of users or services, we further extend above nearest neighbor graph

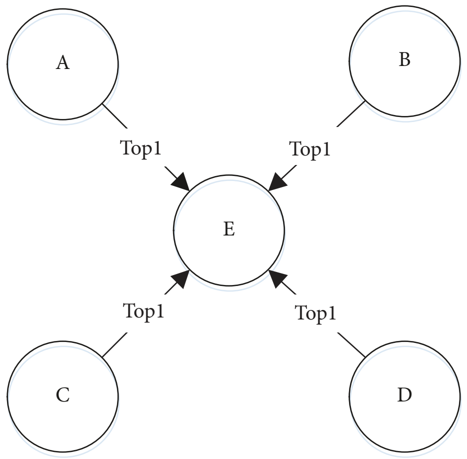


Figure 4: Nearest neighbor graph: we found that, in the sets of most similar neighbors for different users, some users tended to appear frequently.

to `usrG=(usrID, usrWeight, usrEdge)` and `servG=(servID, servWeight, servEdge)`, respectively, in which `usrWeigh` is the stability of user's status and `servWeigh` is the stability of service's status.

In this paper, we believe both popularity and stability will play an important role in QoS prediction and should be used to obtain better prediction accuracy. Therefore, we propose an algorithm called NearestGraph to achieve this goal, which can generate an optimal prediction order by nearest neighbor graph based on different popularity and stability. The main strategies of NearestGraph are the following three key points.

(1) Select the `usrID` with the minimum indegree.

(2) Select the `usrID` with the maximum weight if more than one vertex has the same indegree.

(3) Select the `usrID` with the minimum dictionary order if more than one vertex has the same indegree as well as weight.

The reason why we use those three rules comes from two facts: (1) those vertices with larger indegree, which means they are more popular, will affect more users and should be kept longer in our nearest neighbor graph to make full use of their important influence; (2) those vertices with larger weight, which means they are more stable, will have more positive impact on QoS prediction and should be kept longer in our nearest neighbor graph to make full use of their important influence. Now we take an example, shown in Figure 5, to illustrate the process of our NearestGraph algorithm.

Phase (a) is the initial state of nearest neighbor graph in which a property of vertex called `weight` is introduced to represent the status of stability and the directed edge is used to show the relationship between the user and his most similar neighbor. For example, vertex v_5 with weight 0.42 means it is more stable than vertex v_2 with weight 0.40, and vertex v_1 pointing to vertex v_2 expresses v_2 is the most similar neighbor of v_1. Then we will decide which vertex will be predicted according to the weight and indegree shown in nearest neighbor graph. According to No. 1 strategy of NearestGraph, v_1 will be predicted first since it has minimum indegree. In the next phase (b), there are two vertices with the same indegree after applying No. 1 strategy. Here No. 2 strategy can help us to make a decision in such a situation. v_5 should be predicted in phase (b) for its high stability. Then we loop through the three-key-point strategies to obtain a complete prediction order until there is only one vertex left in the graph structure. We can get a prediction order $usrSet$: $v_1 \Rightarrow v_5 \Rightarrow v_2 \Rightarrow v_4 \Rightarrow v_3 \Rightarrow v_7 \Rightarrow v_6$ in this example. The details of our NearestGraph algorithm for $usrSet(t)$ are as Algorithm 1.

Based on the prediction order generated by Algorithm 1, user-based method employs the values of entries to predict the missing entry $r_{i,j}$ in the user-service matrix as follows:

$$r^{U}_{i,j} = \overline{r}_i + \sum_{k\in\Psi_i} \frac{\text{sim}(u_i, u_k)}{\sum_{a\in\Psi_i}\text{sim}(u_i, u_a)} \times \left(r_{k,j} - \overline{r}_k\right), \quad (i,j) \in \{(\Omega - \Lambda) \cap usrSet\} \tag{8}$$

where $\overline{r}_i$ and $\overline{r}_k$ are the average existence QoS values of different cloud services rated by u_i and u_k, respectively.

