Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 452492, 9 pages
http://dx.doi.org/10.1155/2015/452492

Research Article

Online Optimization of Collaborative Web Service QoS Prediction Based on Approximate Dynamic Programming

Xiong Luo,[1,2] Hao Luo,[1,2] and Xiaohui Chang[1,2]

[1]*School of Computer and Communication Engineering, University of Science and Technology Beijing (USTB), Beijing 100083, China*
[2]*Beijing Key Laboratory of Knowledge Engineering for Materials Science, Beijing 100083, China*

Correspondence should be addressed to Xiong Luo; xluo@ustb.edu.cn

Received 19 December 2014; Accepted 30 January 2015

Academic Editor: Xiuzhen Cheng

More recently, with the increasing demand of web services on the World Wide Web used in the Internet of Things (IoTs), there has been a growing interest in the study of efficient web service quality evaluation approaches based on prediction strategies to obtain accurate quality-of-service (QoS) values. However, it is obvious that the web service quality changes significantly under the unpredictable network environment. Such changes impose very challenging obstacles to web service QoS prediction. Most of the traditional web service QoS prediction approaches are implemented only using a set of static model parameters with the help of designer's a priori knowledge. Unlike the traditional QoS prediction approaches, our algorithm in this paper is realized by incorporating approximate dynamic programming- (ADP-) based online parameter tuning strategy into the QoS prediction approach. Through online learning and optimization, the proposed approach provides the QoS prediction with automatic parameter tuning capability, and prior knowledge or identification of the prediction model is not required. Therefore, the near-optimal performance of QoS prediction can be achieved. Experimental studies are carried out to demonstrate the effectiveness of the proposed ADP-based prediction approach.

1. Introduction

Recently, with the increasing presence and adoption of cloud computing, the new idea of "anything as a service (XaaS)" is becoming more and more popular. XaaS enables the consumers to use the software with the form of "Use and Not Have". Therefore, it plays an important role in the applications of Internet of Things (IoTs). However, with the emergence of a huge number of cloud services, it is more and more difficult to choose an appropriate service in accordance with demand from the users. A number of web service composition and web service selection approaches have been proposed. It has led to the development of the service of computing (SOC) [1–4].

Obviously, only considering the services from the function has been unable to meet the requirements from users. Then, the service recommendation based on nonfunctional indexes (e.g., quality of service (QoS)) has become one of the attractive research fields in SOC [5, 6]. QoS represents the real user experience of a cloud service. Generally speaking, the QoS data from the user or server includes the availability, response time, throughput, delay, and delay variation and loss. While recommending a service based on the QoS data, one of the biggest problems is that the QoS data we have is not complete [7–12]. Actually, the QoS values of web services can be collected from the server side or the client side. At the server side, QoS values are usually provided and collected by the service providers. Here, we only focus on the QoS values measured at the client side. Due to the influence of the unpredictable network connections and complex user application environment on the Internet, QoS values vary widely at the client side. Thus, the web service evaluation is conducted for obtaining detailed and accurate QoS values at the client side. However, because there are huge amounts of web services on today's Internet, it will take too much time to evaluate all the web service candidates for service users. Therefore, it might be difficult or even impractical to fulfil the above web service evaluation task at the client side.

To obtain accurate web service QoS values on condition that there are no sufficient service evaluations at the client

side, some effective approaches with the help of prediction strategies are widely studied. The traditional QoS prediction algorithms are simple, and they are generally implemented by carrying out the arithmetic average operation for QoS values. Here, they use the average QoS values to predict the unknown QoS values. The major drawback of such methods is that they ignore some personalized factors, which may lead to a low prediction accuracy. In view of it, the collaborative filtering based approaches for making personalized QoS value prediction for the service users have been proposed. Specifically, the collaborative filtering technique is developed to automatically predict QoS values of the current user by collecting information from other similar users or items. In general, the collaborative filtering based QoS prediction approaches can be categorized into three major groups. The first group is the user-based collaborative filtering method using Pearson correlation coefficient (PCC), namely, UPCC. It is a very classical method. The QoS value prediction is implemented by employing similar users [13, 14]. The second group is the item-based collaborative filtering method using PCC, namely, IPCC. It is widely used in industrial companies (e.g., Amazon). This approach employs similar web services (i.e., items) for the QoS value prediction [15]. The third group is the probabilistic matrix factorization (PMF) based collaborative prediction. This idea was proposed by Salakhutdinov and Mnih [16], where the user preference matrix is fitted by a product of two lower-rank matrices. This method may perform well on some large, sparse, and imbalanced data sets. Furthermore, through the use of fusion techniques [17, 18], some advanced methods are also developed. For instance, Zheng et al. proposed a neighborhood-integrated matrix factorization (NIMF) approach by fusing the neighborhood-based and model-based collaborative filtering approaches to improve the prediction accuracy [19].

Although those approaches mentioned above play important roles in applications, there are also some limitations. Most of the QoS prediction approaches depend on designers' a priori knowledge about the prediction model parameters. It is obvious that the information about a prediction model or more specifically precise knowledge about a prediction system is quite difficult to obtain in some practical applications. Then, model parameter identification through experiments is usually needed. However, it is time consuming for some large-scale experiments. This practical limitation imposes challenging obstacles to the applicability of those QoS prediction methods. For instance, in [19], a series of experiments were conducted for the purpose of parameter identification used in QoS prediction. But it was computationally intensive for the scale of web service QoS data set. Furthermore, it was not suitable for online parameter tuning under complex network environment on the Internet.

