World Wide Web (2014) 17:33–57
DOI 10.1007/s11280-012-0186-0

QoS-aware service selection via collaborative QoS evaluation

Qi Yu

Received: 2 November 2011 / Revised: 21 August 2012 /
Accepted: 23 August 2012 / Published online: 14 September 2012

Abstract We present in this paper a novel collaborative filtering based scheme for evaluating the QoS of large scale Web services. The proposed scheme automates the process of assessing the QoS of a priori unknown service providers and thus facilitates service users in selecting services that best match their QoS requirements. Most existing service selection approaches ignore the great diversity in the service environment and assume that different users receive identical QoS from the same service provider. This may lead to inappropriate selection decisions as the assumed QoS may deviate significantly from the one actually received by the users. The collaborative filtering based approach addresses this issue by taking the diversity into account instead of uniformly applying the same QoS value to different users. They predict a user's QoS on an unknown service by exploiting the historical QoS experience of similar users. Nevertheless, when only limited historical QoS data is available, these approaches either fail to make any predictions or make very poor ones. The cornerstone of the proposed QoS evaluation scheme is a Relational Clustering based Model (or **RCM**) that effectively addresses the data scarcity issue as stated above. Experimental results on both real and synthetic datasets demonstrate that the proposed scheme can more accurately predict the QoS on unknown service providers. The efficient performance also makes it applicable to QoS evaluation for large scale Web services.

Keywords Web service · service selection · QoS · collaborative filtering

Q. Yu (✉)
College of Computing and Information Sciences, Rochester Institute of Technology,
Rochester, NY 14623, USA
e-mail: qi.yu@rit.edu

1 Introduction

Service computing is gaining momentum nowadays by offering an attractive paradigm for various business organizations to deliver their functionalities. Within this paradigm, the key functionalities are wrapped into *Web services*, which can be programmatically accessed over the Web in a standard way. Many Web services have been developed and deployed on the Web over the past few years. This trend has been further expedited by the emergence of the cloud, which offers a powerful and cost-effective platform to host a large number of Web services. While the ever increasing number of Web services holds significant promise, a key challenge arises: selecting a user desired service becomes nontrivial as users are easily overloaded by a vast amount candidates.

Achieving the full potential of service computing requires the ability to efficiently and accurately retrieve the services that meet users' requirements. A user's requirements may cover both the functional and nonfunctional aspects of a service. A number of Web service search engines have been developed that help users locate services with their required functionality [4, 13, 23, 28]. Information retrieval [11, 19, 24] and semantic Web technologies [3, 20, 29] have been employed to achieve more precise and reliable search results. QoS-aware service selection has also received considerable attention as multiple service providers may compete to offer the same functionality but with different QoS. As the number of services has gone far beyond the reach of any manual search, plenty of algorithms have been developed that can help users efficiently locate providers that best match their QoS preferences [35, 37].

A key issue with most existing service selection approaches is that they ignore the disparity of service users and do not differentiate the QoS delivered from the same provider to different users. The QoS information is either obtained from the service description of the provider or from some monitoring system. Nevertheless, service providers may not always deliver according to their "promised" quality [30]. Furthermore, a user may access the service from an environment that is a very different with the monitoring system in terms of location and network condition, etc. [40]. Therefore, the actual QoS received by the user may deviate dramatically from the one used by the service selection algorithms. As a result, these algorithms will lead to inappropriate selection decisions.

Collaborative filtering techniques have been recently applied to service selection [16, 25, 39, 40]. Collaborative filtering aims to more accurately estimate the QoS of unknown service providers by explicitly taking the user disparity into account. Most existing collaborative filtering approaches fall into two categories: memory based and model based. The memory based approaches can be further divided into user based and item based approaches. The intuitive idea is to identify "similar" users with a given active user and assign higher weights to these users when aggregating their QoS values to predict unknown QoS. The *similarity* between two users is measured based on the QoS values on the common services they have used. All existing efforts that employed collaborative filtering for service selection fall into the memory based category. A key issue with the memory based approach is data sparsity. Since only users that have used the same services can be regarded as "similar", it may be difficult to identify sufficient number of users that are similar to the active user from a very sparse QoS dataset. As a common service user may only use a limited number of Web services, a typical QoS dataset is expected to be very sparse. In this regard, these approaches either fail to make any predictions or make very poor ones, which is also confirmed by our empirical study on a real QoS dataset.

In this paper, we propose a Relational Clustering based collaborative filtering Model (or **RCM**) to predict the QoS of a priori unknown service providers. QoS evaluation of unknown service providers is a key step toward accurately selecting services that match users' QoS preferences. Relational clustering leverages the interaction information between the two participating entities, i.e., users and services. More specifically, the historical QoS data is denoted by a relation matrix, where each row represents a user, each column represents a service, and each entry signifies the interaction information (i.e., QoS) between the user and the service. Based on the available interaction information, users and services are divided into a set of user and service clusters, respectively. Besides cluster coefficients, a second key component resulted from relational clustering is a cluster interaction matrix (a.k.a. prototype matrix) that captures the interactions between user and service clusters. As the interactions between clusters of users and services are much more likely to occur than the interactions between individual users and services, the proposed relational clustering based model can more effectively deal with data sparsity. The unknown QoS is finally obtained as the convex combination of the cluster interaction matrix. It is worth to note that **RCM** does not aim to completely avoid the cold start issue in collaborative filtering. Instead, due its relational clustering nature, **RCM** is less sensitive to the sparsity of the datasets. This is critical for QoS evaluation as most real-world QoS datasets are very sparse. We demonstrate the effectiveness of RCM through an extensive experimental study and comparison with other collaborative filtering techniques.

Our overall contribution is summarized as follows:

- We propose a novel relational clustering based model, **RCM**, which is applicable to incomplete QoS data matrices, for accurate QoS evaluation.
- **RCM** is robust to very sparse data matrices due to its relational clustering nature, which makes it especially suitable for QoS evaluation in the service environment.
- We present the system architecture and use a case study to demonstrate how **RCM** can be used in service selection.
- We conduct an extensive experimental study to assess the effectiveness and efficiency of **RCM**.

The remainder of the paper is organized as follows. We provide an overview of collaborative filtering and discuss how it can be used for Web service QoS evaluation in Section 2. This helps set the stage for the discussion of the proposed relational clustering based model for QoS evaluation, which is detailed in Section 3. We present the algorithm that constructs **RCM** in Section 4. We present the system architecture and use a small scale case study to illustrate how the proposed RCM model helps achieve personalized service selection in Section 5. We evaluate the effectiveness and efficiency of the proposed model by using both real and synthetic datasets in Section 6. We give an overview of related works and discuss the difference with the proposed **RCM** in Section 7. We provide our concluding remarks and identify some important future directions in Section 8.

2 Preliminaries

In this section, we start by describing an illustrative example, which helps further motivate the proposed research. We then provide an overview of collaborative filtering techniques and discuss how to apply them to Web service QoS evaluation.

2.1 An illustrative example

Consider five (5) example services and five (5) users who have invoked some of these services. Assume that the active user is seeking a service with a high availability. Table 1 gives the historical QoS data on availability from these users. ϕ means that the user has not invoked the corresponding service, so no QoS data is recorded. Assume that there are two services, $service_2$ and $service_4$, which provide the user desired functionality. Since they may offer different availability, the task is to select a service with a higher availability.

The first important observation that we make from the historical QoS data is that different users may receive dramatically different QoS from the same service. For example, the availability of $service_1$ varies from 0.75 to 0.98 across the five different users. As discussed in Section 1, this may be caused by the differences of the location and network condition of the users. For example, all users except for $user_3$ perceived a high availability from $service_1$. By taking a closer look at $user_3$, we find out that it also perceived a relatively low availability from other services, (e.g., $service_2$ and $service_3$). In this case, it is highly probable that this user comes from a computing environment with an unstable network. This observation also implies that applying a uniform QoS value (obtained from the service description or a monitoring system) to different users does not consider the disparity of users. Hence, using such QoS for service selection may result in inappropriate selection decisions.

Collaborative filtering explicitly considers user disparity, which aims to accurately predict the QoS of unknown service providers. It identifies users that have similar QoS experiences with the active user and leverages the QoS they have perceived to make predictions. For example, two users that are most similar to the active user in Table 1 are $user_1$ and $user_4$. In particular, both $user_1$ and the active user have invoked $service_1$ and $service_3$ and perceived similar QoS from these two services. Similarly, both $user_4$ and the active user have invoked $service_1$ and $service_5$ and perceived similar QoS from them. Since $user_1$ and $user_4$ have invoked $service_2$ and $service_4$, respectively, we can use their QoS data to predict QoS that the active user may receive from these two services.

A typical QoS dataset will be in a much larger scale, which includes many more users and services. Therefore, more than one similar users will be identified. Multiple

Table 1 Historical QoS data on availability.

	$service_1$	$service_2$	$service_3$	$service_4$	$service_5$
user$_1$	0.95	0.8	0.8	ϕ	ϕ
$user_2$	0.98	0.96	ϕ	0.85	0.95
$user_3$	0.75	0.75	0.72	ϕ	ϕ
user$_4$	0.93	ϕ	ϕ	0.98	0.85
Active user	0.93	**?**	0.82	**?**	0.87

QoS values are usually aggregated based on certain similarity functions (which will be introduced next) to produce more accurate predictions.

2.2 Collaborative filtering for web service QoS evaluation

The illustrative example captures some intuitive ideas about how collaborative filtering can be applied to QoS evaluation. A central component of collaborative filtering is to identify users who have similar QoS experiences with the active user. Pearson Correlation Coefficient is one of the most widely used similarity functions by existing collaborative filtering systems. More specifically, the similarity between two users i and j can be calculated using Pearson Correlation Coefficient as:

$$sim(i, j) = \frac{\sum_k (v_{i,k} - \overline{v}_i)(v_{j,k} - \overline{v}_j)}{\sqrt{\sum_k (v_{i,k} - \overline{v}_i)^2 \sum_k (v_{j,k} - \overline{v}_j)^2}} \tag{1}$$

where $v_{i,k}$ denotes the QoS value that user i received from service k, $\overline{v}_i$ denotes the average QoS that user i received from all the services s/he has invoked, and the summation is over all the services that have been invoked by both i and j. $sim(i, j)$ gives a value in $[-1, 1]$, where a positive value implies that i and j are correlated, 0 implies independent. a negative value implies anti-correlated.