We can also give the prediction order for $servSet(t)$ in a similar way as shown in Algorithm 2. And the values from service prediction order are correspondingly employed for prediction in service-based method as follows:

$$r^{S}_{i,j} = \overline{r}_j + \sum_{k\in\Phi_j} \frac{\text{sim}(s_j, s_k)}{\sum_{a\in\Phi_j}\text{sim}(s_j, s_a)} \times \left(r_{i,k} - \overline{r}_k\right), \quad (i,j) \in \{(\Omega - \Lambda) \cap servSet\} \tag{9}$$

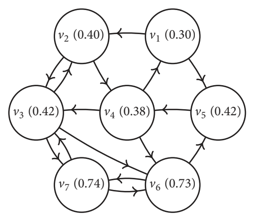
where $\overline{r}_j$ and $\overline{r}_k$ are the average existence QoS values of s_j and s_k rated by different users, respectively.

In this paper, both user-based and service-based approaches are adopted as follows:

$$r^{*}_{i,j} = \lambda \times r^{S}_{i,j} + (1-\lambda) \times r^{U}_{i,j} \tag{10}$$

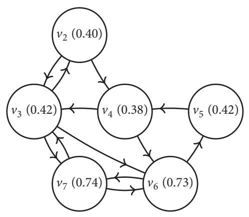
(1)

(a)



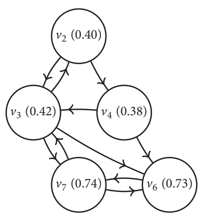
ID	weight	indegree
v_1	0.30	1
v_2	0.40	2
v_3	0.42	3
v_4	0.38	2
v_5	0.42	2
v_6	0.73	3
v_7	0.74	2

(b)



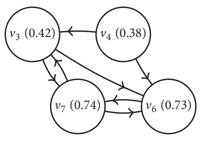
ID	weight	indegree
v_2	0.40	1
v_3	0.42	3
v_4	0.38	2
v_5	0.42	1
v_6	0.73	3
v_7	0.74	2

(c)



ID	weight	indegree
v_2	0.4	1
v_3	0.42	3
v_4	0.38	1
v_6	0.73	3
v_7	0.74	2

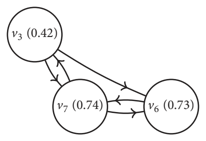
(d)



ID	weight	indegree
v_3	0.42	2
v_4	0.38	0
v_6	0.73	3
v_7	0.74	2

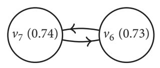
(2)

(e)



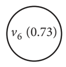
ID	weight	indegree
v_3	0.42	1
v_6	0.73	2
v_7	0.74	2

(f)



ID	weight	indegree
v_6	0.73	2
v_7	0.74	2

(g)



ID	weight	indegree
v_6	0.73	0

Prediction Order:

phase	ID
(a)	v_1
(b)	v_5
(c)	v_2
(d)	v_4
(e)	v_3
(f)	v_7
(g)	v_6

FIGURE 5: NearestGraph process.

The mixed approach can help us to get much more missing QoS values and therefore can improve the accuracy of prediction. The parameter λ controls how much fusion proportion of these two methods and can be trained on a sample dataset from the real world.

The complete QoS prediction algorithm is summarized in Algorithm 3.

4. Experiment

In this section, we evaluate the effectiveness of our proposed method on a distributed and parallel platform, Spark system. Section 4.1 introduces two typical metrics to assess the prediction accuracy. The comparison experiments on the prediction accuracy are conducted with different baseline algorithms in neighbor-based CF fields in Section 4.2 and three key parameters of NearestGraph on the prediction accuracy are further demonstrated in Sections 4.3, 4.4, and 4.5. All the experiments are conducted by using 4 PCs with i5-4460 CPU and 16G RAM as our hardware platform.

We evaluate the QoS prediction accuracy of our proposed method based on a real world QoS dataset which is widely used to evaluate the performance of QoS prediction. It contains response-time (response-time measures the time duration between user sending a request and receiving a response) and throughput (throughput stands for the data transmission rate of a user invoking a service) of 5828 services invoked by 339 distributed computers located in 30 countries from PlanetLab. According to statistics of this QoS dataset as shown in Table 1, the range of response-time and throughput are 0–20 s and 0–1000 kbps, respectively, and the means of response-time and throughput are 0.910 s and 47.386 kbps, respectively.