In this paper, an optimal online parameter tuning methodology based on approximate dynamic programming (ADP) is proposed to improve the QoS prediction approach, and the experiment is conducted on a large-scale web service QoS data set that has some characteristics of big data. Because of the fast adaptability and approximation capabilities of neural network (NN) in its model-free reinforcement learning scheme, ADP using NNs is a powerful tool for computing optimal solution of a multistage dynamic decision process while avoiding the "curse of dimensionality" [20–23]. Under the ADP scheme with actor-critic architecture, a critic network is designed for approximating cost function and an action network is designed for generating optimal actions [24]. We propose a model-free online ADP learning algorithm for parameter tuning used in collaborative web service QoS prediction. By designing an ADP architecture and defining an appropriate reinforcement signal, the ADP tunes the QoS prediction model parameter optimally to achieve the satisfactory QoS prediction. In addition, for big data applications with large-scale data sets, some available QoS prediction approaches are time consuming in implementing model parameter identification by trial and error with painstakingly handcrafted exhaustion method. In contrast, the ADP-based method has automatic model parameter tuning capability and it may not be very computationally intensive for the scale of web service QoS data set. In this way, the ADP-based method may be a considerable alternative to big data analytics in this case.

Different from the traditional QoS prediction approaches, our methodology has the following advantages: (1) the optimal parameter tuner used in prediction approach can be implemented online; (2) no prior knowledge or identification of the prediction model is required; (3) the near-optimal performance may be achieved while the parameters of prediction model can adapt to the changes of network environment; and (4) the proposed structure and the associated algorithm can be also extended to applications for other types of QoS prediction model.

This paper is organized as follows. In Section 2, we introduce a collaborative QoS prediction framework using PMF model. An ADP-based parameter tuner for QoS prediction model is proposed in Section 3. Experiment results are presented in Section 4. Conclusion is provided in Section 5.

2. QoS Prediction Model and Parameter Analysis

2.1. Collaborative QoS Prediction Model. A collaborative QoS prediction framework is shown in Figure 1, where the service users are encouraged to share their individually observed past web service QoS information. In this collaborative framework, a service user will obtain the QoS prediction service from the centralized server only if he/she contributes some QoS values. Meanwhile, more web service QoS values are contributed by a service user; more user features can then be mined from those contributed data. In this way, higher QoS value prediction accuracy can therefore be achieved. It is the essence of this collaborative framework [19].

Furthermore, after providing the local QoS values to the server, the service users can get the prediction results via the following three phases. In the first phase, the system calculates the users' similarities using PCC and determines a set of top-K similar users (i.e., neighbors) for the current user. In the second phase, by employing those neighbors' information mentioned above, the system designs a collaborative filtering model to predict the missing web service QoS values in the user-item matrix, where each element

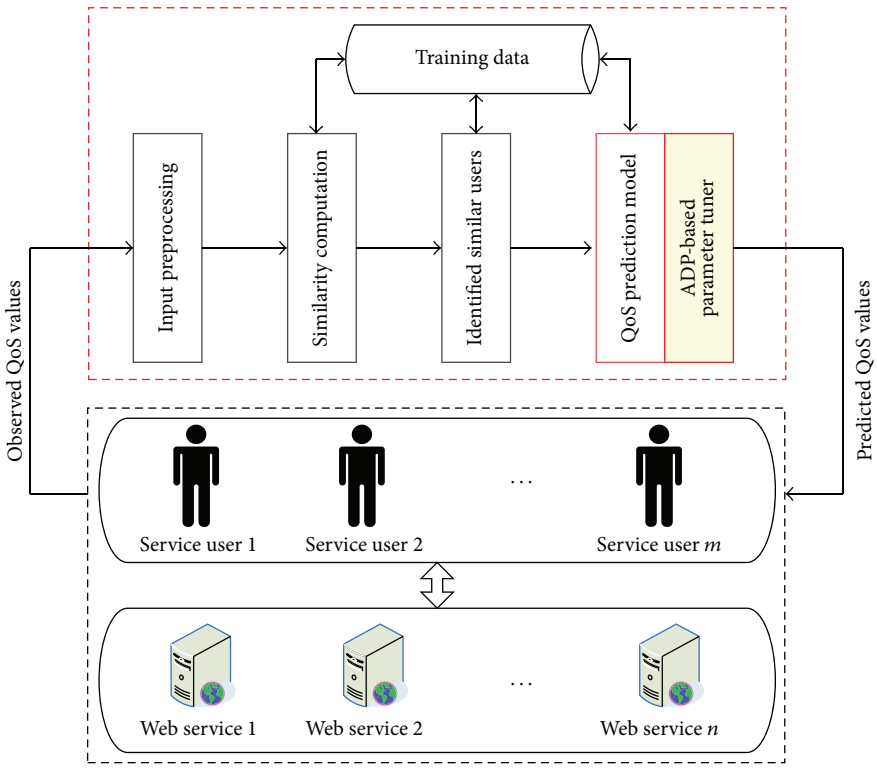

FIGURE 1: The proposed collaborative QoS prediction framework.

in this matrix is the value of a certain QoS property of a web observed by a service. In the last phase, an ADP-based parameter tuner is proposed to adjust the key parameters of the above collaborative filtering model to find out the near-optimal results. Details of this phase are presented in the next section.