Assume that we want to predict the QoS that an active user i may receive from an unknown service provider p. The first step is to identify the set of users that have invoked p. We then compute the similarity between these users and the active user using (1) and identify similar users. This can be achieved by setting a threshold value, say δ. A user j is regarded as a similar user only when $sim(i, j) \geq \delta$. After all similar users are identified, the QoS can be predicted as:

$$v_{i,p} = \overline{v}_i + \frac{\sum_{j \in \mathcal{S}_i} sim(i, j)(v_{j,p} - \overline{v}_j)}{\sum_{j \in \mathcal{S}_i} sim(i, j)} \tag{2}$$

where $\mathcal{S}_i$ denotes the set of similar users of i, i.e., $\mathcal{S}_i = \{j | sim(i, j) \geq \delta\}$.

The above approach is also known as *user based* collaborative filtering. The *item based* approach follows a very similar idea but identifies services that deliver similar QoS as the unknown service provider p. Then, these similar services' QoS behavior will be leveraged to make the prediction. More specifically, using Pearson Correlation Coefficient, the similarity between two services p and q can be calculated as:

$$sim(p, q) = \frac{\sum_l (v_{l,p} - \overline{v}_p)(v_{l,q} - \overline{v}_q)}{\sqrt{\sum_l (v_{l,p} - \overline{v}_p)^2 \sum_l (v_{l,q} - \overline{v}_q)^2}} \tag{3}$$

where $\overline{v}_p$ denotes the average QoS that service p delivered to all the users who have invoked p and the summation is over all the users that have invoked by both p and q. The prediction can be made similarly as in (2):

$$v_{i,p} = \overline{v}_p + \frac{\sum_{q \in \mathcal{U}_p} sim(p, q)(v_{i,q} - \overline{v}_q)}{\sum_{q \in \mathcal{U}_p} sim(p, q)} \tag{4}$$

where $\mathcal{U}_p$ denotes the set of similar services of p, i.e., $\mathcal{U}_p = \{q|sim(p, q) \geq \delta'\}$ and δ' is a threshold value. Some hybrid approaches may also be exploited, which combine both the user based and item based approaches. For example, a hybrid collaborative filtering method is presented in [40] that integrates prediction values specified in (2) and (4) using the prediction confidence as weights.

Apart from Pearson Correlation Coefficient, cosine similarity is another widely used similarity measure in existing collaborative filtering systems. In particular, each user i is modeled as a vector $\mathbf{v}_i \in \mathbb{R}^n$, where the p-th component $v_{i,p}$ denote the QoS that user i received from service p and $v_{i,p}$ is set to 0 if i has not invoked p. The similarity between users i and j can be calculated as:

$$sim(i, j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{||\mathbf{v}_i||\ ||\mathbf{v}_j||} = \frac{\sum_k v_{i,k} v_{j,k}}{\sqrt{\sum_k v_{i,k}^2 \sum_k v_{j,k}^2}} \tag{5}$$

where $||\mathbf{v}_i||$ is the Euclidean norm of vector $\mathbf{v}_i$. Once the similarity is computed, (2) and (4) can be used to make QoS predictions.

3 The relational clustering based model for QoS evaluation

We present our relational clustering based collaborative filtering model in this section. As can be seen from Section 2, collaborative filtering relies on the identification of users who have similar QoS experiences with the active user and uses their QoS behavior to make the prediction. The similarity of users is essentially decided by their *interaction* result (i.e., QoS) with different services. Relational clustering, which follows a similar rationale, also exploits the interaction information to determine similarity of participating entities (e.g., users or services). Instead of identifying similar users (or services) for each given user (or service), relational clustering seeks a global structure that partition users and services into a set of user and service clusters, respectively. This enables us to approximate the interaction behavior (i.e., QoS) between individual users and services though a convex combination of the interaction between user and service clusters.

3.1 Problem formulation and notations

Under relational clustering, the historical QoS data is modeled as a relational matrix $\mathbf{X}_{m\times n}$, which captures the interaction information between m users and n services. Subscripts are used to denote the sizes of matrices (e.g., $\mathbf{X}_{m\times n}$ means a matrix with m rows and n columns). Each entry $\mathbf{X}(i, j) \in \mathbf{X}$ represents the QoS that user i received from service j. The i-th row of $\mathbf{X}$, denoted by $\mathbf{X}(i, :)$, corresponds to the i-th user while the j-th column of $\mathbf{X}$, denoted by $\mathbf{X}(:, j)$, corresponds to the j-th service. Since $\mathbf{X}$ contains missing entries, we use $\mathcal{O}$ to denote indices of observed QoS entries, i.e., $\mathcal{O} = \{(i, j)|\mathbf{X}(i, j)$ is an observed QoS entry$\}$. Further, $\mathbf{X}^2_{\mathcal{O}} = \sum_{(i,j)\in\mathcal{O}} \mathbf{X}^2(i, j)$ is the sum of the square of all observed QoS entries in $\mathbf{X}$.

Table 2 Notations.

Notation	Description
$\mathbf{X}$	A matrix
$\mathcal{O}$	A set
$\mathbf{X}'$	The transpose of matrix $\mathbf{X}$
$\mathbf{X}(i, j)$	The element at the i-th row and j-th column of matrix $\mathbf{X}$
$\mathbf{X}(i, :)$	The i-th row of matrix $\mathbf{X}$
$\mathbf{X}(:, j)$	The j-th column of matrix $\mathbf{X}$
R_p	The p-th cluster
$\mathbf{x}_j$	A column vector

Based on the above setting and notations (Table 2), we formally formulate the problem as follows:

Definition 1 Given a relational matrix $\mathbf{X}_{m\times n}$ and two positive integer values k and l, relational clustering seeks an asymmetric convex encoding of matrix $\mathbf{X}_{m\times n}$ by optimizing the following objective function:

$$\min_{\substack{\mathbf{R}\in\mathbb{R}_+^{m\times k},\mathbf{R1}=\mathbf{1}\\ \mathbf{C}\in\mathbb{R}_+^{n\times l},\mathbf{C1}=\mathbf{1}}} (\mathbf{X}_{m\times n} - \mathbf{R}_{m\times k}\mathbf{P}_{k\times l}\mathbf{C}'_{l\times n})^2_{\mathcal{O}} \tag{6}$$

where $\mathbf{1}$ is a vector of 1's, $\mathbf{C}'$ is the transpose of $\mathbf{C}$, and $\mathbf{R} \in \mathbb{R}_+^{m\times k}$ means that all entries in $\mathbf{R}$ are nonnegative.

In order to prove some important properties of the proposed model, we first transform the objective function in (6) into a different format. We define $\mathbf{O}$ as an indicator matrix, where $\mathbf{O}(i, j)$ is 1 if $\mathbf{X}(i, j)$ is an observed QoS entry and 0 if otherwise. Hence, solving (6) is equivalent to solving the following objective function:

$$\min_{\substack{\mathbf{R}\in\mathbb{R}_+^{m\times k},\mathbf{R1}=\mathbf{1}\\ \mathbf{C}\in\mathbb{R}_+^{n\times l},\mathbf{C1}=\mathbf{1}}} \left\|\mathbf{O}_{m\times n} \circ (\mathbf{X}_{m\times n} - \mathbf{R}_{m\times k}\mathbf{P}_{k\times l}\mathbf{C}'_{l\times n})\right\|^2 \tag{7}$$

where $||\mathbf{A}|| = \sqrt{\sum_{ij} \mathbf{A}^2(i, j)}$ is matrix norm and $\circ$ is entry-wise product (or the Schur product) of two matrices of the same dimensions.

3.2 User and service clustering

We have claimed that the asymmetric convex encoding of matrix $\mathbf{X}$ has the effect of clustering the rows and columns of $\mathbf{X}$ into a set of row and column clusters, respectively. It has been shown in [10], with an orthogonal constraint, performing nonnegative factorization on a complete matrix is equivalent to K-means clustering. We follow a similar rationale but apply it to an incomplete relational matrix $\mathbf{X}$ in order to demonstrate the connection of the proposed model with clustering. In order to show this, we first make a modification on the constraint of the objective function in (7). That is, instead of having $\mathbf{R1} = \mathbf{1}$ and $\mathbf{C1} = \mathbf{1}$, we enforce an orthogonal constraint on $\mathbf{R}$ and $\mathbf{C}$: $\mathbf{R}'\mathbf{R} = \mathbf{I}$ and $\mathbf{C}'\mathbf{C} = \mathbf{I}$ to make $\mathbf{C}$ and $\mathbf{R}$ the following cluster indicator matrices.

$$\mathbf{C}(j, q) = \begin{cases} \frac{1}{\sqrt{|C_q|}} & \text{if } \mathbf{x}_j \in C_q \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$\mathbf{R}(i, p) = \begin{cases} \frac{1}{\sqrt{|R_p|}} & \text{if } \mathbf{x}_i \in R_p \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where C_q and R_p denote the q-th and p-th column and row clusters, respectively. We then generalize result to the original convex encoding constraint.

