A Time-Aware and Data Sparsity Tolerant Approach for Web Service Recommendation

Yan Hu*, Qimin Peng, Xiaohui Hu
Science and Technology on Integrated Information System Laboratory
Institute of Software, Chinese Academy of Sciences
Beijing, China
Email: {huyan, qimin, hxh}@iscas.ac.cn

***Abstract*—With the incessant growth of Web services on the Internet, designing effective Web service recommendation technologies based on Quality of Service (QoS) is becoming more and more important. Neighborhood-based Collaborative Filtering has been widely used for Web service recommendation, in which similarity measurement and QoS prediction are two key steps. However, traditional similarity models and QoS prediction methods rarely consider the influence of time information, which is an important factor affecting the QoS of Web services. Furthermore, traditional similarity models fail to capture the actual relationships between users or services due to data sparsity. These shortcomings seriously devalue the performance of neighborhood-based Collaborative Filtering. In order to make high-quality Web service recommendation, we propose a novel time-aware approach, which integrates time information into both the similarity measurement and the final QoS prediction. Additionally, in order to alleviate the data sparsity problem, a hybrid personalized random walk algorithm is employed to infer more indirect user similarities and service similarities. Finally, we conduct series of experiments to validate the effectiveness of our approaches.**

***Keywords*-Web service recommendation; time information; data sparsity; hybrid personalized random walk**

I. Introduction

In recent years, Service-Oriented Computing (SOC) [1] has attracted a lot of attention as a promising computing paradigm for software engineering and distributed computing [2]. In this paradigm, Web services are basic constructs. According to recent statics [3], there are 28,606 Web services available on the Internet as of 07/01/2013, provided by 7,739 providers. The Increasing number of Web services calls for effective Web service selection and recommendation technologies. And with the proliferation of multiple Web services with identical or similar functionalities, we need additional information to differentiate them.

Quality of Service (QoS) describes the non-functional characteristics of Web services, including response time, throughput, availability, etc. QoS has been employed as an important factor to differentiate Web services which provide analogous functionalities [4]. QoS-based Web service recommendation helps service users to select Web services which not only meet their functional requirements but also have optimal QoS performance. However, service providers rarely deliver the QoS as declared, due to:

- Non-functional performance of Web services is highly correlated with invocation time, since the service status (e.g., number of clients, workload, etc.) and the network condition (e.g., bandwidth, etc.) change over time.
- Service users are typically geographically distributed. The QoS performance of Web services observed by users is highly influenced by the network connections between users and Web services. Different users may observe totally different QoS performance even if they invoke a same Web service.

Based on the above analyses, making time-aware personalized QoS prediction is important for high-quality Web service recommendation.

In recent literature, neighborhood-based Collaborative Filtering (CF) has become the most popular technology for personalized QoS prediction [5–10], mainly including user-based [5, 6], item-based [7, 8] and hybrid [9, 10] approaches. The user-based method utilizes the historical QoS experience from similar users to make QoS prediction, while the item-based method uses the historical QoS information from similar services to predict missing QoS values. The hybrid approach is the combination of the previous two methods, which can achieve superior performance. Furthermore, to improve the basic CF methods, some researchers propose to take context information into consideration for more accurate Web service recommendation. The most discussed context information is location [11, 12]. However, to the best of our knowledge, none of the existing Web service recommendation approaches relying on neighborhood-based CF consider the temporal dynamics of QoS, which is also an important context factor. This results in biased QoS prediction, which greatly limits the performance of Web service recommendation.

Apart from the lack of consideration of time information, another drawback of neighborhood-based CF is that it cannot handle data sparsity well. The number of users and services on the Internet is very large. Even very active users may invoke only a few Web services and even very popular Web services may be invoked by only a few users. This problem, commonly referred to as the $sparsity\ problem$ [13]. In literature, the random walk mechanism has been widely discussed to alleviate data sparsity [14–17]. Its basic idea is to construct a graph by treating items as nodes and the relationships between items as edges, and assign each candidate item a

978-1-4799-5054-6/14 $31.00 © 2014 IEEE
DOI 10.1109/ICWS.2014.18

stationary visiting probability as its ranking score. Items with higher ranking scores are more likely to be recommended to users. However, Web service recommendation is a multi-criteria decision-making procedure [18]. We should predict the missing value on each QoS property for a candidate Web service and merge the values on different properties into a comprehensive ranking score, rather than assign each Web service a single-value ranking score simply. So, motivated by the existing random walk algorithms and the features of Web services, we propose a hybrid personalized random walk algorithm to handle data sparsity for Web service recommendation. It performs random walk on both the user graph and the service graph to identify more similar neighbors for the target user and each candidate Web service, respectively. Thus, we can extract enough historical information from these similar neighbors to make more accurate QoS prediction.

The key contributions of our work are two-fold:

- Time information is integrated into the similarity measurement and the QoS prediction of the traditional neighborhood-based CF, for higher-quality Web service recommendation.
- A hybrid personalized random walk algorithm is employed to handle data sparsity, by performing personalized random walk on both the user graph and the service graph, to discover more indirect similar users and services. Then, the user and service random walk results are combined together for final QoS prediction.

The remainder of this paper is organized as follows. Section II presents some related work. In Section III, we propose a time-aware similarity model. In Section IV, a hybrid personalized random walk algorithm is employed to handle data sparsity. Section V discusses how to make time-aware QoS prediction. Section VI presents the experimental results. Conclusions are finally drawn in Section VII.

II. Related Work

A Web service recommender system selects a Web service with optimal QoS performance from a group of service candidates and recommends it to a service user. Since the QoS performance of Web services is generally not predetermined, neighborhood-based CF algorithms, which mainly include user-based [5, 6], item-based [7, 8] and hybrid [9, 10] approaches, are employed to make QoS prediction. In addition, to achieve better prediction performance, other researchers incorporate context information into the basic CF methods. The most widely discussed context information is location [11, 12]. They hold the opinion that the geographically close users or services may have similar QoS experience. Thus, more historical QoS values from geographically close neighbors can be obtained for more accurate QoS prediction. However, to the best of our knowledge, none of the existing neighborhood-based CF methods for Web service recommendation take into consideration the service invocation time information, which is another important context factor affecting QoS. This greatly limits the QoS prediction accuracy, since the QoS performance of Web services is highly related to invocation time due to some time-varying factors (e.g., service status, network condition, etc.).