There is an $m \times n$ user-item matrix R, where m and n are the number of service users and web services, respectively. Here, we use algorithm PCC to compute the similarities between different service users. Generally, because PCC considers the differences in the user value style, it can achieve high accuracy [13]. According to the computation approach of PCC, the similarity between two service users i and k can be expressed as

$$\mathrm{PCC}(i,k) = \frac{\sum_{j\in B}\left(R_{ij}-\overline{R}_i\right)\left(R_{kj}-\overline{R}_k\right)}{\left(\sum_{j\in B}\left(R_{ij}-\overline{R}_i\right)^2\right)^{1/2}\left(\sum_{j\in B}\left(R_{kj}-\overline{R}_k\right)^2\right)^{1/2}}, \tag{1}$$

where B is the subset of web services that are invoked by both service user i and user k. And R_{ij} is the element in the matrix R, which is the value of a certain client-side QoS property of web service j observed by service user i. In addition, $\overline{R}_i$ and $\overline{R}_k$ are the average QoS values of different web services observed by service users i and k, respectively. With those PCC values computed in (1), we can identify a set of top-K similar users. For a service user i, a set of similar users $T(i)$ is defined as [19]

$$T(i) = \{k \mid k \in \text{top-}K(i), \mathrm{PCC}(i,k) > 0,\ i \neq k\}, \tag{2}$$

where top-$K(i)$ represents a set of top-K similar users for the user i.

Then, a neighborhood-integrated matrix factorization approach is employed to make prediction in the user-item matrix [19]. For an $m \times n$ user-item matrix R, it is fitted by a matrix $X = U^T V$ using the matrix factorization algorithm, where $U \in \mathbb{R}^{l\times m}$, $V \in \mathbb{R}^{l\times n}$, and l represents the rank of the matrix X [16]. With this factorization of the matrix X, the web service QoS values from a user can be predicted by making a linear combination of the factor vectors in U. Actually, each column of V can be regarded as a linear predictor for a web service. Therefore, the prediction can be implemented by minimizing the following sum-of-squared-errors objective function E with quadratic regularization terms [19]:

$$\begin{aligned}\min E(U,V) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(R_{ij}-\left(\alpha U_i^T V_j + (1-\alpha)\sum_{k\in T(i)} S_{ik}U_k^T V_j\right)\right)\\ &\quad + \frac{\lambda_U}{2}\|U\|^2 + \frac{\lambda_V}{2}\|V\|^2,\end{aligned} \tag{3}$$

where $\alpha \in [0,1]$ is a weight coefficient designed to balance the usages of information from user and from the user's neighbors, λ_U and λ_V are two parameters used to avoid overfitting, and $\|\cdot\|$ represents the Frobenius norm. In addition, S_{ik} represents the normalized similarity between users i and k, where it is defined as

$$S_{ik} = \frac{\mathrm{PCC}(i,k)}{\sum_{k\in T(i)}\mathrm{PCC}(i,k)}. \tag{4}$$

By applying a gradient descent rule for U_i and V_j, the objective function $E(U, V)$ is minimized and a local minimum can be found.

2.2. Parameter Analysis. In function (3), the weight coefficient α bounded within $[0, 1]$ determines how much the objective function relies on the users themselves and their similar users, where the QoS prediction can be made only by using the information from the current user when $\alpha = 1$, and the missing QoS value is predicted purely by using the information from those similar users (i.e., the current user's neighbors) when $\alpha = 0$.

In addition, the top-K value determines the number of similar users employed in collaborative QoS prediction. Generally speaking, a too large top-K value may hurt the prediction accuracy, because it means that some dissimilar users may be involved in the prediction computation. Meanwhile, a too small top-K value may be often computationally debilitating.

Therefore, an appropriate α and a top-K value may help to improve the QoS prediction accuracy with low computational complexity. Here, we design an ADP-based parameter tuner which aims to find the satisfactory parameters α and top-K through online optimization.

3. ADP-Based Parameter Tuner

3.1. System Architecture. For this QoS prediction problem, those available collaborative filtering approaches (e.g., [19]) are developed in accordance with the designer's experience for parameter setting used in system model. And a set of static parameters lack adaptability. Thus, for an unpredictable network environment, the prediction performance is unsatisfactory only by using predefined static parameters.

The QoS prediction framework using our ADP-based parameter tuner is illustrated in Figure 2, which aims to address the above issue in the design of QoS prediction model. Here, the proposed ADP-based tuner is employed to update two key parameters in the QoS prediction model, that is, α and top-K value.

In Figure 2, the vector $X \triangleq (x_1, x_2) = (\alpha, k)$ from the QoS prediction model is taken as the input for the ADP-based tuner. Two regulatory signals $(\Delta\alpha, \Delta k)$ are generated through modulation transform for u which is the output of the ADP-based tuner, where k represents the the top-K value. And $(\Delta\alpha, \Delta k)$ are employed to update the parameters of the original QoS prediction model by

$$\begin{aligned} \alpha(t) &= \alpha(t-1) + \Delta\alpha(t), \\ k(t) &= k(t-1) + \Delta k(t). \end{aligned} \tag{5}$$

Moreover, the ADP-based parameter tuner in Figure 2 is implemented in accordance with actor-critic architecture of ADP [21]. It is illustrated in Figure 3, where the solid lines are signal flow and the dashed lines are the paths for NNs' parameter tuning. As mentioned above, there are two NNs in the ADP-based tuner. One is designed as an action network to take the system state $X(t)$ as input to output the control signal $u(t)$ at time t. And the other is designed as a critic network to take $X(t)$ and $u(t)$ as inputs to generate $J(t)$. The discount factor is $\gamma \in (0, 1)$. In the ADP-based tuner, $r(t)$ is a reinforcement signal which is provided from the QoS prediction model. Specifically, for the prediction problem, the reinforcement signal can be defined as a function of the measure of prediction results at time t (e.g., mean absolute error (MAE)). A small MAE will be encouraged. Actually, $r(t)$ is generated to indicate effectiveness of the control $u(t)$ and speed up the convergence.