There are 100837 QoS records about response-time property and 143422 QoS records about throughput property in this QoS dataset. The corresponding user-service matrices on both these two QoS properties have some entries with the value of -1, which means the current QoS value cannot be obtained or the service is unreachable in the real world. Therefore, the entries with the value of -1 are where we need to predict in the matrix.

4.1. Metrics. To evaluate the performance of our proposed NearestGraph method, we compare its prediction accuracy with some neighbor-based CF methods by computing mean absolute error (MAE) and root-mean-square error (RMSE), which is to calculate the errors between predicted values and real values. The metric MAE is defined as

$$MAE = \frac{\sum_{i,j} \left| r_{i,j} - r^{*}_{i,j} \right|}{N} \tag{11}$$

```
Input: $R$: user-service matrix; $K$: lowest ranking position; $\lambda$: degree of fusion prediction results
Output: $R^*$
1: for all $(i, j) \in \Omega - \Lambda$ do
2:     compute the similarity $\mathrm{sim}(u_i, u_j)$ by Eq.(2)
3:     compute the similarity $\mathrm{sim}(s_i, s_j)$ by Eq.(3)
4: end for
5: for all $(i, j) \in \Omega - \Lambda$ do
6:     similar user set $\Psi_i$ by Eq.(4)
7:     similar service set $\Phi_j$ by Eq.(5)
8: end for
9: Learn usrSet by applying Algorithm 1
10: Learn servSet by applying Algorithm 2
11: for all $(i, j) \in \Lambda$ do
12:     compute $r^U{}_{i,j}$ by Eq.(8)
13:     compute $r^S{}_{i,j}$ by Eq.(9)
14:     fill by Eq.(10)
15: end for
```

Algorithm 3: QoS prediction algorithm.

Table 1: Statistics of QoS dataset.

Statistics	Response-Time(seconds)	Throughput(kbps)
Value Range	(0,20)	(0,1000)
Mean	0.910	47.386
Median	0.3320	11.07
Standard Variance	1.9320	107.4093
User Num	339	339
Service Num	5828	5828
Records Num	1974675	1974675

and RMSE is defined as

$$RMSE = \sqrt{\frac{\sum_{i,j}\left(r_{i,j} - r^*{}_{i,j}\right)^2}{N}} \quad (12)$$

where $r_{i,j}$ is the QoS value of cloud-aware service s_j observed by user u_i, $r^*{}_{i,j}$ is QoS value of cloud-aware service s_j that would be observed by user u_i as predicted by a method, and N is the number of predicted QoS values. According to the definitions, the smaller value of metric indicates the higher accuracy of prediction.

4.2. Performance Comparison. In this part, we conduct an overall comparison experiment on our NearestGraph method and some baseline algorithms in neighbor-based CF fields on both MAE and RMSE. They are listed as follows:

UMean: mean QoS values obtained by a user are used to predict the missing QoS value which has not been obtained by this user.

IMean: mean QoS values obtained by all users are used to predict the missing QoS value which has not been obtained by some users.

UPCC: it is a user-based collaborative filtering method, which uses similar users calculated by Pearson Correlation Coefficient to make a prediction [24].

IPCC: it is an item-based collaborative filtering method, which uses similar items calculated by Pearson Correlation Coefficient to make a prediction [27].

WSRec: it is a hybrid collaborative filtering method that combines IPCC and UPCC and uses both similar users and similar services for QoS prediction [5].

In order to simulate the users' invocation of cloud-aware services in the real world, we remove some entries from user-service matrix in random and compare their values with predicted ones. For example, 10% represents that we randomly remove 90% entries and use the remaining 10% entries to predict the values of removed entries. The parameter settings of NearestGraph are $top - K = 10$ and $\lambda = 0.5$ in the experiments.

Experiment results are shown in Table 2. We highlight the best performance of all methods for each row in Table 2. We can easily see from Table 2 that NearestGraph always obtains the minimum MAE and RMSE of response-time and throughput almost for all different matrix densities, which means it can improve the prediction accuracy. Moreover, with the value of matrix density increasing from 10% to 30%, the MAE and RMSE of NearestGraph method become smaller and smaller since a denser matrix will provide more information for the missing QoS value prediction.