Another drawback of neighborhood-based CF is that it is vulnerable to data sparsity. The indirect similarity inference approach [13] focuses on inferring indirect relationship between any two given users to alleviate data sparsity. However, this method cannot automatically recognize indirect similar neighbors for a given user. Other researchers employ random walk algorithms for automatic discovery of related objects. Yildirim et al. [14] perform random walk on an item graph (extracted by treating items as nodes and item similarities as edges) to find more related items, and assign each item a stationary visiting probability as its ranking score. This approach can alleviate data sparsity to some extent, but does not take user information into consideration. In [15–17, 19], random walk is performed on a hybrid graph, which is extracted by treating both users and items as nodes and the user-item interactions as edges. Similarly, items with higher stationary visiting probabilities are more likely to be recommended to target users. These methods based on random walk are applicable to one-dimensional rating systems. However, a Web service recommender system is a multi-criteria decision-making system, in which users should evaluate the performance of a Web service based on more than one QoS property, rather than simply assign each candidate service a single-value ranking score during the random walk.

In this paper, we build a user graph and a service graph, whose edges are weighted by user similarities and service similarities, respectively. The random walk algorithm is performed on both of them to discover more indirect similar neighbors for the target user and each candidate Web service. Plentiful historical data extracted from these similar neighbors can yield higher QoS prediction accuracy. Finally, the user-based and service-based prediction results are effectively merged into a comprehensive prediction result.

III. A Time-Aware Similarity Measurement

In this section, we first formalize the QoS prediction problem in Section III-A. Then, a time-aware similarity model is described in Section III-B.

A. Problem Definition

Suppose that a Web service recommender system contains a set of users $U = \{u_1, u_2, \ldots, u_M\}$ and a set of Web services $S = \{s_1, s_2, \ldots, s_N\}$. The user-service relationships are denoted by a $M \times N$ matrix called the user-service matrix $\boldsymbol{Q}$, as shown in Figure 1. Each entry of $\boldsymbol{Q}$ is a 2-tuple $(\boldsymbol{q}_{ij}, t_{ij})(1 \leq i \leq M, 1 \leq j \leq N)$, where $\boldsymbol{q}_{ij}$ is the vector of QoS values observed by user u_i on service s_j, and t_{ij} is the timestamp when u_i invoked s_j. If total k QoS properties are involved in the system, each $\boldsymbol{q}_{ij}$ is a k-dimensional vector, with each dimension denoting one QoS property. If u_i never invoked s_j, we have $\boldsymbol{q}_{ij} = \mathbf{0}$ (a missing QoS vector). For simplicity, if a user invoked a same service for more than once, only his latest QoS observation and timestamp on this service are recorded in $\boldsymbol{Q}$. Suppose that u_i is a target user, and (s)he requests a Web service recommendation by providing some functional requirements. Then the system should recommend a candidate Web service which not only provides the required functionalities but also has optimal QoS performance to the target user u_i. Consequently, the missing QoS vector $\boldsymbol{q}_{ij}$ should be predicted for each pair (u_i, s_j) (s_j is one of the candidate Web services) at the current time $t_{current}$ (the time when the recommendation is requested), based on the historical QoS information contained in $\boldsymbol{Q}$.

$$\begin{array}{c|cccc|}
 & s_1 & s_2 & \cdots & s_N \\
\hline
u_1 & (\boldsymbol{q}_{11}, t_{11}) & (\boldsymbol{q}_{12}, t_{12}) & \cdots & (\boldsymbol{q}_{1N}, t_{1N}) \\
u_2 & (\boldsymbol{q}_{21}, t_{21}) & (\boldsymbol{q}_{22}, t_{22}) & \cdots & (\boldsymbol{q}_{2N}, t_{2N}) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
u_M & (\boldsymbol{q}_{M1}, t_{M1}) & (\boldsymbol{q}_{M2}, t_{M2}) & \cdots & (\boldsymbol{q}_{MN}, t_{MN}) \\
\hline
\end{array}$$

Fig. 1. Time-Aware User-Service Matrix

B. *A Time-Aware Similarity Model*

Similarity measurement is a key step of neighborhood-based CF [9]. Almost all similarity methods of neighborhood-based CF are listed in [20] (e.g., Pearson Correlation Coefficient (PCC), cosine measure, adjusted cosine measure, constrained PCC, etc.). However, all these similarity models neglect the influence of service invocation time, which is an influential context factor because the QoS performance of Web services is highly related to service invocation time due to the time-varying service status and network environment. Generally speaking, a longer timespan indicates a higher probability that a QoS value deviates from its original value.

We now elaborate on the time-aware similarity measurement for users, and that for services is analogous. First, we assume that a user can finish a service invocation in a very short timespan, which can be regarded as an instant. This assumption is reasonable, since compared with the entire life cycle of the recommender system, the duration of a service invocation is short enough to be ignored. There are two intuitive principles behind the user similarity measurement:

1) More temporally close QoS experience from two users on a same service contributes more to the user similarity measurement.

As shown in Figure 2, suppose that the Web service recommender system starts at t_{start}. A user u_i invoked a Web service s_k at t_{ik}, and another user u_j invoked the same service s_k at t_{jk}. The timespan between t_{ik} and t_{jk} is denoted by $\triangle t_1$. If $\triangle t_1$ is long, even though u_i and u_j have very similar QoS experience on s_k, it does not really mean high similarity between u_i and u_j, since u_i's QoS experience on s_k may change violently over $\triangle t_1$. Therefore, a shorter $\triangle t_1$ (more temporally close) generally indicates a greater contribution of s_k to the similarity measurement between u_i and u_j. Thus, the contribution of s_k can be approximately weighted by a exponential decay function of $\triangle t_1$, which is defined as:

$$f_1(t_{ik}, t_{jk}) = e^{-\alpha |t_{ik} - t_{jk}|}, \tag{1}$$

where $\alpha \geq 0$ is a non-negative decay constant, with a larger α making the value of f_1 (ranging from 1 to 0) vanish more rapidly with the increase of the timespan $|t_{ik} - t_{jk}|$.

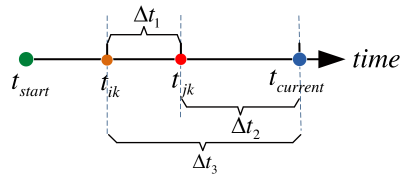

Fig. 2. Time Factors Affecting Similarity Measurement

2) More recent QoS experience from two users on a same service contributes more to the user similarity measurement.