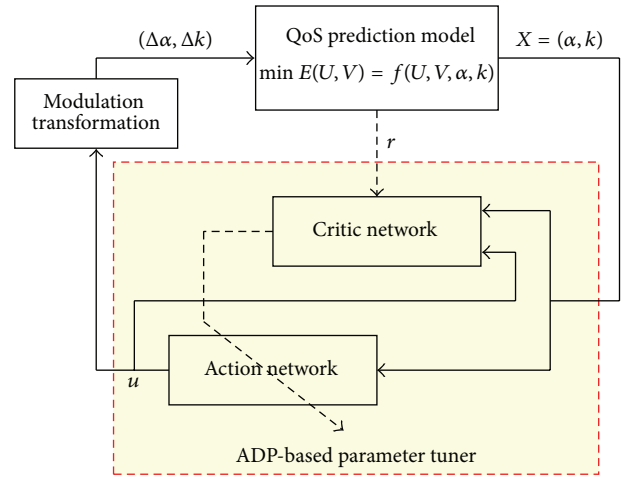

Figure 2: The QoS prediction framework with ADP-based parameter tuner.

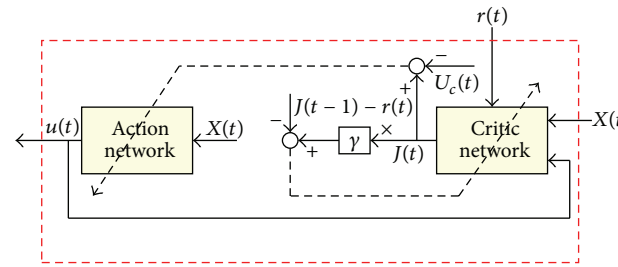

Figure 3: Schematic diagram for implementation of ADP-based parameter tuner.

Considering the reinforcement signal $r(t)$ as an instantaneous cost function, the cost function can be defined as

$$R(t) = \sum_{i=1}^{\infty} \gamma^{i-1} r(t+i). \tag{6}$$

The optimal parameter tuning problem is to find a control policy to minimize the cost function. So the optimal cost function can be defined as

$$R^*(t) = \min_{u(t)} \{ r(t+1) + \gamma R^*(t+1) \}. \tag{7}$$

Here, the output of the critic network (i.e., $J(t)$) is to approximate the cost function $R(t)$. When the critic network is well trained, $J(t)$ should satisfy the equation $J(t-1) = r(t) + \gamma J(t)$ or $\gamma J(t) - [J(t-1) - r(t)] = 0$ in accordance

with (7). And the output of the action network (i.e., $u(t)$) is to minimize the difference between the approximate function $J(t)$ and the desired objective, denoted by $U_c(t)$ which is set to 0 without loss of generality in this paper. Under this actor-critic architecture with convergence guarantees, a near-optimal control policy is achieved.

In addition, the regulatory signals are generated by using modulation transform for $u(t) \triangleq (u_1(t), u_2(t))$ as follows:

$$\begin{aligned} \Delta\alpha(t) &= M_1 u_1(t), \\ \Delta k(t) &= M_2 u_2(t), \end{aligned} \tag{8}$$

where $M_1, M_2 \in \mathbb{R}^+$ are modulation factors. Then, we provide the implementation details for our ADP-based parameter tuner.

3.2. The Action Network and the Critic Network. In our implementation, the feedforward NN with one hidden layer is used to design the action network and the critic network. The action network is illustrated in Figure 4, where $(x_1(t), x_2(t)) = (\alpha(t), k(t))$ are the inputs, $(u_1(t), u_2(t))$ are the outputs, $w^{(1)}_{a_{ij}}$ is the weight connecting the jth input node to the ith hidden node, and $w^{(2)}_{a_{ik}}$ is the weight connecting the ith hidden node to the kth output node. In addition, $v(t)$ is the input vector of the output nodes and $h(t)$ and $g(t)$ are the input and output vectors of the hidden nodes, respectively. Let $\varphi(\cdot)$ be a hyperbolic tangent threshold function used in the hidden layer, and it can be defined as

$$\varphi(x) = \frac{1 - e^{-x}}{1 + e^{-x}}. \tag{9}$$

In the action network, the approximate error and the objective function to be minimized are defined as

$$\begin{aligned} e_a(t) &= J(t) - U_c(t), \\ E_a(t) &= \frac{1}{2} e_a^2(t). \end{aligned} \tag{10}$$

The critic network is illustrated in Figure 5, where $w^{(1)}_{cij}$ is the weight connecting the jth input node to the ith hidden node and $w^{(2)}_{ci}$ is the weight connecting the ith hidden node to the output node. Moreover, $q(t)$ and $p(t)$ are the input and output vectors of the hidden nodes, respectively. Similarly, the error and the objective function to be minimized can be expressed as follows:

$$\begin{aligned} e_c(t) &= \gamma J(t) - [J(t-1) - r(t)], \\ E_c(t) &= \frac{1}{2} e_c^2(t). \end{aligned} \tag{11}$$

3.3. Weight Online Learning Rules. In our proposed online learning and optimization strategy, the feature of not requiring prior knowledge means that the action network and the critic network in ADP-based parameter tuner can both be randomly initialized toward their network weights in the initialization phase of collaborative QoS prediction.