Comparing the MAE and RMSE of response-time and throughput in Table 2, we can also find that the MAE and

Table 2: Performance comparisons.

Matrix Density(%)	Metrics	Response-Time (seconds)					
		UMean	IMean	UPCC	IPCC	WSRec	NearestGraph
10	MAE	0.8785	0.7015	0.6761	0.6897	0.6679	**0.6643**
	RMSE	1.8591	1.5813	1.4078	1.4296	1.4053	**1.4027**
20	MAE	0.8768	0.6867	0.5517	0.5917	0.5431	**0.5104**
	RMSE	1.8548	1.5342	1.3151	1.3268	1.2986	**1.2785**
30	MAE	0.8747	0.6818	0.5159	0.5037	0.4927	**0.4723**
	RMSE	1.8557	1.5311	1.2680	1.2252	1.1973	**1.1246**
Matrix Density(%)	Metrics	Throughput(kbps)					
		UMean	IMean	UPCC	IPCC	WSRec	NearestGraph
10	MAE	54.0084	29.2651	26.1230	29.2651	24.3285	**24.3269**
	RMSE	110.2821	66.6334	63.9573	64.2285	64.1908	**63.5435**
20	MAE	53.6768	27.3393	24.2695	26.8318	22.7717	**21.7493**
	RMSE	110.2977	64.3986	54.4783	60.0825	54.3701	**52.8731**
30	MAE	53.8792	26.6239	23.7455	26.4319	21.3194	**19.6530**
	RMSE	110.1751	64.3986	54.4783	57.8593	51.7768	**50.5765**

RMSE of response-time are larger than those of throughput which means NearestGraph performs better on throughput property than on response-time property. Taking the matrix density of 30% as an example, we can calculate the MAE improvements of NearestGraph over WSRec, respectively. For the response-time property, the MAE improvement is $((1.1973 - 1.1246)/1.1973) \times 100\% = 4.14\%$, while for the throughput property, the MAE improvement is $((21.3194 - 19.653)/21.3194) \times 100\% = 7.82\%$. That confirms that our proposed method focuses on facts of the QoS fluctuation and can make a better performance in a wide range of QoS values (just like the range of throughput is 0-1000 kbps and the range of response-time is only 0-20 s).

4.3. Impact of Matrix Density. In order to explore the impact of matrix density, we compare the prediction accuracy of all the methods under different matrix densities and present the results in Figure 6. The density of the matrix increases from 10% to 30% with a step of 10%. The parameter settings in this experiment are $top - K = 10$ and λ=0.5.

The MAE and RMSE results of response-time are shown in Figures 6(a) and 6(b) and the MAE and RMSE results of throughput are shown in Figures 6(c) and 6(d). In these figures, the green line NearestGraph stands for is always below any other lines, which means our proposed NearestGraph method gets the smallest values of MAE and RMSE under different matrix densities. Moreover, we can observe that the performance of our NearestGraph method improves with the increase of matrix density, which indicates that collecting more QoS information will greatly enhance prediction accuracy when the matrix is sparse.

4.4. Impact of λ. The parameter λ here controls how much fusion proportion of user-based and service-based method. A larger value of λ means user-based approach will contribute more to the hybrid prediction. In Figure 7, we study the impact of parameter λ in the proposed NearestGraph method on prediction accuracy by varying the values of λ from 0 to 1 with a step of 0.1 under the condition of $top - K$=10.