The similarity between two users at the current time generally depends more on their recent QoS experience. Namely, if two users invoked a same service a long time ago, their QoS experience on this service should be less considered while computing their similarity at the current time. As illustrated in Figure 2, $\triangle t_2$ is the timespan between u_j's invocation on service s_k and the current moment, and $\triangle t_3$ is the timespan between u_i's invocation on s_k and the current moment. We utilize $\triangle t_4 = (\triangle t_2 + \triangle t_3)/2$ (the average of $\triangle t_2$ and $\triangle t_3$) to denote the second time factor. A shorter $\triangle t_4$ (more recent) generally indicates a greater contribution of s_k to the user similarity measurement. Thus, we consider that the contribution of s_k also decays exponentially with the increase of $\triangle t_4$. This exponential decay function is defined as:

$$f_2(t_{ik}, t_{jk}) = e^{-\beta |t_{current} - (t_{ik} + t_{jk})/2|}, \tag{2}$$

where $\beta \geq 0$ is a non-negative decay constant, with a larger β making f_2 (ranging from 1 to 0) vanish much more rapidly with the increase of the timespan $|t_{current} - (t_{ik} + t_{jk})/2|$.

Here, we take PCC as an example to describe the time-aware similarity model. Time information can be integrated into other similarity models listed in [20] in a similar way. Based on the definitions of f_1 and f_2, the time-aware PCC-based similarity measurement between u_i and u_j, can be defined as:

$$w_{ij}^u = \frac{\sum\limits_{s_k \in S} (\boldsymbol{q}_{ik} - \bar{\boldsymbol{q}}_i)(\boldsymbol{q}_{jk} - \bar{\boldsymbol{q}}_j) f_1(t_{ik}, t_{jk}) f_2(t_{ik}, t_{jk})}{\sqrt{\sum\limits_{s_k \in S} (\boldsymbol{q}_{ik} - \bar{\boldsymbol{q}}_i)^2} \sqrt{\sum\limits_{s_k \in S} (\boldsymbol{q}_{jk} - \bar{\boldsymbol{q}}_j)^2}}, \tag{3}$$

where $S = S_{u_i} \cap S_{u_j}$ is the subset of Web services commonly invoked by u_i and u_j, and $\bar{\boldsymbol{q}}_i$ and $\bar{\boldsymbol{q}}_j$ are the average QoS vectors observed by u_i and u_j, respectively. The above equation often overestimates the similarities between two users who are not similar actually but happen to have similar QoS observations on a small number of co-invoked Web services. To deal with this problem, we utilize a similarity weight proposed in [9, 10] to reduce the impact of a small number of co-invoked Web services:

$$w_{ij}^{\prime u} = \frac{2|S_{u_i} \cap S_{u_j}|}{|S_{u_i}| + |S_{u_j}|} \cdot w_{ij}^u, \tag{4}$$

where $|S_{u_i}|$ and $|S_{u_j}|$ are the numbers of Web services invoked by u_i and u_j, and $|S_{u_i} \cap S_{u_j}|$ is the number of their co-invoked Web services. $w_{ij}^{\prime u}$ ranges from -1 to 1, with a larger $w_{ij}^{\prime u}$ indicating a higher similarity between u_i and u_j.

Likewise, the time-aware similarity between two Web services s_k and s_l is defined by Eq. (5) and (6):

$$w_{kl}^{s} = \frac{\sum\limits_{u_i \in U} (\boldsymbol{q}_{ik} - \bar{\boldsymbol{q}}_k)(\boldsymbol{q}_{il} - \bar{\boldsymbol{q}}_l) f_1(t_{ik}, t_{il}) f_2(t_{ik}, t_{il})}{\sqrt{\sum\limits_{u_i \in U} (\boldsymbol{q}_{ik} - \bar{\boldsymbol{q}}_k)^2} \sqrt{\sum\limits_{u_i \in U} (\boldsymbol{q}_{il} - \bar{\boldsymbol{q}}_l)^2}}, \quad (5)$$

$$w_{kl}^{\prime s} = \frac{2|U_{s_k} \cap U_{s_l}|}{|U_{s_k}| + |U_{s_l}|} \cdot w_{kl}^{s}, \quad (6)$$

where U_{s_k} and U_{s_l} are the subsets of users who invoked s_k and s_l, respectively, and $U = U_{s_k} \cap U_{s_l}$ denotes the group of users who invoked both s_k and s_l. Additionally, $\bar{\boldsymbol{q}}_k$ and $\bar{\boldsymbol{q}}_l$ are the average QoS vectors of s_k and s_l observed by their respective users. $w_{kl}^{\prime s}$ also ranges from -1 to 1, with a larger value indicating that s_k and s_l are more similar. If α and β are both set to 0, we have $f_1 = f_2 = 1$. Namely, in this particular case, the time-aware PCC degenerates into the traditional PCC described in [9, 10]. Therefore, the latter is just a special case of the former. The entire procedure of the time-aware user similarity calculation is shown in Algorithm 1.

In this algorithm, $\boldsymbol{sumq}_i$ and $\boldsymbol{sumq}_j$ are the sum QoS vectors observed by u_i and u_j, respectively. Line 1 performances some initialization. Line 2 to 10 calculate the users' average QoS vectors. Then, the similarity weight and the denominator of time-aware PCC (line 11 and 12) are computed. Afterwards, the contribution of each co-invoked Web service (revised by f_1 and f_2) is added to a variable sum (line 13 to 17). Finally, the similarity between u_i and u_j is calculated and returned (line 18 and 19). The service similarity calculation is similar to Algorithm 1. Due to limited space, no more tautology here.