We first show that optimizing the following objective function is equivalent to performing K-mean clustering on the columns of an incomplete relational matrix $\mathbf{X}$.

$$\min_{\mathbf{C}\in\mathbb{R}_+^{n\times l}, \mathbf{C}'\mathbf{C}=\mathbf{I}} J_C = \left\| \mathbf{O}_{m\times n} \circ (\mathbf{X}_{m\times n} - \mathbf{S}_{m\times l}\mathbf{C}'_{l\times n}) \right\|^2 \tag{10}$$

where $\mathbf{S} = \mathbf{RP}$. In order to do this, we need to further enforce a row normalization on $\mathbf{C}$, which gives $\sum_{q=1}^{l} \tilde{\mathbf{C}}(j, q) = 1$ where $\tilde{\mathbf{C}} = \mathbf{Cdiag}(\sqrt{|C_1|}...\sqrt{|C_l|})$. Accordingly, $\mathbf{S}$ will be updated to $\tilde{\mathbf{S}} = \mathbf{Sdiag}^{-1}(\sqrt{|C_1|}...\sqrt{|C_l|})$ to ensure that $\mathbf{SC}' = \tilde{\mathbf{S}}\tilde{\mathbf{C}}'$. Hence, we can reformulate J_C as follows:

$$J_C = \sum_{j=1}^{n} \left\| \mathbf{o}_j \circ [\mathbf{x}_j - \sum_{q=1}^{l} \tilde{\mathbf{C}}(j, q)\tilde{\mathbf{s}}_q] \right\|^2 \tag{11}$$

$$= \sum_{j=1}^{n} \left\| \sum_{q=1}^{l} \tilde{\mathbf{C}}(j, q)[\mathbf{o}_j \circ (\mathbf{x}_j - \tilde{\mathbf{s}}_q)] \right\|^2 \tag{12}$$

$$= \sum_{j=1}^{n} \sum_{q=1}^{l} \tilde{\mathbf{C}}(j, q) \left\| \mathbf{o}_j \circ (\mathbf{x}_j - \tilde{\mathbf{s}}_q) \right\|^2 \tag{13}$$

$$= \sum_{q=1}^{l} \sum_{\mathbf{x}_j \in C_q} \left\| \mathbf{o}_j \circ (\mathbf{x}_j - \tilde{\mathbf{s}}_q) \right\|^2 \tag{14}$$

where $\mathbf{o}_j$ is the j-th column of $\mathbf{O}$ and $\tilde{\mathbf{s}}_q$ is the q-th column of $\tilde{\mathbf{S}}$. From (11) to (12), we use the fact that $\sum_{q=1}^{l} \tilde{\mathbf{C}}(j, q) = 1$. Since there is only one non-zero element in each row of $\tilde{\mathbf{C}}(j, q)$, we have $\tilde{\mathbf{C}}(j, q) = 0$ or 1. This gives $\tilde{\mathbf{C}}^2(j, q) = \tilde{\mathbf{C}}(j, q)$, which leads (12) to (13).

K-means clustering minimizes (14), via which $\tilde{\mathbf{s}}_q$ will converge to the centroid of cluster C_q. Next, we show that minimizing J_C does lead $\tilde{\mathbf{s}}_q$ to the centroid of cluster C_q

$$J_C = \sum_{(i,j)\in\mathcal{O}} [\mathbf{X}(i,j) - \mathbf{SC}'(i,j)]^2 \tag{15}$$

$$= \sum_{i,j} \mathbf{O}(i,j)[\mathbf{X}(i,j) - \mathbf{SC}'(i,j)]^2 \tag{16}$$

$$= \mathrm{Tr}[(\mathbf{O}' \circ (\mathbf{X} - \mathbf{SC}')')(\mathbf{O} \circ (\mathbf{X} - \mathbf{SC}'))] \tag{17}$$

$$= \mathrm{Tr}[(\mathbf{O}' \circ (\mathbf{X} - \mathbf{SC}')')(\mathbf{X} - \mathbf{SC}')] \tag{18}$$

$$= \mathrm{Tr}[\mathbf{X}'(\mathbf{O} \circ \mathbf{X}) - 2\mathbf{S}'(\mathbf{O} \circ \mathbf{X})\mathbf{C} + \mathbf{S}'(\mathbf{O} \circ (\mathbf{SC}'))\mathbf{C}] \tag{19}$$

From (17) to (18), we use the following lemma.

Lemma 1

$$Tr(\mathbf{O} \circ A)'(\mathbf{O} \circ A) = Tr(\mathbf{O}' \circ A')A = TrA'(\mathbf{O} \circ A) \tag{20}$$

In order to minimize J_C, we take the partial derivative of J_C with respect to S, which gives

$$\frac{\partial J_C}{\partial \mathbf{S}} = -2(\mathbf{O} \circ \mathbf{X})\mathbf{C} + 2(\mathbf{O} \circ (\mathbf{SC}'))\mathbf{C} \tag{21}$$

$$= -2(\mathbf{O} \circ (-\mathbf{X} + \mathbf{SC}'))\mathbf{C} \tag{22}$$

Setting $\frac{\partial J_C}{\partial \mathbf{S}} = 0$ gives $-\mathbf{X} + \mathbf{SC}' = 0$. Multiplying both sides by $\mathbf{C}$ and using the fact that $\mathbf{C}'\mathbf{C} = \mathbf{I}$ lead to $\mathbf{S} = \mathbf{XC}$. This gives

$$\tilde{\mathbf{S}} = \mathbf{S}\mathbf{diag}^{-1}(\sqrt{|C_1|}...\sqrt{|C_l|}) \tag{23}$$

$$= \mathbf{XC}\mathbf{diag}^{-1}(\sqrt{|C_1|}...\sqrt{|C_l|}) \tag{24}$$

$$\text{i.e., } \tilde{\mathbf{s}}_q = \frac{1}{|C_q|} \sum_{\mathbf{x}_j \in C_q} \mathbf{x}_j \tag{25}$$

Equation (25) shows that minimizing J_C indeed leads $\tilde{\mathbf{s}}_q$ to the centroid of cluster C_q. This also confirms that optimizing (10) is equivalent to performing K-means clustering on the columns of $\mathbf{X}$. Similarly, we can demonstrate that optimizing the following objective function is equivalent to performing K-mean clustering on the rows of an incomplete relational matrix $\mathbf{X}$.

$$\min_{\mathbf{R}\in\mathbb{R}_+^{m\times k}, \mathbf{R}'\mathbf{R}=\mathbf{I}} J_R = \|\mathbf{O}_{m\times n} \circ (\mathbf{X}_{m\times n} - \mathbf{R}_{m\times k}\mathbf{U}_{k\times n})\|^2 \tag{26}$$

where $\mathbf{U} = \mathbf{PC}'$. Therefore, solving a modified version of the objective function in (7) (i.e., with orthogonal constraint on $\mathbf{R}$ and $\mathbf{C}$) is equivalent to clustering the columns and rows of an incomplete relational matrix $\mathbf{X}$.

We now generalize result to the original objective function in (7). We define $\tilde{\mathbf{C}} = \mathbf{C}\mathbf{diag}(\sqrt{|C_1|}...\sqrt{|C_l|})$ and $\tilde{\mathbf{R}} = \mathbf{R}\mathbf{diag}(\sqrt{|R_1|}...\sqrt{|R_k|})$, where C_j and R_i denote the j-th and i-th column and row cluster, respectively. Accordingly, $\mathbf{P}$ will be updated to

$$\tilde{\mathbf{P}} = \mathbf{diag}^{-1}(\sqrt{|R_1|}...\sqrt{|R_k|})\mathbf{P}\mathbf{diag}^{-1}(\sqrt{|C_1|}...\sqrt{|C_l|})$$

to ensure that $\mathbf{RPC}' = \tilde{\mathbf{R}}\tilde{\mathbf{P}}\tilde{\mathbf{C}}'$. This gives $\sum_{q=1}^{l} \tilde{\mathbf{C}}(j, q) = 1$ and $\sum_{p=1}^{k} \tilde{\mathbf{R}}(i, p) = 1$ with each row of $\tilde{\mathbf{C}}$ and $\tilde{\mathbf{R}}$ only having one non-zero element, respectively. We now relax the constraint by allowing each row of $\tilde{\mathbf{C}}$ and $\tilde{\mathbf{R}}$ to have multiple non-zero elements. This leads to the original constraints in (7), i.e., $\tilde{\mathbf{C1}} = \mathbf{1}$ and $\tilde{\mathbf{R1}} = \mathbf{1}$. Hence, solving (7) is equivalent to simultaneously clustering the rows and columns of an incomplete relational matrix $\mathbf{X}$, where soft cluster membership is allowed. Entry $\tilde{\mathbf{C}}(j, q) \in \tilde{\mathbf{C}}$(or $\tilde{\mathbf{R}}(i, p) \in \tilde{\mathbf{R}}$) denotes the probability of column j (or row i) belongs to the q-th (or the p-th) column (or row) cluster.

3.3 QoS estimation

Besides the cluster indicator matrices $\mathbf{R}$ and $\mathbf{C}$, another key component of the model is the prototype matrix $\mathbf{P}$. An entry $\mathbf{P}(p, q) \in \mathbf{P}$ captures the interaction between the p-th row cluster and the q-th column cluster. As each row of the relational matrix $\mathbf{X}$ corresponds to a user and each column corresponds to a service, $\mathbf{P}$ captures the interactions between the user clusters and service clusters. Hence, the QoS that a user i will receive from an unknown service j can approximated by the convex combination of the cluster interaction matrix $\mathbf{P}$:

$$\mathbf{X}(i, j) = \sum_{p=1}^{k} \sum_{q=1}^{l} \mathbf{R}(i, p)\mathbf{P}(p, q)\mathbf{C}(j, q) \tag{27}$$

As the interaction between a user cluster and a service cluster is much more likely to occur than the interaction between individual users and services, the proposed model can more effectively hand the scarcity of the QoS data. Our experimental result clearly justify the effectiveness of the proposed model in QoS prediction especially when the QoS data is very scarce.