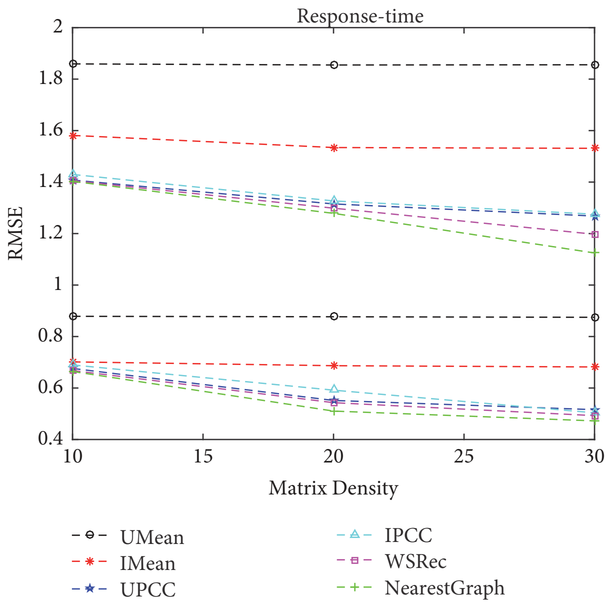
Figures 7(a) and 7(b) show the MAE and RMSE results of response-time and throughput, respectively. The prediction accuracies increase when we increase the value of λ at first. But when λ surpasses a certain threshold, the prediction accuracy decreases with the further increase of λ. From Figure 7, we can also find that NearestGraph gets the best performance when $\lambda \in [0.4, 0.7]$.

4.5. Impact of $Top - K$. The parameter $top - K$ determines the size of candidates sets including similar users and similar services. In Figure 8, we study the impact of parameter $top - K$ in the proposed NearestGraph method on prediction accuracy by varying the values of $top - K$ from 2 to 20 with a step of 2 under the condition of λ=0.5.

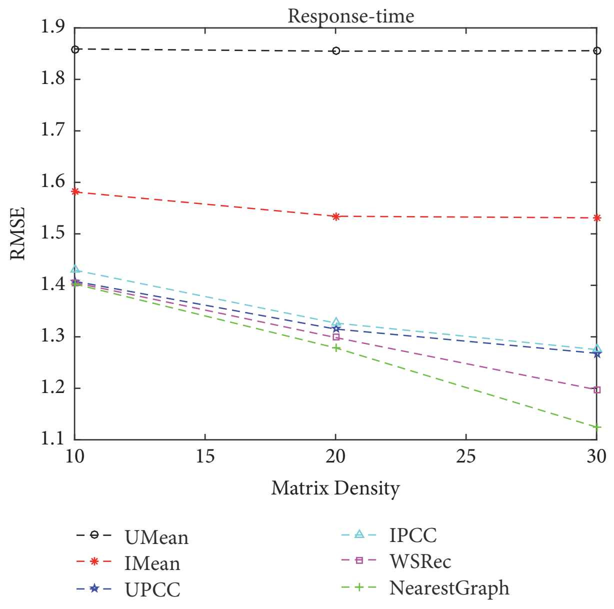
Figures 8(a) and 8(b) represent the MAE and RMSE results of response-time and throughput, respectively. The experimental results show that our NearestGraph will achieve best prediction accuracy (minimum MAE and RMSE) when $top - K$ is set around 10. This is because too small $top - K$ value will exclude useful information from some similar candidates, while too large $top - K$ value will introduce noise from dissimilar candidates, which will impact the prediction accuracy.

5. Conclusion and Future Work

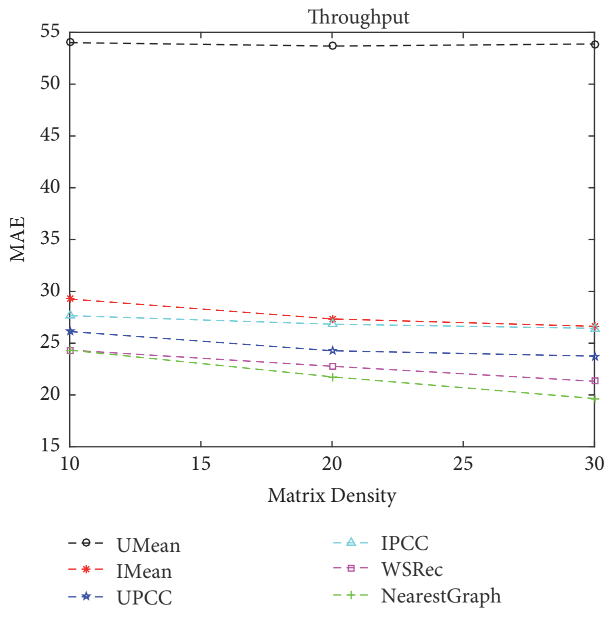
In the fog cloud environment, to reduce the data transmission cost from mobile users to the cloud, QoS information is often first handled by distributed fog servers instead of being sent to a remote cloud directly. However, such a cross-platform data distribution will lead to the sparsity of QoS information for service recommendation. Focusing on the fact that existing researches on missing QoS value prediction often ignore the QoS fluctuation in a wide range especially in the fog cloud environment, we propose a novel QoS prediction method by using NearestGraph algorithm for service recommendation.