Algorithm 1 Time-Aware User Similarity Computation

```
Input:
    user $u_i$; user $u_j$; time-aware user-service matrix $\boldsymbol{Q}$; current time $t_{current}$; time decay constants $\alpha$, $\beta$
Output:
    the time-aware similarity between $u_i$ and $u_j$: $w_{ij}^{\prime u}$
1:  $S_{u_i} \leftarrow \emptyset$; $S_{u_j} \leftarrow \emptyset$; $\boldsymbol{sumq}_i \leftarrow \boldsymbol{0}$; $\boldsymbol{sumq}_j \leftarrow \boldsymbol{0}$;
2:  for $k \leftarrow 1$ to $m$ do
3:      if $\boldsymbol{q}_{ik} \neq \boldsymbol{0}$ then
4:          $S_{u_i} \leftarrow S_{u_i} \cup \{s_k\}$; $\boldsymbol{sumq}_i \leftarrow \boldsymbol{sumq}_i + \boldsymbol{q}_{ik}$;
5:      end if
6:      if $\boldsymbol{q}_{jk} \neq \boldsymbol{0}$ then
7:          $S_{u_j} \leftarrow S_{u_j} \cup \{s_k\}$; $\boldsymbol{sumq}_j \leftarrow \boldsymbol{sumq}_j + \boldsymbol{q}_{jk}$;
8:      end if
9:  end for
10: $\bar{\boldsymbol{q}}_i \leftarrow \boldsymbol{sumq}_i/|S_{u_i}|$; $\bar{\boldsymbol{q}}_j \leftarrow \boldsymbol{sumq}_j/|S_{u_j}|$;
11: $sim\_weight \leftarrow 2|S_{u_i} \cap S_{u_j}|/(|S_{u_i}| + |S_{u_j}|)$;
12: $A \leftarrow \sqrt{\sum_{s_k \in S_{u_i} \cap S_{u_j}} (\boldsymbol{q}_{ik} - \bar{\boldsymbol{q}}_i)^2} \cdot \sqrt{\sum_{s_k \in S_{u_i} \cap S_{u_j}} (\boldsymbol{q}_{jk} - \bar{\boldsymbol{q}}_j)^2}$;
13: $sum \leftarrow 0$;
14: for each service $s_k \in S_{u_i} \cap S_{u_j}$ do
15:     $f_1 \leftarrow e^{-\alpha|t_{ik} - t_{jk}|}$; $f_2 \leftarrow e^{-\beta|t_{current} - (t_{ik} + t_{jk})/2|}$;
16:     $sum \leftarrow sum + (\boldsymbol{q}_{ik} - \bar{\boldsymbol{q}}_i)(\boldsymbol{q}_{jk} - \bar{\boldsymbol{q}}_j) \cdot f_1 \cdot f_2$;
17: end for
18: $w_{ij}^{\prime u} \leftarrow sim\_weight \cdot sum/A$;
19: return $w_{ij}^{\prime u}$;
```

IV. A Random Walk Approach to Handle Data Sparsity

A. Similarity Transition

After similarity calculation, similar neighbors should be identified for the target user and each candidate Web service. However, the user-service matrix is usually sparse. Even very active users may invoke just a few Web services, and even very popular Web services may be invoked by only a few users. Therefore, many users may have no co-invoked Web service, and many Web services may also have no common service user. According to Algorithm 1, the time-aware similarity between two users who have no co-invoked Web service or two services which share no common user is calculated as 0. Thus, we cannot get enough similar neighbors (only users or Web services whose similarities are greater than 0 can be considered as similar neighbors [9, 10]) to extract plentiful historical information for reliable and accurate QoS prediction.

However, taking user similarity as an example, no co-invoked Web service does not really mean no similarity. A counter-example is illustrated in Figure 3. Suppose that u_i invoked four Web services, $s_1 \sim s_4$, and u_k invoked $s_3 \sim s_6$, and u_j invoked $s_5 \sim s_8$. According to the above time-aware user similarity model, the similarity between u_i and u_j is calculated as 0 due to $|S_{u_i} \cap S_{u_j}| = 0$. However, from another point of view, u_i and u_k commonly invoked s_3 and s_4, and we assume that their calculated similarity is greater than 0, namely $similarity(u_i, u_k) > 0$. Similarly, u_k and u_j commonly invoked s_5 and s_6, and we also assume $similarity(u_k, u_j) > 0$. In this case, we cannot simply consider that the similarity between u_i and u_j is 0, since they have a common similar neighbor u_k and similarity is a transitive relationship. Therefore, u_i and u_j can establish an indirect similarity relationship via their common similar neighbor u_k. If two users have no co-invoked Web service, but their indirect transitive similarity is greater than 0, they are called $indirect\ similar\ neighbors$.

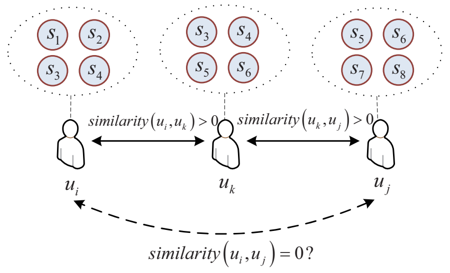

Fig. 3. Transitive Similarity

B. Indirect Similarity Inference

1) Random Walk Algorithm: The random walk algorithm is similar to Google's PageRank algorithm [21], which is originally designed for Web pages ranking. PageRank performs random walk on an entire Web graph, which is constructed by treating Web pages as nodes and the links between pages as edges. Each random step is either walking through one of the outgoing links of the current page or jumping back to the start page with a uniform probability. Finally, each page p is assigned a ranking score, which is the stationary visiting probability of page p during the random walk. In this paper, we apply the random walk algorithm to both the user graph (extracted by treating users as nodes and the similarity associations between users as edges) and the service graph (extracted by treating services as nodes and the similarity associations between services as edges), to identify indirect similar neighbors for the target user and each candidate service for more accurate QoS prediction.

Here, we utilize the user graph to detail the identification of indirect similar neighbors. Similar to PageRank, a higher score a user gets after a random walk implies that (s)he is more similar to the target user (the restart node of the random walk). To perform random walk, we first select $Top\text{-}K$ direct similar neighbors for each user to build a user adjacency matrix as:

$$\boldsymbol{W_u} = \begin{pmatrix} \hat{w}^u_{11} & \cdots & \hat{w}^u_{1M} \\ \vdots & \ddots & \vdots \\ \hat{w}^u_{M1} & \cdots & \hat{w}^u_{MM} \end{pmatrix}, \tag{7}$$

where $\hat{w}^u_{ij}(1 \leq i,j \leq M)$ is set equal to w'^u_{ij} (the time-aware similarity between u_i and u_j) if u_j is one of the $Top\text{-}K$ similar neighbors of u_i and w'^u_{ij} is positive, otherwise $\hat{w}^u_{ij}$ is set to 0. Additionally, $\hat{w}^u_{ii} = 0(1 \leq i \leq M)$ means that the similarity between a user and himself is not considered.