4 RCM construction

We present algorithms that efficiently solve the objective function given by (7). This will lead to the optimal cluster indicator matrices $\mathbf{R}$ and $\mathbf{C}$ as well as the cluster interaction matrix $\mathbf{P}$. These matrices will then be used to estimate the QoS of unknown services by following (27). In order to deal with the constraints, we convert them into two penalty terms and update (7) as:

$$\min_{\mathbf{R}\in\mathbb{R}_+^{m\times k}\mathbf{C}\in\mathbb{R}_+^{n\times l}} J(\mathbf{R}, \mathbf{P}, \mathbf{C})$$

$$\begin{aligned} J(\mathbf{R}, \mathbf{P}, \mathbf{C}) = &\left\|\mathbf{O}_{m\times n} \circ (\mathbf{X}_{m\times n} - \mathbf{R}_{m\times k}\mathbf{P}_{k\times l}\mathbf{C}'_{l\times n})\right\|^2 \\ &+ \lambda_R||\mathbf{R1} - \mathbf{1}||^2 + \lambda_C||\mathbf{C1} - \mathbf{1}||^2 \end{aligned} \tag{28}$$

where $\lambda_R,\ \lambda_C > 0$. As can be seen, when $\mathbf{P}$ and $\mathbf{C}$ are fixed, $J(\mathbf{R}, \mathbf{P}, \mathbf{C})$ is a convex function of $\mathbf{R}$. Similarly, when $\mathbf{P}$ and $\mathbf{R}$ are fixed, $J(\mathbf{R}, \mathbf{P}, \mathbf{C})$ is a convex on $\mathbf{C}$; when $\mathbf{R}$ and $\mathbf{C}$ are fixed, $J(\mathbf{R}, \mathbf{P}, \mathbf{C})$ is convex on $\mathbf{P}$. This property allows us to exploit an iterative algorithm that updates $\mathbf{R}, \mathbf{P}, \mathbf{C}$ in turn to optimize objective function (28). In each iteration, the three matrices $\mathbf{R}, \mathbf{P}, \mathbf{C}$ are updated in turn. When one matrix is updated, the other two are fixed.

A central component of the iterative algorithm is a set of auxiliary functions that play a key role for updating the matrices in each iteration. An auxiliary function forms the upper bound of the objective function based on the result obtained from the previous iteration. Hence, a matrix is updated by minimizing its corresponding auxiliary function. Auxiliary function was first introduced in [18]. In what follows, we give its definition and describe the features that are key to the update rules employed in our iterative algorithm.

Definition 2 $Z(\mathbf{S}, \mathbf{S}')$ is an auxiliary function of function $J(\mathbf{S})$ if it satisfies the following conditions for any $\mathbf{S}$ and $\mathbf{S}'$: $Z(\mathbf{S}, \mathbf{S}') \geq J(\mathbf{S})$; $Z(\mathbf{S}, \mathbf{S}) = J(\mathbf{S})$.

Lemma 2 *J is non-increasing under the following update rule if Z is an auxiliary function of J:*

$$\mathbf{X}^{(t+1)} = \arg\min_{\mathbf{X}} Z(\mathbf{X}, \mathbf{X}^{(t)}) \tag{29}$$

where $\mathbf{X}^{(t)}$ and $\mathbf{X}^{(t+1)}$ are matrix $\mathbf{X}$ at the t-th and $(t+1)$-th iterations, respectively.

Proof

$$J(\mathbf{X}^{(t)}) = Z(\mathbf{X}^{(t)}, \mathbf{X}^{(t)}) \geq Z(\mathbf{X}^{(t)}, \mathbf{X}^{(t+1)}) \geq J(\mathbf{X}^{(t+1)})$$

□

Lemma 3 *Given a function J($\mathbf{R}$), which only contains terms in $(\mathbf{R}, \mathbf{P}, \mathbf{C})$ that are relevant to $\mathbf{R}$:*

$$J(\mathbf{R}) = \left\| \mathbf{O}_{m\times n} \circ (\mathbf{X}_{m\times n} - \mathbf{R}_{m\times k}\mathbf{P}_{k\times l}\mathbf{C}'_{l\times n}) \right\|^2 + \lambda_R ||\mathbf{R1} - \mathbf{1}||^2 \tag{30}$$

The auxiliary function of $J(\mathbf{R})$ is given by

$$\begin{aligned} &Z(\mathbf{R}, \tilde{\mathbf{R}}) \\ &\quad = \sum_{(i,j)\in\mathcal{O}} \Bigg[\mathbf{X}^2(i,j) - 2\sum_{pq} \mathbf{X}(i,j)\tilde{\mathbf{R}}(i,p)\mathbf{P}(p,q)\mathbf{C}(j,q)\left(1 + \log\frac{\mathbf{R}(i,p)}{\tilde{\mathbf{R}}(i,p)}\right) \\ &\qquad\qquad + \sum_{pq} [\tilde{\mathbf{R}}\mathbf{P}\mathbf{C}'](i,j)\mathbf{P}(p,q)\mathbf{C}(j,q)\frac{\mathbf{R}^2(i,p)}{\tilde{\mathbf{R}}(i,p)} \Bigg] \qquad (31) \\ &\quad + \lambda_R \sum_{ip}\left([\tilde{\mathbf{R}}\mathbf{1}]_i \frac{\mathbf{R}^2(i,p)}{\tilde{\mathbf{R}}(i,p)} \right) - \lambda_R \sum_{ip} 2\tilde{\mathbf{R}}(i,p)\left(1 + \log\frac{\mathbf{R}(i,p)}{\tilde{\mathbf{R}}(i,p)}\right) + m\lambda_R \qquad (32) \end{aligned}$$

Proof We define

$$J(\mathbf{R})_1 = \left\| \mathbf{O}_{m\times n} \circ (\mathbf{X}_{m\times n} - \mathbf{R}_{m\times k}\mathbf{P}_{k\times l}\mathbf{C}'_{l\times n}) \right\|^2 \tag{33}$$

$$J(\mathbf{R})_2 = \lambda_R ||\mathbf{R1} - \mathbf{1}||^2 \tag{34}$$

We derive $Z(\mathbf{R}, \tilde{\mathbf{R}})$ by identifying the upper bounds of $J(\mathbf{R})_1$ and $J(\mathbf{R})_2$, respectively.

$$J(\mathbf{R})_1 = \sum_{(i,j)\in\mathcal{O}} \left(\mathbf{X}(i,j) - \sum_{pq} \mathbf{R}(i,p)\mathbf{P}(p,q)\mathbf{C}(j,q) \right)^2 \tag{35}$$

$$\leq \sum_{(i,j)\in\mathcal{O}} \left[\sum_{pq} \frac{\tilde{\mathbf{R}}(i,p)\mathbf{P}(p,q)\mathbf{C}(j,q)}{[\tilde{\mathbf{R}}\mathbf{P}\mathbf{C}'](i,j)} \times \left(\mathbf{X}(i,j) - \frac{[\tilde{\mathbf{R}}\mathbf{P}\mathbf{C}'](i,j)}{\tilde{\mathbf{R}}(i,p)\mathbf{P}(p,q)\mathbf{C}(j,q)} \mathbf{R}(i,p)\mathbf{P}(p,q)\mathbf{C}(j,q) \right)^2 \right] \tag{36}$$

$$= \sum_{(i,j)\in\mathcal{O}} \left(\mathbf{X}^2(i,j) - 2\sum_{pq} \mathbf{A}(i,j)\mathbf{R}(i,p)\mathbf{P}(p,q)\mathbf{C}(j,q) + \sum_{p,q} [\tilde{\mathbf{R}}\mathbf{P}\mathbf{C}'](i,j)\mathbf{P}(p,q)\mathbf{C}(j,q)\frac{\mathbf{R}(i,p)}{\tilde{\mathbf{R}}(i,p)} \right) \tag{37}$$

$$\leq \sum_{(i,j)\in\mathcal{O}} \left[\mathbf{X}^2(i,j) - 2\sum_{pq} \mathbf{X}(i,j)\tilde{\mathbf{R}}(i,p)\mathbf{P}(p,q)\mathbf{C}(j,q) \left(1 + \log\frac{\mathbf{R}(i,p)}{\tilde{\mathbf{R}}(i,p)}\right) + \sum_{pq} [\tilde{\mathbf{R}}\mathbf{P}\mathbf{C}'](i,j)\mathbf{P}(p,q)\mathbf{C}(j,q)\frac{\mathbf{R}^2(i,p)}{\tilde{\mathbf{R}}(i,p)} \right] \tag{38}$$

From (35) to (36), we use Jensen's inequality and the convexity of the quadratic function. From (37) to (38), we use inequality $x \geq 1 + \log x, \forall x > 0$.

$$J(\mathbf{R})_2 = \lambda_R \sum_{i=1}^{m} \left(\sum_{p=1}^{k} \mathbf{R}(i,p) - 1 \right)^2 \tag{39}$$

$$= \lambda_R \sum_{i=1}^{m} \left(\sum_{p=1}^{k} \frac{\tilde{\mathbf{R}}(i,p)}{[\tilde{\mathbf{R}}\mathbf{1}]_i} - \sum_{p=1}^{k} \frac{\tilde{\mathbf{R}}(i,p)}{[\tilde{\mathbf{R}}\mathbf{1}]_i} \frac{[\tilde{\mathbf{R}}\mathbf{1}]_i}{\tilde{\mathbf{R}}(i,p)} \mathbf{R}(i,p) \right)^2 \tag{40}$$

$$\leq \lambda_R \sum_{i=1}^{m} \sum_{p=1}^{k} \frac{\tilde{\mathbf{R}}(i,p)}{[\tilde{\mathbf{R}}\mathbf{1}]_i} \left(\frac{[\tilde{\mathbf{R}}\mathbf{1}]_i}{\tilde{\mathbf{U}}(i,p)} \mathbf{R}(i,p) - 1 \right)^2 \tag{41}$$

$$= \lambda_R \sum_{i=1}^{m} \sum_{p=1}^{k} \left(\frac{[\tilde{\mathbf{R}}\mathbf{1}]_i}{\tilde{\mathbf{R}}(i,p)} \mathbf{U}^2(i,p) - 2\tilde{\mathbf{R}}(i,p) \frac{\mathbf{R}(i,p)}{\tilde{\mathbf{R}}(i,p)} \right) + m\lambda_R \tag{42}$$

$$\le \lambda_R \sum_{ip} \left([\tilde{\mathbf{R}}\mathbf{1}]_i \frac{\mathbf{R}^2(i,p)}{\tilde{\mathbf{R}}(i,p)} \right)$$

$$- \lambda_R \sum_{ip} 2\tilde{\mathbf{R}}(i,p) \left(1 + \log \frac{\mathbf{R}(i,p)}{\tilde{\mathbf{R}}(i,p)} \right) + m\lambda_R \tag{43}$$

Similarly, we also use Jensen's inequality, the convexity of the quadratic function, and inequality $x \ge 1 + \log x, \forall x > 0$ in the above derivations.