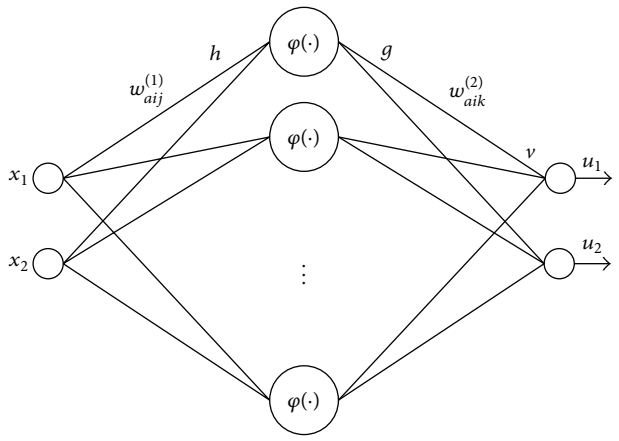

Figure 4: The action network.

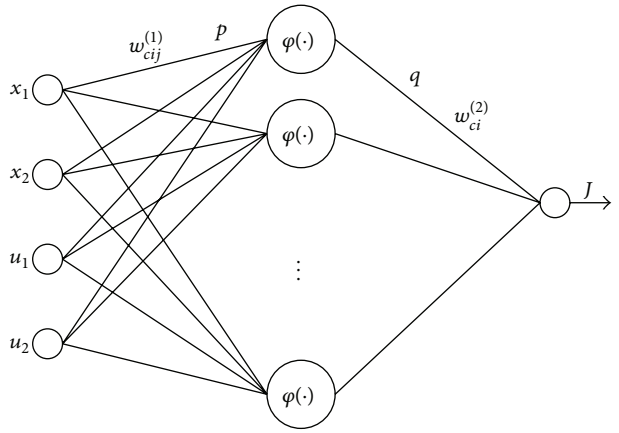

Figure 5: The critic network.

Once a system state is observed, an action will be subsequently produced through the computation for the combination of those weights in the action network. A good action will be encouraged, while a bad action will be punished.

The weight online learning rule for the action network is a gradient-based adaptation designed as

$$\begin{aligned} w_a(t+1) &= w_a(t) - l_a(t) \frac{\partial E_a(t)}{\partial w_a(t)}, \\ \frac{\partial E_a(t)}{\partial w_a(t)} &= \frac{\partial E_a(t)}{\partial J(t)} \frac{\partial J(t)}{\partial u(t)} \frac{\partial u(t)}{\partial w_a(t)}, \end{aligned} \tag{12}$$

where $w_a(t)$ is the weight vector in the action network and $l_a(t) > 0$ represents the learning rate of the action network at time t.

Here, the output $u(t)$ can be written as

$$\begin{aligned} u_k(t) &= \varphi(v_k(t)), \quad k = 1, 2, \\ v_k(t) &= \sum_{i=1}^{N_{ha}} w^{(2)}_{a_{ik}}(t)\, g_i(t), \quad k = 1, 2, \end{aligned}$$

$$g_i(t) = \varphi\left(h_i(t)\right), \quad i = 1, \ldots, N_{ha},$$
$$h_i(t) = \sum_{j=1}^{2} w_{a_{ij}}^{(1)}(t)\, x_j(t), \quad i = 1, \ldots, N_{ha}, \tag{13}$$

where N_{ha} is the number of hidden nodes in the action network.

The weight online learning rule for the critic network is also a gradient-based adaptation designed as

$$w_c(t+1) = w_c(t) - l_c(t)\frac{\partial E_c(t)}{\partial w_c(t)},$$
$$\frac{\partial E_c(t)}{\partial w_c(t)} = \frac{\partial E_c(t)}{\partial J(t)}\frac{\partial J(t)}{\partial w_c(t)}, \tag{14}$$

where $w_c(t)$ is the weight vector in the critic network and $l_c(t) > 0$ represents the learning rate of the critic network at time t.

Then, the output $J(t)$ is written as

$$J(t) = \sum_{i=1}^{N_{hc}} w_{ci}^{(2)} p_i(t),$$
$$p_i(t) = \varphi\left(q_i(t)\right), \quad i = 1, \ldots, N_{hc},$$
$$q_i(t) = \sum_{j=1}^{2} w_{cij}^{(1)} x_j(t) + \sum_{j=3}^{4} w_{cij}^{(1)}(t)\, u_{j-2}(t), \quad i = 1, \ldots, N_{hc}, \tag{15}$$

where N_{hc} is the number of hidden nodes in the critic network.

In the implementation of ADP-based tuner, it should be noted that the weight normalization is performed in both networks as follows:

$$w_c(t+1) = \frac{w_c(t) + \Delta w_c(t)}{\left\|w_c(t) + \Delta w_c(t)\right\|},$$
$$w_a(t+1) = \frac{w_a(t) + \Delta w_a(t)}{\left\|w_a(t) + \Delta w_a(t)\right\|}. \tag{16}$$

Finally, the ADP-based parameter tuner algorithm is summarized in Algorithm 1. Here, N_c and N_a are the internal cycles of the critic network and the action network, respectively. In addition, T_c and T_a are the internal training error thresholds for the critic network and the action network, respectively.