The ranking (column) vector $\boldsymbol{r}$, whose each entry $\boldsymbol{r}_k(1 \leq k \leq M)$ denotes the stationary visiting probability of u_k, is defined as the solution of the following equation:

$$\boldsymbol{r}^n = d \cdot \tilde{\boldsymbol{W}}_{\boldsymbol{u}} \cdot \boldsymbol{r}^{n-1} + (1-d) \cdot \boldsymbol{p}, \tag{8}$$

where $\tilde{\boldsymbol{W}}_{\boldsymbol{u}}$, the column normalized matrix of $\boldsymbol{W_u}$, is the user transition probability matrix, whose each entry $\tilde{w}^u_{ij}$ represents the probability of u_i being the next state when the current state is u_j, and $\boldsymbol{p}$ is a personalized vector defined by Eq. (9), which is concentrated in a single node (the target user), and $d \in (0,1)$ (A common choice for d is 0.85) is a damping factor used to control the probability of the random walk restarting to the personalized vector $\boldsymbol{p}$ (with a probability $1-d$) or moving forward from the current node (with a probability d) at each iteration step. The initial value of $\boldsymbol{r}$ is defined by Eq. (10):

$$\boldsymbol{p}(v) = \begin{cases} 1, & \text{if } v = i \ (u_i \text{ is the target user}), \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

$$\boldsymbol{r}^0 = \frac{1}{|M|} \cdot \boldsymbol{1}_{|M|}, \tag{10}$$

where $|M|$ is the total number of service users. The definition of $\boldsymbol{r}^0$ means that at the beginning of the iteration, all users share a same ranking score. After finite iterations, $\boldsymbol{r}$ converges to a stationary distribution vector, with its each entry r_k $(1 \leq k \leq M, k \neq i)$ denoting a long-term visiting probability of user u_k. Therefore, r_k can be considered as a measurement of the proximity between u_k and the target user u_i, with a larger r_k indicating a closer proximity. Specially, the i^{th} entry r_i (i is the ID of the target user) of the converged $\boldsymbol{r}$ should be set to 0, because we do not consider the proximity between the target user and himself.

2) Similarity Reconstruction: The converged stationary ranking vector $\boldsymbol{r}$ denotes the relative proximities rather than the actual similarities between the target user and his neighbors. The actual similarities are necessary to QoS prediction. Therefore, before QoS prediction, the actual similarities between users should be reconstructed. Suppose that the $Top\text{-}K$ direct similar neighbors of the target user u_i is denoted by $N_{u_i} = \{u_{i_1}, u_{i_2}, \ldots, u_{i_K}\}(\forall l \in \{1,2,\ldots,K\}, 1 \leq i_l \leq M)$, and their time-aware similarities with u_i are $\{\hat{w}^u_{ii_1}, \hat{w}^u_{ii_2}, \ldots, \hat{w}^u_{ii_K}\}$, in which all $\hat{w}^u_{ii_l}(1 \leq l \leq K)$ should be greater than 0. Then, we select the corresponding entries $\{r_{i_1}, r_{i_2}, \ldots, r_{i_K}\}$ from the converged vector $\boldsymbol{r}$. Finally, the reconstructed similarity (row) vector $\boldsymbol{s}_{u_i}$, which represents the actual similarities between the target user u_i and all his (direct and indirect) similar neighbors, is reconstructed as:

$$\boldsymbol{s}_{u_i} = \frac{1}{|N_{u_i}|} \cdot \sum_{l=1}^{K} \frac{\hat{w}^u_{ii_l}}{r_{i_l}} \cdot \boldsymbol{r}^{\mathrm{T}}. \tag{11}$$

We summarize the user similarity inference procedure in Algorithm 2. Line 1 builds the user adjacency matrix. Line 2 and 3 initialize the personalized vector $\boldsymbol{p}$ and the ranking vector $\boldsymbol{r}$. $\boldsymbol{r}_{old}$ is used to record the value of $\boldsymbol{r}$ from the previous iteration and initialized to $[0,\ldots,0]^{\mathrm{T}}$ (line 3). Line 4 to 9 build the transition probability matrix . Line 10 to 13 iteratively perform random walk until $\boldsymbol{r}$ converges. Finally, the reconstructed similarity vector for the target user is calculated and returned (line 14 to 22). The service similarity inference algorithm is similar to Algorithm 2. Due to limited space, no more tautology here.

Algorithm 2 User Similarity Inference

```
Input:
    calculated time-aware user similarities; target user $u_i$, damping factor $d$, threshold $\varepsilon$, $Top\text{-}K$
Output:
    reconstructed similarity vector for $u_i$: $\boldsymbol{s}_{u_i}$
1: build user adjacency matrix $\boldsymbol{W}_u$;
2: $\boldsymbol{p} = [0,\ldots,1,0,\ldots,0]^{\mathrm{T}}$; //only the $i^{th}$ entry of $\boldsymbol{p}$ is 1
3: $\boldsymbol{r} = \frac{1}{M} \cdot [1,\ldots,1]^{\mathrm{T}}$; $\boldsymbol{r}_{old} = [0,\ldots,0]^{\mathrm{T}}$;
4: for $j \leftarrow 1$ to $M$ do
5:     $colsum \leftarrow \sum_{k=1}^{M} \hat{w}^u_{kj}$;
6:     for $k \leftarrow 1$ to $M$ do
7:         $\tilde{w}^u_{kj} \leftarrow \hat{w}^u_{kj}/colsum$; //column normalization
8:     end for
9: end for
10: while $\|\boldsymbol{r} - \boldsymbol{r}_{old}\| \geq \varepsilon$ do
11:     $\boldsymbol{r}_{old} \leftarrow \boldsymbol{r}$;
12:     $\boldsymbol{r} \leftarrow d \cdot \tilde{\boldsymbol{W}}_u \cdot \boldsymbol{r} + (1-d) \cdot \boldsymbol{p}$;
13: end while
14: $r_i = 0$; //$r_i$ is the $i^{th}$ entry of $\boldsymbol{r}$
15: $sum \leftarrow 0$; $N_{u_i} = \emptyset$;
16: for $j \leftarrow 1$ to $M$ do
17:     if $\hat{w}^u_{ij} > 0$ then
18:         $N_{u_i} \leftarrow N_{u_i} \cup \{u_j\}$;
19:         $sum \leftarrow sum + \hat{w}^u_{ij}/r_j$; //$r_j$ is the $j^{th}$ entry of $\boldsymbol{r}$
20:     end if
21: end for
22: $\boldsymbol{s}_{u_i} \leftarrow 1/|N_{u_i}| \cdot sum \cdot \boldsymbol{r}^{\mathrm{T}}$
23: return $\boldsymbol{s}_{u_i}$;
```