Putting $J(\mathbf{R})_1$ and $J(\mathbf{R})_2$ together gives $Z(\mathbf{R}, \tilde{\mathbf{R}}) \ge J(\mathbf{R})$ and $Z(\tilde{\mathbf{R}}, \tilde{\mathbf{R}}) = J(\tilde{\mathbf{R}})$. Thus, $Z(\mathbf{R}, \tilde{\mathbf{R}})$ is an auxiliary function of $J(\mathbf{R})$. □

Based on the auxiliary function, the update strategy can be detailed as follows. Assume that $\tilde{\mathbf{R}}$ is the result obtained from the t-th iteration and we want to compute an updated $\mathbf{R}$ in the $(t+1)$-th iteration. We know that $Z(\mathbf{R}, \tilde{\mathbf{R}})$ forms the upper bound of $J(\mathbf{R})$ through the proof of Lemma 3. The idea is to compute $\tilde{\mathbf{R}}'$ that minimizes $Z(\mathbf{R}, \tilde{\mathbf{R}})$, i.e., $\tilde{\mathbf{R}}' = \arg\min_{\mathbf{R}} Z(\mathbf{R}, \tilde{\mathbf{R}})$. We then use $\tilde{\mathbf{R}}'$ to update $\mathbf{R}$.

Theorem 1 *The objective function $J(\mathbf{R})$ decreases monotonically under the following update rule:*

$$\mathbf{R}(i,p) = \tilde{\mathbf{R}}(i,p) \left[\frac{[\mathbf{O} \circ \mathbf{XCP}'](i,p) + \lambda_R}{[(\mathbf{O} \circ (\tilde{\mathbf{R}}\mathbf{PC}))\mathbf{CP}' + \lambda_R \tilde{\mathbf{R}}\mathbf{E}](i,p)} \right]^{\frac{1}{2}} \tag{44}$$

Proof In order to minimize $Z(\mathbf{R}, \tilde{\mathbf{R}})$, we take the derivative of $Z(\mathbf{R}, \tilde{\mathbf{R}})$ with respect to $\mathbf{R}(i,p)$, which gives

$$\frac{\partial Z(\mathbf{R}, \tilde{\mathbf{R}})}{\partial \mathbf{R}(i,p)} = \frac{\tilde{\mathbf{R}}(i,p)}{\mathbf{R}(i,p)} \left(\sum_{\substack{j \\ (i,j)\in\mathcal{O}}} \sum_{q} -2\mathbf{X}(i,j)\mathbf{P}(p,q)\mathbf{C}(j,q) \right)$$

$$+ \frac{\mathbf{R}(i,p)}{\tilde{\mathbf{R}}(i,p)} \left(\sum_{\substack{j \\ (i,j)\in\mathcal{O}}} \sum_{q} 2[\tilde{\mathbf{R}}\mathbf{PC}'](i,j)\mathbf{P}(p,q)\mathbf{C}(j,q) \right)$$

$$+ 2\lambda_R [\tilde{\mathbf{R}}\mathbf{E}](i,p) \frac{\mathbf{R}(i,p)}{\tilde{\mathbf{R}}(i,p)} - 2\lambda_R \frac{\tilde{\mathbf{R}}(i,p)}{\mathbf{R}(i,p)}$$

Set $\frac{\partial Z(\mathbf{R},\tilde{\mathbf{R}})}{\partial \mathbf{R}(i,p)} = 0$ and formulate the above function in matrix form, we get

$$\begin{aligned} 0 = & -2[\mathbf{O} \circ \mathbf{XCP}'](i,p)\frac{\tilde{\mathbf{R}}(i,p)}{\mathbf{R}(i,p)} \\ & + 2[\mathbf{O} \circ (\tilde{\mathbf{R}}\mathbf{PC}')\mathbf{P}(p,q)\mathbf{C}(j,q)](i,p)\frac{\mathbf{R}(i,p)}{\tilde{\mathbf{R}}(i,p)} \\ & + 2\lambda_R[\tilde{\mathbf{R}}\mathbf{E}](i,p)\frac{\mathbf{R}(i,p)}{\tilde{\mathbf{R}}(i,p)} - 2\lambda_R\frac{\tilde{\mathbf{R}}(i,p)}{\mathbf{R}(i,p)} \end{aligned}$$

Solving the above equation leads to the update rule given in (44). Furthermore, if we take the second order derivative of $Z(\mathbf{R},\tilde{\mathbf{R}})$ with respect to $\mathbf{R}(i,p)$, we get the Hessian matrix for $Z(\mathbf{R},\tilde{\mathbf{R}})$:

$$\begin{aligned} \frac{\partial^2 Z(\mathbf{R},\tilde{\mathbf{R}})}{\partial \mathbf{R}(i,p)\partial \mathbf{R}(x,y)} = \delta_{ix}\delta_{py}\Bigg(& \frac{2[\mathbf{O} \circ \mathbf{XCP}'](i,p)\tilde{\mathbf{R}}(i,p) + 2\lambda_R\tilde{\mathbf{R}}(i,p)}{\mathbf{R}^2(i,p)} \\ & + \frac{2[\mathbf{O} \circ (\tilde{\mathbf{R}}\mathbf{PC}')\mathbf{P}(p,q)\mathbf{C}(j,q)](i,p) + 2\lambda_R[\tilde{\mathbf{R}}\mathbf{E}](i,p)}{\tilde{\mathbf{R}}(i,p)}\Bigg) \end{aligned}$$

where $\delta_{ij} = 1$ when $i = j$ and 0 otherwise. It is straightforward to verify that the Hessian matrix is a diagonal matrix with positive diagonal elements. Since the Hessian is positive definite, the above derivation guarantees to converge to the global minimum of $Z(\mathbf{R},\tilde{\mathbf{R}})$. In another word, we indeed update $\mathbf{R}$, which minimizes $Z(\mathbf{R},\tilde{\mathbf{R}})$. □

Following the same rationale, we can prove the following two theorems.

Theorem 2 *Given a function J(*$\mathbf{C}$*), which only contains terms in* $(\mathbf{R},\mathbf{P},\mathbf{C})$ *that are relevant to* $\mathbf{C}$*:*

$$J(\mathbf{C}) = \left\|\mathbf{O}_{m\times n} \circ (\mathbf{X}_{m\times n} - \mathbf{R}_{m\times k}\mathbf{P}_{k\times l}\mathbf{C}'_{l\times n})\right\|^2 + \lambda_C||\mathbf{C1} - \mathbf{1}||^2 \tag{45}$$

The objective function $J(\mathbf{C})$ *decreases monotonically under the following update rule:*

$$\mathbf{C}(j,q) = \tilde{\mathbf{C}}(j,q)\left[\frac{[(\mathbf{O} \circ \mathbf{X})'\mathbf{RP}](j,q) + \lambda_C}{[(\mathbf{O} \circ (\mathbf{RP}\tilde{\mathbf{C}}))'\mathbf{RP} + \lambda_C\tilde{\mathbf{C}}\mathbf{E}](j,q)}\right]^{\frac{1}{2}} \tag{46}$$

Theorem 3 *Given a function J(*$\mathbf{P}$*), which only contains terms in* $(\mathbf{R},\mathbf{P},\mathbf{C})$ *that are relevant to* $\mathbf{P}$*:*

$$J(\mathbf{C}) = \left\|\mathbf{O}_{m\times n} \circ (\mathbf{X}_{m\times n} - \mathbf{R}_{m\times k}\mathbf{P}_{k\times l}\mathbf{C}'_{l\times n})\right\|^2 \tag{47}$$

The objective function $J(\mathbf{P})$ *decreases monotonically under the following update rule:*

$$\mathbf{P}(p,q) = \tilde{\mathbf{P}}(p,q)\left[\frac{[\mathbf{R}'(\mathbf{O} \circ \mathbf{X})\mathbf{C}](p,q)}{[\mathbf{R}'(\mathbf{O} \circ (\mathbf{R}\tilde{\mathbf{P}}\mathbf{C}'))\mathbf{C}](p,q)}\right]^{\frac{1}{2}} \tag{48}$$

Built upon the update rules derived above, Algorithm 1 gives the details of constructing the **RCM** model.

Algorithm 1 RCM construction.

Input: Historical QoS data formated as a relational matrix **X**.
Output: User and service cluster indicator matrices **R** and **C**; Cluster interaction matrix **P**.

```
1: Initialize matrices R, C, P, and MAXITER.
2: for all $i \in [1, \text{MAXITER}]$ do
3:     $\mathbf{P}(p,q) = \mathbf{P}(p,q)\left[\frac{[\mathbf{R}'(\mathbf{O}\circ\mathbf{X})\mathbf{C}](p,q)}{[\mathbf{R}'(\mathbf{O}\circ(\mathbf{RPC}'))\mathbf{C}](p,q)}\right]^{\frac{1}{2}}$
4:     $\mathbf{R}(i,p) = \mathbf{R}(i,p)\left[\frac{[\mathbf{O}\circ\mathbf{XCP}'](i,p)+\lambda_R}{[(\mathbf{O}\circ(\mathbf{RPC}))\mathbf{CP}'+\lambda_R\mathbf{RE}](i,p)}\right]^{\frac{1}{2}}$
5:     $\mathbf{C}(j,q) = \mathbf{C}(j,q)\left[\frac{[(\mathbf{O}\circ\mathbf{X})'\mathbf{RP}](j,q)+\lambda_C}{[(\mathbf{O}\circ(\mathbf{RP}\tilde{\mathbf{C}}))'\mathbf{RP}+\lambda_C\mathbf{CE}](j,q)}\right]^{\frac{1}{2}}$
6: end for
```

5 System architecture and case study

We present the system architecture that supports the RCM based service selection. We will also use a small scale case study to illustrate how the proposed RCM model helps achieve personalized service selection.