4. Experiment Results and Discussions

4.1. Data Set Description. To test the performance of our approach in real-world, we use a common data set collected from 339 service users (in 30 countries) on 5,825 web services (in 73 countries) [25, 26]. This experiment focuses on the investigation for the response time of different web services and service users. Response time is one of the representative QoS values, which is defined as the time duration between sending a request and receiving a corresponding response for a service user.

In this experiment, there is a total of 1,974,675 real-world web service invocation data. After processing those web service invocation data, there is a 339 × 5,825 user-item matrix, where each element in this matrix represents the response time value observed by a user on a web service [19].

4.2. Algorithm Settings. The parameters used in the experiments are summarized in Table 1, where the notations are defined as follows:

$l_a(0)$: initial learning rate of the action network,

$l_c(0)$: initial learning rate of the critic network,

$l_a(t)$: learning rate of the action network at time t which is decreased by 0.05 every 5 time steps until it reaches $l_a(f)$ and stays thereafter,

$l_c(t)$: learning rate of the critic network at time t which is decreased by 0.05 every 5 time steps until it reaches $l_c(f)$ and stays thereafter,

N_a: internal cycle of the action network,

N_c: internal cycle of the critic network,

T_a: internal training error threshold for the action network,

T_c: internal training error threshold for the critic network.

Here, the weights in the action and the critic networks are trained using their internal cycles or their internal training error threshold.

4.3. Experiment Results for Case 1

4.3.1. Statement of Evaluation. In this experiment, the mean absolute error (MAE) and root-mean-squared error (RMSE) metrics are used to measure the prediction quality of our approach in the comparative study. MAE is defined by

$$\text{MAE} = \frac{1}{N}\sum_{i,j}\left|R_{ij} - \widehat{R}_{ij}\right|, \tag{17}$$

where $\widehat{R}_{ij}$ represents the predicted QoS value of web service j observed by service user i and N is the number of predicted values. And RMSE is defined by

$$\text{RMSE} = \left(\frac{1}{N}\sum_{i,j}\left(R_{ij} - \widehat{R}_{ij}\right)^2\right)^{1/2}. \tag{18}$$

Here, the MAE is to measure how close predicted QoS values are to the observed QoS values. Compared to the MAE, the RMSE provides an effective way to severely punish large errors.

4.3.2. Performance Comparison. We adopt the proposed optimal online parameter tuning methodology based on ADP to achieve the QoS prediction model parameters identification for α and k. The values of α and k obtained by using our approach are similar to those achieved by trial and error with method of exhaustion in [19]. Figures 6 and 7 show the

(1) initialize the weights of the critic network and the action network arbitrarily;
(2) **for** each trail (from 1 to maximum trail) **do**
(3) initialize $X(t) = (\alpha(t), k(t))$;
(4) **for** each step t in a trail **do**
(5) send $X(t)$ to the action network and get the output $u(t) = (u_1(t), u_2(t))$;
(6) send $X(t)$ and $u(t)$ to the critic network, then get the output $J(t)$;
(7) calculate $\Delta\alpha(t) \leftarrow M_1 u_1(t)$ and $\Delta k(t) \leftarrow M_2 u_2(t)$;
(8) take $X(t+1) = (\alpha(t+1), k(t+1))$, where $\alpha(t+1) \leftarrow \alpha(t) + \Delta\alpha(t)$ and $k(t+1) \leftarrow k(t) + \Delta k(t)$;
(9) check the boundedness for $\alpha(t+1) \in (0, 1)$ and $k(t+1) \in (0, \text{Max\_User})$;
(10) send $\alpha(t+1)$ and $k(t+1)$ to the QoS prediction model;
(11) get reinforcement signal $r(t+1) = \text{MAE}$;
(12) **repeat** to update weight of the critic network $w_c(t)$
(13) **until** $E_c(t) < T_c$ or the maximum iteration number N_c is met;
(14) **repeat** to update weight of the action network $w_a(t)$
(15) **until** $E_a(t) < T_a$ or the maximum iteration number N_a is met;
(16) $X(t) \leftarrow X(t+1)$ and $J(t-1) \leftarrow J(t)$;
(17) **end for**
(18) **end for**
(19) return prediction result and MAE.

Algorithm 1: ADP-based parameter tuner.

Table 1: Summary of parameters used in ADP-based tuner for QoS prediction.

Parameter	$l_a(0)$	$l_c(0)$	$l_a(f)$	$l_c(f)$	N_{ha}	N_a	N_c	T_a	T_c	N_{hc}
Value	0.3	0.3	0.005	0.005	6	100	50	0.005	0.05	6

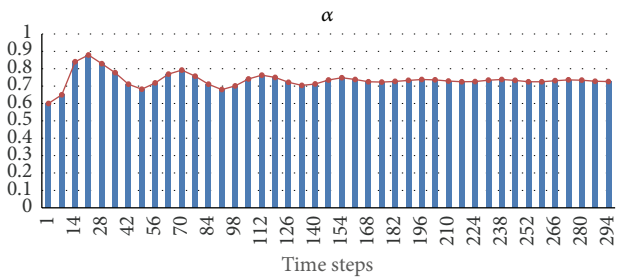

Figure 6: A typical trajectory of α during a successful learning trial for the ADP-based parameter tuner.