V. Time-Aware QoS Prediction

In this section, we utilize the historical QoS experience from all (direct and indirect) similar users and similar services to make QoS prediction. Suppose that u_i and s_j are the target user and a candidate Web service, respectively. Additionally, u_a is a similar user of u_i, and s_b is a similar service of s_j. In order to utilize time information to make more accurate QoS prediction, we define another two time factors: $f_3 = e^{-\gamma_1 |t_{current} - t_{aj}|}$ and $f_4 = e^{-\gamma_2 |t_{current} - t_{ib}|}$, where $\gamma_1, \gamma_2 \geq 0$ are two non-negative decay constants. In addition,

t_{aj} is the timestamp when u_a invoked s_j, and t_{ib} is the timestamp when u_i invoked s_b. f_3 and f_4 both range from 1 to 0. The definition of f_3 means that if the timespan between u_a's invocation on s_j and the current moment is shorter, the QoS experience of u_a on s_j contributes more to the user-based QoS prediction. Similarly, f_4 indicates that if the timespan between u_i's invocation on s_b and the current moment is shorter, the QoS experience of u_i on s_b contributes more to the service-based QoS prediction. Accordingly, the time-aware user-based and service-based QoS prediction results for the target user u_i on the candidate service s_j are given by Eq. (12) and (13):

$$\hat{\boldsymbol{q}}_{ij}^{u} = \bar{\boldsymbol{q}}_i + \frac{\sum\limits_{u_a \in R_{u_i}} \boldsymbol{s}_{u_i}(a) \cdot f_3 \cdot (\boldsymbol{q}_{aj} - \bar{\boldsymbol{q}}_a)}{\sum\limits_{u_a \in R_{u_i}} \boldsymbol{s}_{u_i}(a) \cdot f_3}, \tag{12}$$

$$\hat{\boldsymbol{q}}_{ij}^{s} = \bar{\boldsymbol{q}}_j + \frac{\sum\limits_{s_b \in R_{s_j}} \boldsymbol{s}_{s_j}(b) \cdot f_4 \cdot (\boldsymbol{q}_{ib} - \bar{\boldsymbol{q}}_b)}{\sum\limits_{s_b \in R_{s_j}} \boldsymbol{s}_{s_j}(b) \cdot f_4}, \tag{13}$$

where R_{u_i} and R_{s_j} are the sets of similar neighbors of u_i and s_j, respectively. Additionally, $\boldsymbol{s}_{u_i}(a)$ is the a^{th} entry of $\boldsymbol{s}_{u_i}$ (the reconstructed similarity vector for u_i), and $\boldsymbol{s}_{s_j}(b)$ is the b^{th} entry of $\boldsymbol{s}_{s_j}$. If γ_1 and γ_2 are both set to 0, the time-aware QoS prediction approach degenerates into the original QoS prediction method proposed in [9, 10]. So the latter is just a special case of the former.

In order to merge the two kinds of prediction results into a consolidated prediction result, two confidence weights con_u and con_s [9, 10] are utilized to balance the user-based and service-based predicted values, which are defined as:

$$con_u = \sum_{u_a \in R_{u_i}} \frac{\boldsymbol{s}_{u_i}(a)}{\sum\limits_{u_a \in R_{u_i}} \boldsymbol{s}_{u_i}(a)} \cdot \boldsymbol{s}_{u_i}(a), \tag{14}$$

$$con_s = \sum_{s_b \in R_{s_j}} \frac{\boldsymbol{s}_{s_j}(b)}{\sum\limits_{s_b \in R_{s_j}} \boldsymbol{s}_{s_j}(b)} \cdot \boldsymbol{s}_{s_j}(b). \tag{15}$$

Then, the final prediction result is predicted as:

$$\hat{\boldsymbol{q}}_{ij} = h_u \cdot \hat{\boldsymbol{q}}_{ij}^{u} + h_s \cdot \hat{\boldsymbol{q}}_{ij}^{s}, \tag{16}$$

where h_u and h_s [9, 10] are the weights of user-based and service-based prediction results, and defined as:

$$h_u = \frac{\lambda \cdot con_u}{\lambda \cdot con_u + (1-\lambda) \cdot con_s}, \tag{17}$$

$$h_s = \frac{(1-\lambda) \cdot con_s}{\lambda \cdot con_u + (1-\lambda) \cdot con_s}, \tag{18}$$

where λ ($0 \leq \lambda \leq 1$) is employed to determine how much the final prediction result relies on the user-based and the service-based prediction results. For more details about con_u, con_s and λ, please refer to [9, 10].

After QoS prediction, Web service recommendation can be made. When a target user requests a Web service recommendation by providing some functional descriptions, the service which fits the required functionalities and has optimal comprehensive QoS performance will be recommended to the user.

VI. Experiments

In this section, several experiments are conducted to validate the effectiveness of our time-aware and data sparsity tolerant approach for Web service recommendation.

A. Dataset

We adopt a freely available dataset provided by Zhang et al. [22] for experiments. They employed 142 distributed computers from Planet-Lab[1] to evaluate the response time of 4532 Web services during 64 continuous time intervals, with each time interval lasting for 15 minutes. So, the dataset contains 64 142×4532 user-service matrices for response time.

B. Data Processing

We randomly select 140 users and 140 services from the dataset to build 64 140×140 user-service matrices for response time, with each time interval providing one 140×140 matrix. Some initially selected Web services should be replaced by others due to invocation failure. Then, we randomly divide the matrix of the 64^{th} (latest) time interval into two parts, one as the training matrix, which contains about 60% data of the whole matrix, and the other as the test matrix, which contains the remaining 40% data. Afterwards, we randomly select 90% entries of the training matrix and replace each of them with the corresponding QoS value observed during any of the previous 63 time intervals. Thus, a new training matrix is constructed, which contains training data from different time intervals. The 64^{th} time interval is seen as the current moment (the recommendation time). Since we do not need to predict missing QoS values before the current time, the test matrix only contains the QoS data observed at the 64^{th} time interval.