5.1 System architecture

Figure 1 shows the system architecture. The key components and how they interact with each other are elaborated as follows:

- **Service Discovery**: This component accepts users' service requests and identifies a set of services, all of which satisfy users' functional requirements. Existing service discovery approaches that exploit information retrieval techniques and

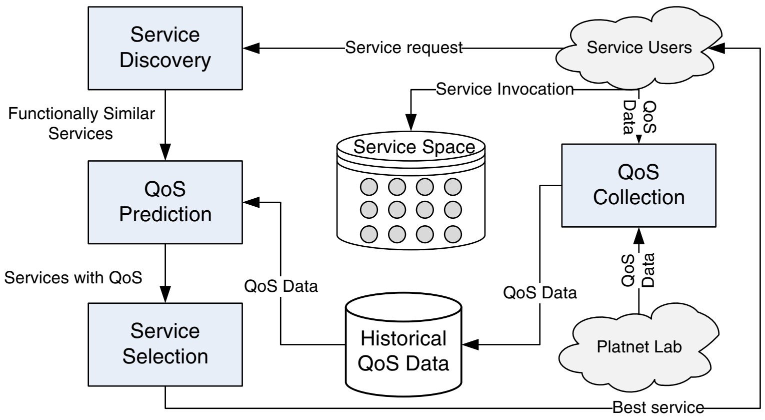

Figure 1 System architecture of RCM based service selection.

semantic Web technologies can be used to locate services with matching functionalities.

- **QoS Prediction**: This component exploits the proposed RCM model to accurately predict the QoS of a priori unknown service providers that compete to offer user desired functionalities.
- **Service Selection**: This component selects the service with the best user desired quality based on the QoS predicted by the RCM model. Existing efforts on QoS aware service selection can be exploited to perform the selection and return the best service to the users.
- **QoS Collection**: This component collects historical QoS information from service users or software clients deployed on a distributed platform, such as Platnet Lab. The QoS information is exploited by the proposed RCM model to make QoS predictions.

The historical QoS data can be collected via two complementary strategies. The first strategy is to deploy a set of software clients on distributed platforms, such as PlanetLab, to simulate actual service users. These software clients automatically invoke services and return the QoS data obtained from the invocations. Similar approaches have been adopted in existing systems, such as *WSRec* in [40]. The second strategy requires user participation. The feedback collection mechanisms used by existing recommendation systems, such as Amazon and Netflix, will shed light on QoS collection from actual service users. Furthermore, privacy-preserving collaborative filtering techniques, such as randomized perturbation [21], can be leveraged to ensure that the private information of service users is protected while sharing their QoS data.

5.2 Case study

In what follows, we use a QoS data sample obtained from a real-world QoS collection [40] to perform a small scale case study. The purpose is to demonstrate the effectiveness of the proposed RCM model and how it can help achieve personalized service selection. It is worth to note that the prediction accuracy and the scalability of the RCM model will be formally evaluated via an extensive empirical study in Section 6.

Consider a QoS dataset that records the response times (in seconds) of eight users on six Web services. Since each user may have only invoked a limited set of services, some QoS entries are missing. Based on the proposed RCM model, the QoS data is denoted by a matrix, which is given by

$$\mathbf{X} = \begin{pmatrix} 0.1432 & 1.4335 & 0.4225 & 0.7802 & 0.0705 & \mathit{null} \\ 0.1322 & 1.4155 & 0.4313 & 0.7624 & 0.0605 & 2.3237 \\ 0.1406 & 1.4672 & \mathit{null} & 0.8169 & 0.0658 & 2.2844 \\ \mathit{null} & 1.4288 & 0.4565 & 0.7841 & 0.0636 & 2.1715 \\ 0.6033 & 2.1097 & 0.7734 & 0.9246 & 0.6292 & 2.3656 \\ 0.6946 & 0.1922 & 0.3856 & 0.6215 & 0.6125 & 2.1733 \\ \mathit{null} & 0.1590 & \mathit{null} & 0.5662 & 0.6707 & \mathit{null} \\ 0.7114 & 0.1568 & \mathit{null} & 0.6071 & 0.6480 & 2.1382 \end{pmatrix}$$

where each row denotes a user, each column denotes a service, *null* signifies missing values. Consider that a user u_7 (i.e., the 7-th row in $\mathbf{X}$) wants to request a service with the fastest response time. There are three candidate services, corresponding to columns 1, 3 and 6 in matrix $\mathbf{X}$, all of which provide the functionality required by the user. Manually selecting a service would require the user to study the APIs of all the candidate services, develop customized software client for each candidate, and then make the service calls. In practice, the number of candidate services is in a much larger scale, which makes a manual process as above prohibitively expensive.

The proposed RCM model automatically predicts the QoS of previously unknown services. Applying RCM on $\mathbf{X}$, we obtain three low-rank matrices:

$$R = \begin{pmatrix} 0.4536 & 0.1006 \\ 0.4542 & 0.2081 \\ 0.4732 & 0.1952 \\ 0.4457 & 0.1967 \\ 0.5879 & 0.2116 \\ 0.0000 & 0.5332 \\ 0.0000 & 0.5467 \\ 0.0000 & 0.5279 \end{pmatrix}, \quad P = \begin{pmatrix} 0.499 & 25.181 \\ 17.464 & 2.318 \end{pmatrix}, \quad C = \begin{pmatrix} 0.0762 & 0.0000 \\ 0.0008 & 0.1255 \\ 0.0362 & 0.0324 \\ 0.0569 & 0.0489 \\ 0.0661 & 0.0000 \\ 0.2191 & 0.1123 \end{pmatrix}$$

Plugging $\mathbf{R}$, $\mathbf{P}$, and $\mathbf{C}$ into (27), RCM provides the predictions of the missing QoS entires in $\mathbf{X}$. In particular, the response times for the three candidate services are estimated as: 0.7273, 0.3869, and 2.2342 s, respectively. These are pretty close to the actual response times, which are 0.6524, 0.3516, and 2.2144 s, respectively. More importantly, the estimations preserve the relative order of the actual response times, which is critical for service selection. It is also worth to note that RCM discovers the cluster structures of users and services. For example, the first five users belong to the first user cluster and the last three users belong to the second user cluster. This is because in the first five rows of $\mathbf{R}$, the first column is larger than the second column. While in the last three rows of $\mathbf{R}$, the second column is larger than the first column.

Finally, the estimated QoS is then sent to the service selection component as shown in the system architecture (see Figure 1), which will return the best candidate service to the user. In this way, the proposed RCM model plays a central role in turning a tedious and time consuming manual process into a fully automatic procedure that achieves accurate service selection decisions.

6 Empirical study

In this section, we present our empirical study that evaluates the proposed **RCM** for Web services QoS evaluation. We use both real-world and synthetic QoS data to assess its effectiveness and efficiency. We implement the classical collaborative filtering based approaches as described in Section 2 and apply them to the QoS datasets. We also compare our model with recently developed service recommendation approaches.

6.1 QoS datasets

Real QoS dataset We use the real-world QoS dataset created by *WSRec* [40]. In *WSRec*, a list of 21,197 publicly available Web services are obtained by crawling leading service provider websites, portal websites that publish publicly available Web services, and service search engines. The actual working services are less due to authentication restriction, permanent invocation failure, or extremely long processing duration. One hundred Web services are randomly selected from the working services. The providers of these selected services are across over 20 counties. The Planet-Lab wide area network has been exploited for QoS data collection. 150 computer nodes distributed in more than 20 countries are used to invoke the selected Web services. Over 1.5 millions Web service invocations are executed and the test results are collected. The averaged results for two QoS parameters are made available: Round-Trip Time (RTT) and failure rate. RTT records the time period between sending a request and receiving a response whereas failure-rate presents the probability that a request is correctly responded within certain amount of time. Limited by space, we focus on evaluating the RTT for unknown Web services in our experiments. The exactly same rationale can be applied to the evaluation of failure rate. Table 3 shows some sample data from the real QoS dataset. The first column gives the ip addresses of the compute nodes of Planet-Lab. The first row gives the ids of the randomly selected services. The sample data clearly demonstrates that the RTTs from the same provider may vary dramatically over different users (i.e., compute nodes).

Synthetic QoS dataset In addition to the real QoS data, we also generate some larger sized random data matrices in order to evaluate the efficiency and scalability of the proposed approach.

6.2 Experiment settings

Data preprocessing The averaged RTT records obtained from the invocation results of 150 computers on 100 services form a 150×100 relational matrix $\mathbf{X}$. Since the computers are used to simulate the users, each row in $\mathbf{X}$ corresponds to a user and each column corresponds to a Web service. As a real-world QoS data may be very sparse, we vary the sparsity of the relational matrix $\mathbf{X}$ from 80 % to 96 % by randomly removing 80 % to 96 % RTT entries from the matrix. We apply the **RCM** model to estimate the removed RTT entries and then compare the estimated value with the true value to evaluate the prediction accuracy. To avoid that the clustering

Table 3 Sample RTT records (ms).

Service ID	1	2	4	650	1015
12.108.127.136	4755.7	5875.6	4647.7	4779.8	4642.2
128.10.19.52	348.42	1010.2	278.5	574.4	280.6
128.112.139.80	1106.2	1482.7	579.5	2307.3	1006.7
128.113.226.235	293.9	1487.3	333.5	715.4	235.1
128.119.247.210	331.1	1219.6	297.9	601.0	215.8
128.135.11.149	349.2	1064.4	252.3	572.2	295.0
128.138.207.45	688.8	1255.7	415.6	635.2	457.5

result is dominated by columns or rows with very large values, we perform a column normalization on **X** before applying **RCM** on it.