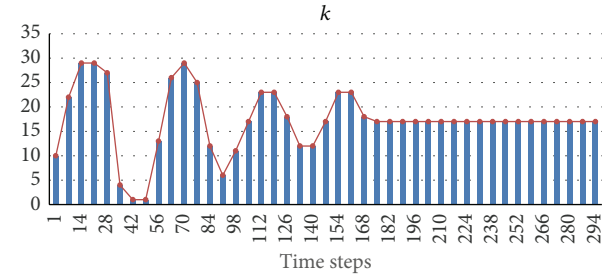

Figure 7: A typical trajectory of k during a successful learning trial for the ADP-based parameter tuner.

trajectories of α and k under the ADP-based parameter tuner for a successful learning trial, respectively. It can be seen that α and k are gradually tuned through the use of our tuner. After 170 time steps, α and k almost reach steady states, which indicates that the learning process of ADP has converged and near-optimal α and k are obtained.

Moreover, to evaluate the prediction accuracy, we compare our approach with the following approaches:

(i) UPCC (user-based collaborative filtering method using PCC): it only employs similar users for the QoS value prediction [13, 14];

(ii) IPCC (item-based collaborative filtering method using PCC): it only employs similar web services for the QoS value prediction [15];

(iii) UIPCC: it employs both similar users and similar web services for the QoS value prediction [27];

(iv) NMF (nonnegative matrix factorization): it is also a collaborative filtering method based on matrix factorization; unlike other matrix factorization methods, it enforces the constraint that the factorized factors must be nonnegative [28];

(v) NIMF (neighborhood-integrated matrix factorization): it fuses the neighborhood-based and model-based collaborative filtering approaches for the QoS value prediction [19].

The experimental results are shown in Table 2. From Table 2, we can observe that our method can almost obtain better prediction accuracy (with smaller MAE and RMSE values) than UPCC, IPCC, UIPCC, and NMF methods for response time. But the performance of our method is slightly poorer than the NIMF method, which is due to the fact that

Table 2: Performance comparison.

Methods	Matrix density = 10%		Matrix density = 20%	
	MAE	RMSE	MAE	RMSE
UPCC	0.5655	1.3326	0.5516	1.3114
IPCC	0.7596	1.6133	0.7624	1.6257
UIPCC	0.5654	1.3309	0.5053	1.2486
NMF	0.6754	1.5354	0.6771	1.5241
NIMF	0.4854	1.2745	0.4357	1.1678
Our method	0.5632	1.3199	0.5314	1.2533

the NIMF method can get the globally optimal solutions of α and k through global search in [19], while our method obtains the near-optimal solutions through online parameter tuning methodology based on ADP without any designers' a priori knowledge. However, the NIMF method needs to conduct an exhaustive search for the optimal parameters mindlessly in a large parameters space by trial and error; our method can automatically tune the parameters through online learning and optimization. In this regard, our method may reduce the computing time and improve the computational efficiency for prediction implementation. Moreover, when the matrix density is 20%, the MAE and RMSE values are smaller than those values when the matrix density is 10%, since denser matrix can provide more information for predicting the missing values.

4.4. Experiment Results for Case 2. To be more complex, we design a changeful network environment by adding some random disturbance to those 1, 974, 675 web service invocation results of the data set used in this experiment. If some similar users' network environment changes, those similar users may not be worthy of trust. It is obvious that the initial parameters of prediction model are unreasonable now. With the online optimization of ADP-based tuner, our method will automatically adjust parameters of prediction model to conform with the changes in the new network environment. In this case, we make QoS predictions for a service user under the changed network environment. Here, we compare our ADP-based prediction method with the NIMF method [19].

In this case, we consider a kind of disturbance for similar users. In the experiment, we artificially add a series of unrelated users to replace part of similar users with the purpose of reducing the PCC value among similar users. As our ADP-based method perceives the changes happening in similar users, it will adjust the number of similar users employed in the method by reducing the top-K value (i.e., the value k in our approach) and reduce the usage of information from similar users by increasing the value α which appeared in (3).

In Figure 8, we check the response time of the first 500 missing web services observed by one service user under the changed network environment. In this figure, the red line with spots represents the actual QoS value in every missing entry, and the blue line with stars represents the prediction data obtain by using the NIMF method with the fixed parameters of prediction model. In addition, the green line with rhombus represents the predictions results obtained by using our ADP-based online method with optimized parameters of prediction model. We can find a big error between the actual data and the prediction data using NIMF method in a changed network environment. Meanwhile, the prediction method with the ADP-based parameter tuner can achieve the satisfactory QoS prediction performance.

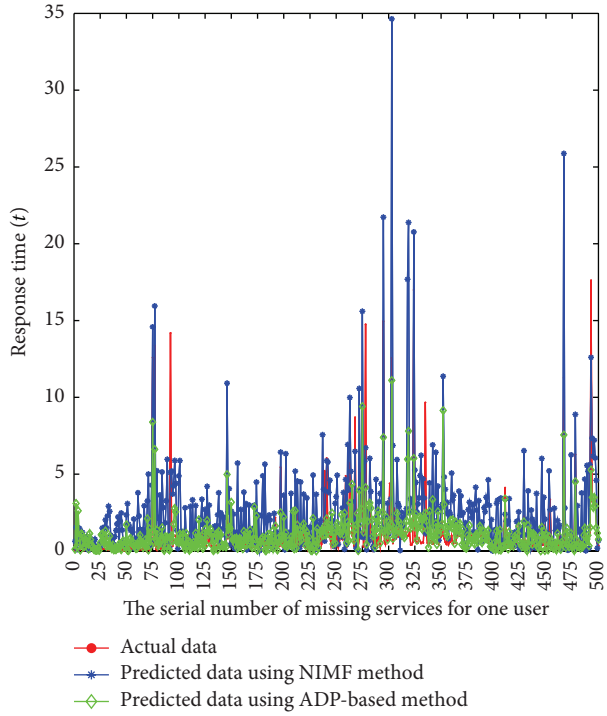

Figure 8: The comparison of prediction results.