C. Evaluation Metric

Mean Absolute Error (MAE) is widely used to measure the prediction accuracy of CF methods, which is defined as:

$$MAE = \frac{1}{N} \cdot \sum_{i,j} |\hat{r}_{ij} - r_{ij}|, \tag{19}$$

where r_{ij} and $\hat{r}_{ij}$ are the actual QoS (response time here) value and the predicted QoS value of s_j observed by u_i, respectively, and N is the total number of predicted values. A smaller MAE indicates a more accurate QoS prediction. Here we also use the MAE metric to measure the prediction accuracy of our approach.

D. Performance Comparison

First, we randomly remove some entries of the training matrix to make the density of training data vary from 100% to 40%, with a step value of -20%, to simulate different sparsity levels of training data. The remaining training data is used for time-aware similarity calculation. Then, we choose the top K most similar neighbors for each user or service as its direct similar neighbors. A similar neighbor should be removed from the $Top\text{-}K$ similar neighbors if its similarity is equal to or smaller than 0. For the sake of simplicity, the values of $Top\text{-}K$ for each user and each service are set equal, simultaneously

[1]http://www.planet-lab.org

varying from 2 to 24, with a step value of 2. The value of λ is fixed at 0.5. Additionally, the four time decay constants α, β, γ_1 and γ_2 are also set equal.

We utilize the hybrid CF named "WSRec" [9, 10] as the fundamental QoS prediction algorithm. If "WSRec" performs random walk while seeking for indirect similar users and services, it is named "WSRec with Hybrid Random Walk". If "WSRec" not only performs hybrid random walk, but also integrates time information into similarity measurement and QoS prediction, it is named "Time-Aware WSRec with Hybrid Random Walk", namely the approach proposed in this paper.

We first validate the effectiveness of the time-aware approach, which integrates time information into both the similarity computation and the QoS prediction. We compare the MAE of "WSRec with Hybrid Random Walk" with that of "Time-Aware WSRec with Hybrid Random Walk". Each experiment loops 50 times and the average values are reported. Figure 4 shows the performance comparison of the two algorithms under different training data densities. It can be observed that "Time-Aware WSRec with Hybrid Random Walk" outperforms "WSRec with Hybrid Random Walk" obviously. This indicates that the time information is valuable to Web service recommendation. Here, the duration of each time interval (15 minutes) is simply denoted as 1 and the time decay constants α, β, γ_1 and γ_2 are all set to 0.085 due to the optimal performance the value produces.

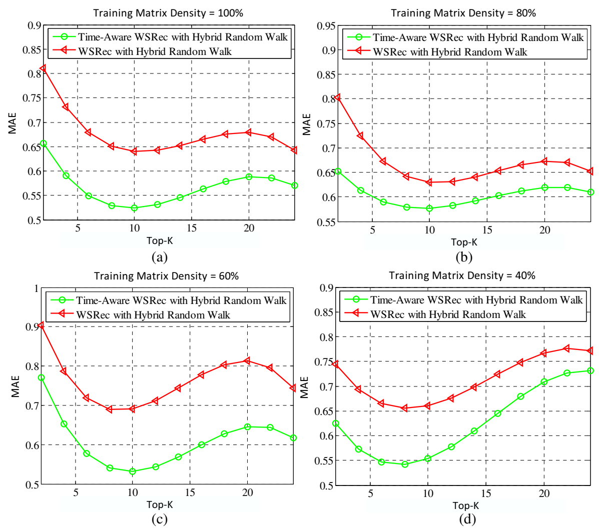

Fig. 4. Evaluation of Time-Aware Approach

Next, we corroborate the effectiveness of the hybrid random walk algorithm for alleviating data sparsity, by comparing the performance of the four algorithms: "Time-Aware WSRec without Random Walk", "Time-Aware WSRec with User Random Walk", "Time-Aware WSRec with Service Random Walk" and "Time-Aware WSRec with Hybrid Random Walk". The first algorithm is the basic "WSRec" introducing time information into similarity calculation and QoS prediction. The second algorithm is based on the first one, and additionally performs random walk while discovering indirect similar neighbors for the target user. Similarly, the third algorithm is also based on the first one and performs random walk to identify indirect similar neighbors for the candidate service. The last one is the combination of the second and the third algorithms. All experiments loop 50 times and we report the average values. The performance comparisons of the four algorithms under different training data densities are shown in Figure 5.

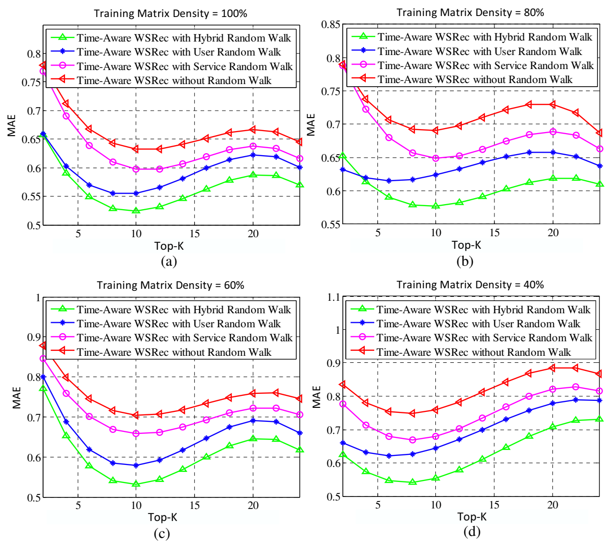

Fig. 5. Evaluation of Random Walk Approach

We can observe that under different training data densities (from 100% to 40%), the algorithms performing either user or service random walk outperform that without random walk. And as described in Section IV, the hybrid random walk is the combination of the user random walk and the service random walk, so it deserves to achieve better performance than the latter two algorithms. Figure 5(a) to (d) all validate this point, because the MAE of the hybrid random walk is obviously smaller than that of the user or service random walk. Besides, even if the training data is very sparse (40%) (Figure 5(d)), the hybrid random walk algorithm can still improve the prediction accuracy remarkably compared with the algorithm without random walk. This shows the powerful capability of the hybrid random walk algorithm to alleviate data sparsity.