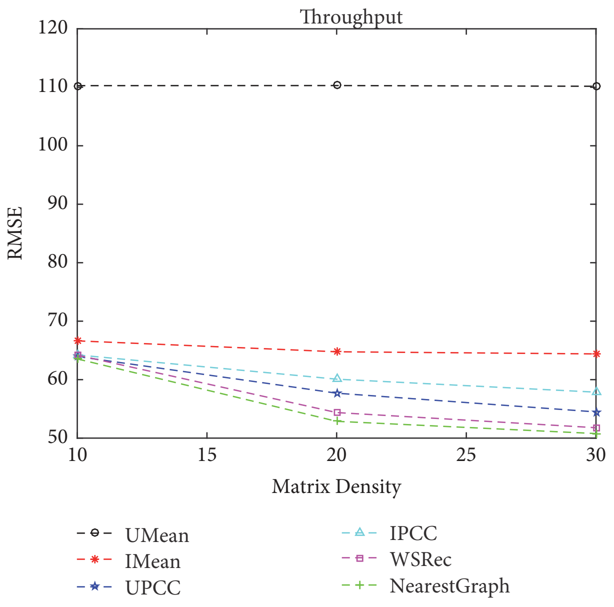
(a)



(b)



(c)



(d)

Figure 6: Impact of matrix density.

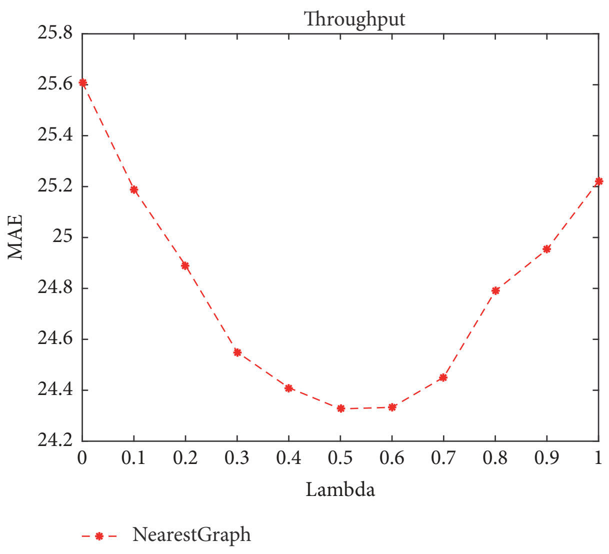
The key point of our approach proposed on the neighbor-based method is the construction of nearest neighbor graph which is designed to expose stable and popular candidates, and the choice of making prediction in a certain order, which applies priorities to different candidates instead of traversing candidates in random to promote the final accuracy. Through a set of experiments on a real world distributed service quality dataset *WS-DREAM* for stimulating the fog cloud environment, we validate the feasibility of our method in terms of service recommendation accuracy and confirm the motivation that NearestGraph can get a good performance in large fluctuation of QoS properties. In summary, the paper makes the following key contributions:

(1) We emphasize the fact of real world QoS values fluctuation in a wide range and take it into account to solve the inaccuracy of predicting missing values.