5. Conclusion

In this paper, we propose an ADP-based parameter tuner for QoS prediction model using collaborative filtering algorithm. Due to its online adaptability and approximation capabilities through successive iterations, this ADP-based parameter tuner shows satisfactory performance in a large-scale experiment studied in this paper. Moreover, our approach can easily adapt to the changes of network environment without requiring prior knowledge or identification of the prediction model. Then, our method may not be computationally intensive for the scale of web service QoS data set in a complex and changeful network environment. Furthermore, with the help of the flexible structure and learning algorithm of ADP, the proposed method can be extended to other types of QoS prediction models and achieve near-optimal performance. Our experiment studies validate the effectiveness of the proposed ADP-based parameter tuner.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was jointly supported by the National Natural Science Foundation of China under Grants 61174103 and 61272357, the National Key Technologies R&D Program of China under Grant 2015BAK38B01, and the Aerospace Science Foundation of China under Grant 2014ZA74001.

References

[1] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Transactions on Software Engineering*, vol. 33, no. 6, pp. 369–384, 2007.

[2] L. J. Zhang, J. Zhang, and H. Cai, *Services Computing*, Springer, Berlin, Germany, 2007.

[3] A. Jula, E. Sundararajan, and Z. Othman, "Cloud computing service composition: a systematic literature review," *Expert Systems with Applications*, vol. 41, no. 8, pp. 3809–3824, 2014.

[4] W. Zhang, K. M. Hansen, and M. Ingstrup, "A hybrid approach to self-management in a pervasive service middleware," *Knowledge-Based Systems*, vol. 67, pp. 143–161, 2014.

[5] G. Shua, X. Chub, and D. Chen, "Supporting QoS-based discovery for visualization web services," *International Journal of Distributed Sensor Networks*, vol. 5, no. 1, p. 19, 2009.

[6] S. Dong and W. Dong, "A QoS driven web service composition method based on ESGA (elitist selection genetic algorithm) with an improved initial population selection strategy," *International Journal of Distributed Sensor Networks*, vol. 5, no. 1, p. 54, 2009.

[7] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for Web services selection with end-to-end QoS constraints," *ACM Transactions on the Web*, vol. 1, no. 1, article 6, 2007.

[8] V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. L. Presti, and R. Mirandola, "MOSES: a framework for QoS driven runtime adaptation of service-oriented systems," *IEEE Transactions on Software Engineering*, vol. 38, no. 5, pp. 1138–1159, 2012.

[9] H. Y. Zheng, W. L. Zhao, J. Yang, and A. Bouguettaya, "QoS analysis for web service compositions with complex structures," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 373–386, 2013.

[10] Z. Zheng, Y. Zhang, and M. R. Lyu, "Investigating QoS of real-world web services," *IEEE Transactions on Services Computing*, vol. 7, no. 1, pp. 32–39, 2014.

[11] X. C. Zhao, Z. C. Wen, and X. M. Li, "QoS-aware web service selection with negative selection algorithm," *Knowledge and Information Systems*, vol. 40, no. 2, pp. 349–373, 2014.

[12] M. Mehdi, N. Bouguila, and J. Bentahar, "Probabilistic approach for QoS-aware recommender system for trustworthy web service selection," *Applied Intelligence*, vol. 41, no. 2, pp. 503–524, 2014.

[13] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pp. 43–52, Madison, Wis, USA, July 1998.

[14] 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.

[15] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: an open architecture for collaborative filteringof netnews," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pp. 175–186, Chapel Hill, NC, USA, October 1994.

[16] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proceedings of Conference on Advances in Neural Information Processing Systems*, pp. 1257–1264, Vancouver, Canda, December 2007.

[17] Q. Wu, X. Yang, and Q. Zhou, "Study on selection strategy of web service based on fusion of subjective and objective evaluation for QoS," *Journal of Information & Computational Science*, vol. 10, no. 13, pp. 4213–4223, 2013.

[18] X. Luo and X. Chang, "A novel data fusion scheme using grey model and extreme learning machine in wireless sensor networks," *International Journal of Control, Automation, and Systems*, vol. 13, no. 5, 2015.

[19] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative web service Qos prediction via neighborhood integrated matrix factorization," *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 289–299, 2013.

[20] J. Si, A. Barto, W. Powell, and D. Wunsch, *Handbook of Learning and Approximate Dynamic Programming*, John Wiley & Sons, Hoboken, NJ, USA, 2004.

[21] J. Si and Y.-T. Wang, "On-line learning control by association and reinforcement," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 264–276, 2001.

[22] F. L. Lewis and D. R. Liu, *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, Wiley-IEEE, Hoboken, NJ, USA, 2012.

[23] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 621–634, 2014.

[24] P. J. Werbos, "Approximate dynamic programming for real-time control and neural modeling," in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, vol. 15, pp. 493–525, Van Nostrand-Reinhold, New York, NY, USA, 1992.

[25] Z. Zheng, Y. Zhang, and M. R. Lyu, "Distributed QoS evaluation for real-world Web services," in *Proceedings of the IEEE 8th International Conference on Web Services (ICWS '10)*, pp. 83–90, Miami, Fla, USA, July 2010.

[26] http://www.wsdream.net/dataset.html.

[27] 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.

[28] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.