Effectiveness evaluation metric We use Mean Absolute Error (MAE) for model effectiveness evaluation. MAE is a commonly exploited to test the quality of collaborative filtering algorithms:

$$MAE = \sum_{(i,j)\in\mathcal{R}} \frac{|\mathbf{X}(i,j) - \sum_{p=1}^{k}\sum_{q=1}^{l} \mathbf{R}(i,p)\mathbf{P}(p,q)\mathbf{C}(j,q)|}{|\mathcal{R}|} \tag{49}$$

where $\mathcal{R}$ denotes the set of removed RTT entires. Since the RTT entries are randomly removed, we run our algorithm 10 times and the average MAE is reported. Due to the sparsity of the QoS dataset, an algorithm may fail to provide any estimation. This is commonly referred to as the *cold start* problem. Therefore, as the second metric, we record the number of RTT entries that an algorithm fails to make an estimation. This signifies the capacity of how an algorithm handles a very sparse QoS dataset, which is quite common in practice.

Algorithm comparison We implement the classical collaborative filtering algorithms as presented in Section 2 and apply them to the QoS datasets. We include both Pearson Correlation Coefficient (referred to as PCC) and cosine similarity (referred to as COS) as similarity measures. For PCC based similarity measure, we implement both the user (referred to as u-PCC) and item (referred to as i-PCC) based approaches. For COS based similarity measure, we implement the user (referred to as u-COS) based approach. We also implement and compare our model with the algorithm used in *WSRec*, which is a hybrid collaborative algorithm that combines both user and item based approaches using their prediction accuracy as the aggregation weights [40]. The key algorithms and their descriptions are summarized in Table 4.

Parameter setting The objective function given in (28) shows that there are four important parameters in the proposed **CRM** model: numbers of user and service clusters (i.e., k and l), and two penalty terms (i.e., λ_R and λ_C) that are used to enforce the convex encoding constraint. To simply the selection of parameters, we set $k = l$ and $\lambda_R = \lambda_C$. This will reduce the total number of parameters to 2. The hybrid algorithm in *WSRec* uses a parameter λ to balance and combine results from user based and item based collaborative filtering. We set λ to 0.1, which is the same value used in the performance study of [40].

Table 4 Algorithm notations and descriptions.

Algorithm	Description
u-PCC	User based collaborative filtering algorithm
i-PCC	Item based collaborative filtering algorithm
u-COS	Cosine similarity based collaborative filtering algorithm
WSRec	Hybrid collaborative filtering algorithm used in *WSRec* [40]
RCM	Relational clustering based collaborative filtering scheme

Table 5 Effectiveness evaluation on the real QoS dataset.

Evaluation metric	MAE					Number of no predictions				
Data sparcity	80 %	84 %	88 %	92 %	96 %	80 %	84 %	88 %	92 %	96 %
u-PCC	0.0461	0.0521	0.0597	0.0752	0.0907	9.9	49.6	69.6	121	1441.1
i-PCC	0.0387	0.0410	0.0454	0.0522	0.0633	0	10	30	91.5	436
u-COS	0.0446	0.0484	0.0522	0.0599	0.0728	0	10	30	91.5	407
WSRec	0.0378	0.0399	0.0438	0.0504	0.0620	0	10	30	91.5	409.1
RCM	0.0379	0.0401	0.0429	0.0474	0.0478	0	0	0	0	0

6.3 Effectiveness evaluation

We evaluate the effectiveness of **RCM** by comparing it with other algorithms using the two metrics stated above. For **RCM**, we set $k = l = 28$ and $\lambda_R = \lambda_C = 10$. Table 5 summarizes the comparison results. For a sparsity ratio no greater than 84 %, the MAE performances of *WSRec* and **RCM** are almost the same. They both outperform all other algorithms. With the increase of the sparsity ratio, **RCM** significantly outperforms all other algorithms. For example, when the sparsity ratio is 96 %, the MAE of the second best algorithm (i.e., *WSRec*) is more than 20 % worse than that of **RCM**. The number of no predictions reflects how the algorithms handle the cold start issue, which arises when the dataset is very sparse. **RCM** obviously outperforms all other algorithms on this metric by successfully generating all predictions even for very sparse datasets. In contrast, all other algorithms fail to make some predictions when the sparsity ratio exceeds 84 %. All of them miss over 400 predictions when the sparsity ratio is no less than 96 %.

We also investigate how the parameters of **RCM** affect its effectiveness. The left chart of Figure 2 shows the effect of λ_R and λ_C. A key observation is that for less sparse data, more strictly enforcing the convex encoding is beneficial. In contrast, increasing the values of λ_R and λ_C may negatively affect the MAE performance for very sparse data. This is because precise cluster structures can be discovered from a less sparse data due to the presence of sufficient information. Convex encoding helps improve accuracy of the cluster membership assignment (see Section 3 for detail). Hence, the MAE performance can be improved accordingly. In contrast, the

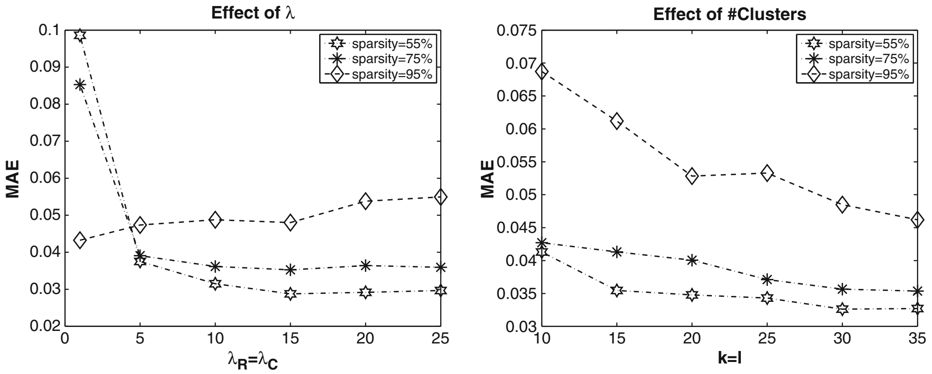

Figure 2 Effect of **RCM** parameters.

cluster structures may not be precise due to the lack of information for very sparse datasets. Strictly enforcing the convex encoding constraint may lead to imprecise cluster structures, which will decrease prediction accuracy.

The right chart of Figure 2 demonstrates how the MAE performance changes with the number of clusters. For very sparse QoS data, increasing the number of clusters benefits the MAE performance. With the increase of number of clusters, a large number of small clusters are formed, where each cluster only consists of highly similar users or services (based on the limited available information). This will be helpful to reduce the prediction error. On the other hand, for less sparse data where more information is present, it is possible to precisely discover cluster structures. In this case, relatively larger clusters that include all relevant user and service information help improve the prediction accuracy. For instance, with a sparsity ratio at 55 %, the best MAE performance is reached when the number of clusters is 30. The performance begins to decrease when further increasing the number of clusters.

6.4 Efficiency evaluation

We evaluate the efficiency of **RCM** by comparing the CPU time with other algorithms on both the real QoS data and the synthetic QoS data. The left chart in Figure 3 shows the total CPU time spent for predicting all the missing RTT entries. The efficiency of **RCM** is almost invariant with the number of predictions. Once the model is constructed, the predictions can be computed instantly. Algorithm 1 efficiently constructs the **RCM** model from the real QoS data and makes all the predictions by just using around 1 second. On the other hand, in all other algorithms, for each test case, they need to first identify the similar users or services from a large user or service space, and then exploits their QoS data to make the prediction. As the sparsity ratio decreases, the number of missing entries increases accordingly. Hence, the time spent for making all the predictions also increases.

We use the right chart to investigate how the algorithms scale with the size of the relational matrix **X**. We construct a set of random relational matrices with the number of entries varying from 10^4 to 10^6. We set the sparsity ratio at 90 % for all the datasets. Since predicting all missing entries is computational very expensive

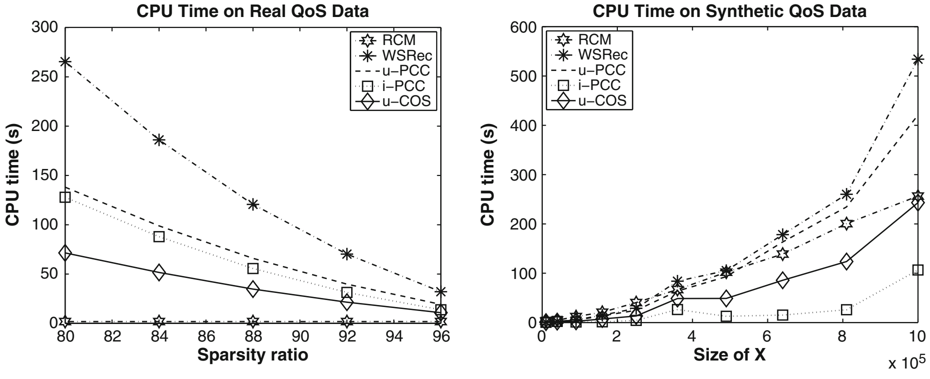

Figure 3 CPU time.

for all other algorithms, we report the average time used for making predictions for a single user on all services. For **RCM**, we report total time used for model construction and making predictions for all users. As can be seen, the CPU time used by **RCM** increases linearly with the size of **X**. The time to construct **RCM** and use it to make all the predictions is even less than the time used to make predictions for a single user by *WSRec* and u-PCC. When the size of **X** becomes very large, we can train the **RCM** model offline and then use it for efficient online prediction.

The time complexity of Algorithm 1 can be briefly analyzed as follows. The three update rules used by the algorithm require matrix product. Based on the sizes of the matrices involved, the time complexity of each update rule is bounded by $O(mn(k+l))$. Assume that the algorithm needs r iterations to converge. The overall complexity will be bounded by $O(mn(k+l)r)$.