E. The Impact of Time Decay Constants

As described in Section III and V, we employ total 4 time decay constants and set them equal. In this section, we investigate the influence of different decay constant values on QoS prediction accuracy. Figure 6(a) shows a series of exponential decay functions, with the decay constants λ varying from 0 to 0.4 and a corresponding step value 0.01. Its x-axis contains 64 time intervals. The shape of the function curve changes violently when λ changes from 0 to 0.3. But when λ is greater than 0.3, the curve shape changes little. So, we investigate the performance variation with the decay constants ranging from 0 to 0.3, with a step value 0.005. Figure 6(b) shows the average MAE reduction of "Time-Aware WSRec with Hybrid Random Walk" compared with "WSRec with Hybrid random walk" with different decay constant values. A larger MAE

reduction indicates a greater performance improvement. When decay constants are 0, there is no performance improvement (MAE reduction is 0). When decay constants range from 0 to 0.085, the MAE reduction of the time-aware approach climbs quickly. When decay constants are 0.085, the MAE reduction reaches its peak. When decay constants are greater than 0.085, the MAE reduction descends slowly. This is why we choose 0.085 as the values of the time decay constants in Section VI-D. So, choosing a proper value for time decay constants is conducive to getting satisfied system performance.

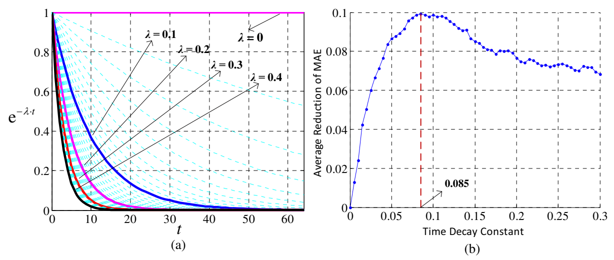

Fig. 6. The Impact of Time Decay Constants

VII. Conclusions

In this paper, we incorporate time information into Web service recommendation for more accurate QoS prediction, and perform a hybrid personalized random walk algorithm to handle data sparsity. The effectiveness of our approach is validated by experiments. However, the work presented in this paper still has some limitations, such as simplifying the problem by overriding a QoS value by a more recent one (maybe some kind of weighted average can be adopted), not taking the properties of timestamps into consideration, and lacking effective mechanisms to incorporate the time dimension into a generalized context-aware technical framework for Web service recommendation. Therefore, in our future work, we will try to solve these problems to further improve the service recommendation performance.

Acknowledgment

This work is fully supported by the Chinese 863 project (2012AA011206).

References

[1] S. Dustdar and B. J. Krämer, "Introduction to special issue on service oriented computing (soc)," *ACM Transactions on the Web (TWEB)*, vol. 2, no. 2, p. 10, 2008.

[2] L. Kuang, Y. Xia, and Y. Mao, "Personalized services recommendation based on context-aware qos prediction," in *IEEE 19th International Conference on Web Services (ICWS'12)*. IEEE, 2012, pp. 400–406.

[3] L. Yao, Q. Z. Sheng, A. Segev, and J. Yu, "Recommending web services via combining collaborative filtering with content-based features," in *Web Services (ICWS), 2013 IEEE 20th International Conference on*. IEEE, 2013, pp. 42–49.

[4] J. Huang and C. Lin, "Agent-based green web service selection and dynamic speed scaling," in *Web Services (ICWS'13), 2013 IEEE 20th International Conference on*. IEEE, 2013, pp. 91–98.

[5] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.

[6] R. Jin, J. Y. Chai, and L. Si, "An automatic weighting scheme for collaborative filtering," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2004, pp. 337–344.

[7] M. Deshpande and G. Karypis, "Item-based top-n recommendation algorithms," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 143–177, 2004.

[8] 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*. ACM, 2001, pp. 285–295.

[9] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Wsrec: A collaborative filtering based web service recommender system," in *IEEE International Conference on Web Services (ICWS'09)*. IEEE, 2009, pp. 437–444.

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

[11] M. Tang, Y. Jiang, J. Liu, and X. Liu, "Location-aware collaborative filtering for qos-based service recommendation," in *Web Services (ICWS), 2012 IEEE 19th International Conference on*. IEEE, 2012, pp. 202–209.

[12] Y. Shen, J. Zhu, X. Wang, L. Cai, X. Yang, and B. Zhou, "Geographic location-based network-aware qos prediction for service composition," in *Web Services (ICWS), 2013 IEEE 20th International Conference on*. IEEE, 2013, pp. 66–74.

[13] M. Papagelis, D. Plexousakis, and T. Kutsuras, "Alleviating the sparsity problem of collaborative filtering using trust inferences," in *Trust management*. Springer, 2005, pp. 224–239.

[14] H. Yildirim and M. S. Krishnamoorthy, "A random walk method for alleviating the sparsity problem in collaborative filtering," in *Proceedings of the 2008 ACM conference on Recommender systems*. ACM, 2008, pp. 131–138.

[15] S. Shang, S. R. Kulkarni, P. W. Cuff, and P. Hui, "A randomwalk based model incorporating social information for recommendations," in *Machine Learning for Signal Processing (MLSP), 2012 IEEE International Workshop on*. IEEE, 2012, pp. 1–6.

[16] Z. Zhang, D. D. Zeng, A. Abbasi, J. Peng, and X. Zheng, "A random walk model for item recommendation in social tagging systems," *ACM Transactions on Management Information Systems (TMIS)*, vol. 4, no. 2, p. 8, 2013.

[17] Y. Zhou, L. Liu, C.-S. Perng, A. Sailer, I. Silva-Lepe, and Z. Su, "Ranking services by service network structure and service attributes," in *Web Services (ICWS'13), 2013 IEEE 20th International Conference on*. IEEE, 2013, pp. 26–33.

[18] L. Liu, N. Mehandjiev, and D.-L. Xu, "Multi-criteria service recommendation based on user criteria preferences," in *Proceedings of the fifth ACM conference on Recommender systems*. ACM, 2011, pp. 77–84.

[19] M. Jiang, P. Cui, F. Wang, Q. Yang, W. Zhu, and S. Yang, "Social recommendation across multiple relational domains," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 1422–1431.

[20] H. J. Ahn, "A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem," *Information Sciences*, vol. 178, no. 1, pp. 37–51, 2008.

[21] A. N. Langville and C. D. Meyer, *Google's PageRank and beyond: The science of search engine rankings*. Princeton University Press, 2006.

[22] Y. Zhang, Z. Zheng, and M. R. Lyu, "Wspred: A time-aware personalized qos prediction framework for web services," in *IEEE 22nd International Symposium on Software Reliability Engineering (ISSRE'11)*. IEEE, 2011, pp. 210–219.