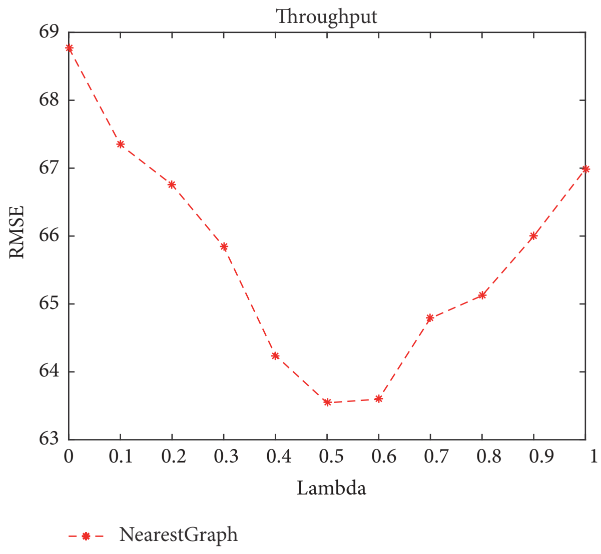
(2) We reveal the inner features of candidates behind neighbors and take their outer characteristic, stability, and popularity, in the fog cloud environment by constructing the nearest neighbor graph.

(3) Graph structure is employed to develop prediction order and enhance prediction accuracy.

Currently we predict the values of different QoS attributes separately. And we are going to investigate on the correlations

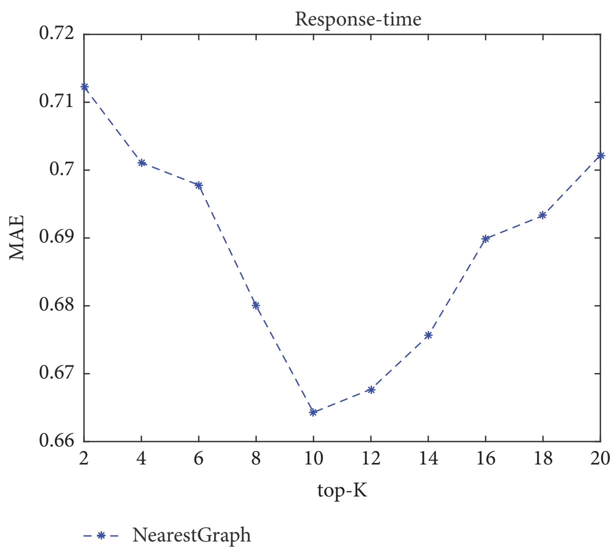


(a) Values of λ

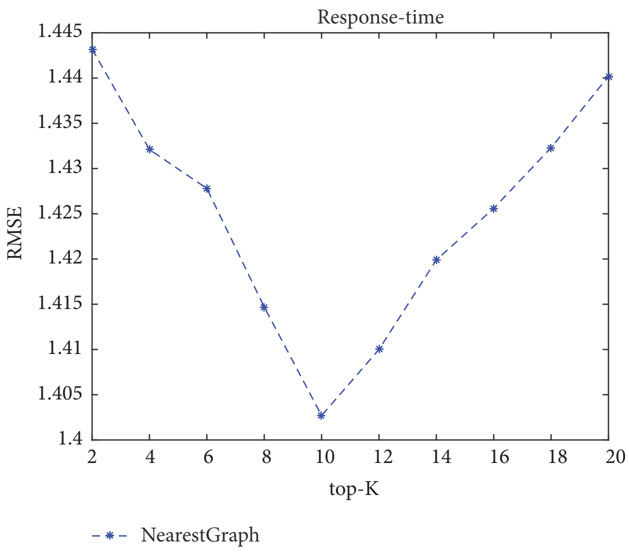


(b) Values of λ

FIGURE 7: Impact of λ.



(a) Values of $top - K$



(b) Values of $top - K$

FIGURE 8: Impact of $top - K$.

and combinations on the QoS attributes in the future. Furthermore, we will use time series analysis for prediction and extend NearestGraph to describe accurate user and service status in the fog cloud environment.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors acknowledge the support received from the National Natural Science Foundation of China (61300176, 61473031, 61472029, and 61672086) and Fundamental Research Funds for the Central Universities (2016JBZ005).

References

[1] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sunm, "Fog computing: Focusing on mobile users at the edge," 2015, https://arxiv.org/abs/1502.01815.

[2] S. K. Datta, C. Bonnet, and J. Haerri, "Fog Computing architecture to enable consumer centric Internet of Things services," in

Proceedings of the IEEE International Symposium on Consumer Electronics, ISCE 2015, Spain, June 2015.