7 Related work

Due to the ever increasing number of Web services, service selection techniques have drawn considerable attention recently. Different from service search engines (such as webservices.seekda.com) that retrieve services based on their functionality, service selection provides a way to locate services with the best user desired quality from a large number of functionally similar candidate services. Two pioneer efforts devoted for quality based service selection are [34, 36]. In [36], a QoS model is presented that includes a set of key quality parameters, such as latency, availability, reliability, reputaion and so on. A linear programming based approach is exploited to efficiently select the composite service with the best overall quality. The service selection problem is tackled in [34] by using a combinatorial model and a graph model. Efficient algorithms are then designed to select the services with the best quality. Other optimization strategies, such as genetic algorithms [6, 33], have also been explored towards selecting the global optimal services. When multiple QoS parameters are involved in the selection process, a weighting mechanism is typically exploited that integrates different QoS values into an objective function. Users have to convert their preferences over different QoS aspects into numeric weights, which forms a rather demanding task. Multi-Criteria Decision Making (or MCDM) has been leveraged to attack this challenge [9, 17, 26]. Skyline analysis techniques, which have been intensively investigated in database community, offer another effective means to tackle multi-criteria service selection [2, 30–32].

Recent research discovered that different users may receive significantly distinct QoS from the same service provider due to the disparity in their locations, network conditions, development tools, and so on [25, 40]. This demands a personalized approach in service selection. In personalized service selection, a service should be selected based on the specific situation of a given user. The collaborative filtering based techniques provide a promising solution to achieve personalized service selection. Collaborative filtering is the backbone technology for most recommendation systems nowadays [5, 7, 12, 14, 15]. It exploits the similarity between users' experiences to predict user preference on unknown items.

Several proposals have been developed that aim to exploit collaborative filtering to achieve personalized service selection [16, 25, 25, 39, 40, 40]. A user-based collaborative filtering approach is presented in [25] to predict the QoS of unknown service providers for an active user. The predicted QoS values can then be used to provide personalized service selection for the user. The QoS data is manually collected from 28 volunteer users using 63 proxy servers located in different parts of the world. *WSRec* improves [25] from two key aspects [40]. First, it proposes a hybrid collaborative filtering approach that integrates both the user and item based approaches to achieve better prediction accuracy. Second, it automates the QoS data collection process by exploiting the Planet-Lab network. Jiang et. al further differentiate user sensitive and user insensitive services and assign higher weights to the sensitive ones when determining the similarity between users [16]. A user insensitive service is the one invoked by a large number of users whereas a sensitive service is the one that has limited number of users. Another user based approach is presented in [8], which is augmented by a region model that integrates users' geographical information to improve accuracy. In addition to the historical QoS data, other types of user feedbacks are also used for personalized service selection, including invocation frequency [22], user assigned ratings [1, 27], and query histories [39].

All existing efforts as discussed above fall into the *memory based* collaborative filtering scheme. Take the user based approach as an example. The underlying idea is to identify similar users with the active user and use their QoS experience to make the prediction. As users have to invoke a number of common Web services in order to be considered as similar, the memory based scheme suffers the cold start problem. Our empirical study showed that memory based collaborative filtering may fail to make predictions when the QoS data becomes sparse. In contrast, the proposed **RCM** approach falls into the model based scheme. It more effectively deals with the cold start problem by building a global prototype matrix and predictions are made via a convex combination of the prototype matrix. Some other collaborative filtering models have also been developed, including latent factor models [7], Bayesian models [38], aspect models [15], and so on. However, most of these approaches suffer a high computational cost. In contrast, the efficient update rules exploited by **RCM** guaranttee its good performance.

8 Conclusion and future work

We present **RCM**, a collaborative filtering based scheme, for evaluating the QoS of a priori unknown service providers. The proposed **RCM** advances the current service computing research by enabling a model based approach for personalized service selection. We exploit and extend the relational clustering model that leverages the user-service interaction information to simultaneously construct a set of user and service clusters. A prototype matrix is also obtained that captures the user-service cluster interaction information. The convex constraint enables to predict the QoS of an unknown service though a convex combination of the prototype matrix. Our empirical study justifies the effectiveness of **RCM** on two important evaluation metrics: MAE and the ability to handle the cold start problem. The experiment results on the larger scale synthetic data show that **RCM** can be quickly constructed through a set of efficient update rules.

An important future direction is to investigate how to accommodate the changes of the QoS data, which we expect to be common and frequent. The aim is to devise an efficient strategy to update the model instead of completely reconstructing the model whenever a change occurs.

References

1. Averbakh, A., Krause, D., Skoutas, D.: Recommend me a service: personalized semantic web service matchmaking. In: 17th Workshop on Adaptivity and User Modeling in Interactive Systems (2009)
2. Benouaret, K., Benslimane, D., Hadjali, A.: On the use of fuzzy dominance for computing service skyline based on QoS. In: IEEE International Conference on Web Services, pp. 540–547 (2011)
3. Bianchini, D., Antonellis, V.D., Melchiori, M.: Flexible semantic-based service matchmaking and discovery. World Wide Web **11**(2), 227–251 (2008)
4. Binding Point: http://www.bindingpoint.com/. Accessed 4 Sept 2012
5. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: UAI '98: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, pp. 43–52. Morgan Kaufmann, San Mateo (1998)
6. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for qos-aware service composition based on genetic algorithms. In: GECCO '05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 1069–1075. ACM, New York (2005)
7. Canny, J.: Collaborative filtering with privacy via factor analysis. In: SIGIR '02: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 238–245. ACM, New York (2002)
8. Chen, X., Liu, X., Huang, Z., Sun, H.: Regionknn: a scalable hybrid collaborative filtering algorithm for personalized web service recommendation. In: ICWS, pp. 9–16 (2010)
9. Cheng, D.-Y., Chao, K.-M., Lo, C.-C., Tsai, C.-F.: A user centric service-oriented modeling approach. World Wide Web **14**(4), 431–459 (2011)
10. Ding, C., Li, T., Peng, W., Park, H.: Orthogonal nonnegative matrix t-factorizations for clustering. In: KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 126–135. ACM, New York (2006)
11. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity search for web services. In: VLDB '04: Proceedings of the Thirtieth International Conference on Very Large Data Bases, pp. 372–383. VLDB Endowment (2004)
12. Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Commun. ACM **35**(12), 61–70 (1992)
13. Grand Central: http://www.grandcentral.com/directory/. Accessed 5 July 2010
14. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 230–237. ACM, New York (1999)
15. Hofmann, T.: Latent semantic models for collaborative filtering. ACM Trans. Inf. Sys. **22**(1), 89–115 (2004)
16. Jiang, Y., Liu, J., Tang, M., Liu, X.F.: An effective web service recommendation method based on personalized collaborative filtering. In: ICWS, pp. 211–218 (2011)
17. Lamparter, S., Ankolekar, A., Studer, R., Grimm, S.: Preference-based selection of highly configurable web services. In: Proceedings of the 16th International Conference on World Wide Web, WWW '07, pp. 1013–1022. ACM, New York (2007)
18. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature **401**, 788–791 (1999)
19. Liu, X., Huang, G., Mei, H.: Discovering homogeneous web service community in the user-centric web environment. IEEE Transactions on Services Computing (TSC) **2**(2), 167–181 (2009)
20. OWL-S: http://www.daml.org/services/owl-s/ (2004). Accessed 4 Sept 2012
21. Polat, H., Du, W.: Privacy-preserving collaborative filtering using randomized perturbation techniques. In: Proceedings of the Third IEEE International Conference on Data Mining, ICDM '03, p. 625. IEEE Computer Society, Washington, DC (2003)

22. Rong, W., Liu, K., Liang, L.: Personalized web service ranking via user group combining association rule. IEEE International Conference on Web Services, pp. 445–452 (2009)
23. Salcentral: http://www.salcentral.com/. Accessed 5 July 2010
24. Schmidt, C., Parashar, M.: A peer-to-peer approach to web service discovery. World Wide Web **7**(2), 211–229 (2004)
25. Shao, L., Zhang, J., Wei, Y., Zhao, J., Xie, B., Mei H.: Personalized QoS prediction for web services via collaborative filtering. In: ICWS, pp. 439–446 (2007)
26. Tran, V.X., Tsuji, H., Masuda, R.: A new QoS ontology and its QoS-based ranking algorithm for web services. Simulation Modelling Practice and Theory **17**(8), 1378–1398 (2009)
27. Wang, H.-C., Lee, C.-S., Ho, T.-H.: Combining subjective and objective QoS factors for personalized web service selection. Expert Syst. Appl. **32**(2), 571–584 (2007)
28. Web Service List: http://www.webservicelist.com/. Accessed 4 Sept 2012
29. WSMO: http://www.wsmo.org/ (2004). Accessed 4 Sept 2012
30. Yu, Q., Bouguettaya, A.: Computing service skyline from uncertain QoWS. IEEE Transactions on Services Computing (TSC) **3**(1), 16–29 (2010)
31. Yu, Q., Bouguettaya, A.: Computing service skylines over sets of services. In: ICWS, pp. 481–488 (2010)
32. Yu, Q., Bouguettaya, A.: Multi-attribute optimization in service selection. World Wide Web **15**(1), 1–31 (2012)
33. Yu, Q., Rege, M., Bouguettaya, A., Medjahed, B., Ouzzani, M.: A two-phase framework for quality-awareweb service selection. Service Oriented Computing and Applications (SOCA) **4**(2), 63–79 (2010)
34. Yu, T., Lin, K.: Service selection algorithms for composing complex services with multiple QoS constraints. In: ICSOC'05 (2005)
35. Yu, T., Zhang, Y., Lin, K.-J.: Efficient algorithms for web services selection with end-to-end QoS constraints. ACM Trans. Web **1**(1), 6 (2007)
36. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.: Quality-driven web service composition. In: Proc. of 14th International Conference on World Wide Web (WWW'03), Budapest, Hungary. ACM Press (2003)
37. Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Trans. Softw. Eng. **30**(5), 311–327 (2004)
38. Zhang, Y., Koren, J.: Efficient bayesian hierarchical user modeling for recommendation system. In: SIGIR '07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 47–54. ACM, New York (2007)
39. Zhang, Q., Ding, C., Chi, C.-H.: Collaborative filtering based service ranking using invocation histories. In: ICWS, pp. 195–202 (2011)
40. Zheng, Z., Ma, H., Lyu, M.R., King, I.: Wsrec: A collaborative filtering based web service recommender system. In: ICWS, pp. 437–444 (2009)