[3] D. U. Gamage, "QoS and trust prediction framework for composed distributed systems, 2016".

[4] A. Yousefpour, A. Patil, G. Ishigaki et al., "QoS-aware Dynamic Fog Service Provisioning," 2018, https://arxiv.org/abs/1804.01796.

[5] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "QoS-aware web service recommendation by collaborative filtering," *IEEE Transactions on Services Computing*, vol. 4, no. 2, pp. 140–152, 2011.

[6] H. Yan and L. Zhi-Zhong, "Research on dynamic prediction method of QoS of cloud service," *Journal of Software Engineering*, vol. 11, pp. 1–11, 2017.

[7] D. Roongpiboonsopit, "Navigation Recommender: Real-Time iGNSS QoS Prediction for Navigation Services, 2011".

[8] D. A. Menascé, "QoS issues in web services," *IEEE Internet Computing*, vol. 6, no. 6, pp. 72–75, 2002.

[9] X. Wu, "Context-aware cloud service selection model for mobile cloud computing environments," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–14, 2018.

[10] Y. Zhao, Z. Li, and X. Chu, *QoS Prediction for the Cloud Service Marketplace: A Grassmann Manifold Approach*, IEEE, 2015.

[11] F. Liu, J. Tong, J. Mao et al., "NIST cloud computing reference architecture," National Institute of Standards and Technology NIST SP 500-292, 2011.

[12] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the 1st ACM Mobile Cloud Computing Workshop, MCC 2012*, pp. 13–16, August 2012.

[13] M. B. Senturk, "in Mission Critical Communication Networks," *in Mission Critical Communication Networks*, vol. 76, 2014.

[14] X. Chen, Z. Zheng, Q. Yu, and M. R. Lyu, "Web service recommendation via exploiting location and QoS information," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 7, pp. 1913–1924, 2014.

[15] X. Wang, J. Zhu, Z. Zheng, W. Song, Y. Shen, and M. R. Lyu, "A spatial-temporal qoS prediction approach for time-aware web service recommendation," *ACM Transactions on the Web (TWEB)*, vol. 10, no. 1, article 7, pp. 1–25, 2016.

[16] 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.

[17] M. Tang, X. Dai, J. Liu, and J. Chen, "Towards a trust evaluation middleware for cloud service selection," *Future Generation Computer Systems*, vol. 74, pp. 302–312, 2017.

[18] K. Su, B. Xiao, B. Liu, H. Zhang, and Z. Zhang, "TAP: A personalized trust-aware QoS prediction approach for web service recommendation," *Knowledge-Based Systems*, vol. 115, pp. 55–65, 2017.

[19] G. White, A. Palade, C. Cabrera, and S. Clarke, "Quantitative Evaluation of QoS Prediction in IoT," in *Proceedings of the 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2017*, pp. 61–66, usa, June 2017.

[20] Q. Yu, "CloudRec: a framework for personalized service Recommendation in the Cloud," *Knowledge and Information Systems*, vol. 43, no. 2, pp. 417–443, 2015.

[21] C. Bauckhage, "k-Means Clustering Is Matrix Factorization," 2015, https://arxiv.org/abs/1802.07891.

[22] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proceedings of the 13th International Conference on Neural Information Processing Systems (NIPS' 00)*, pp. 535–541, MIT Press, Denver, Colo, USA, 2000.

[23] J. Zhu, P. He, Z. Zheng, and M. R. 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.

[24] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized QoS prediction for web services via collaborative filtering," in *Proceedings of the IEEE International Conference on Web Services (ICWS '07)*, pp. 439–446, IEEE, Salt Lake City, Utah, USA, July 2007.

[25] X. Zhu and P. Lu, "A Multi-Dimensional scheduling scheme for QoS-Aware Real-Time Applications on heterogeneous clusters," in *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications, HPCC 2008*, pp. 205–212, chn, September 2008.

[26] Y. Zhang and M. R. Lyu, *QoS Prediction in Cloud and Service Computing*, SpringerBriefs in Computer Science, Singapore, Singapore, 2017.

[27] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*, pp. 285–295, 2001.