Multi-Dimensional QoS Prediction for Service Recommendations

Shangguang Wang, *Senior, Member, IEEE*, You Ma, Bo Cheng and Fangchun Yang, *Senior Member, IEEE, Rong N. Chang*, *Senior Member, IEEE*

Abstract—Advances in mobile Internet technology have enabled the clients of Web services to be able to keep their service sessions alive while they are on the move. Since the services consumed by a mobile client may be different over time due to client location changes, a multi-dimensional spatiotemporal model is necessary for analyzing the service consumption relations. Moreover, competitive Web service recommenders for the mobile clients must be able to predict unknown quality-of-service (QoS) values well by taking into account the target client's service requesting time and location, e.g., performing the prediction via a set of multi-dimensional QoS measures. Most contemporary QoS prediction methods exploit the QoS characteristics for one specific dimension, e.g., time or location, and do not exploit the structural relationships among the multi-dimensional QoS data. This paper proposes an integrated QoS prediction approach which unifies the modeling of multi-dimensional QoS data via multi-linear-algebra based concepts of tensor and enables efficient Web service recommendation for mobile clients via tensor decomposition and reconstruction optimization algorithms. In light of the unavailability of measured multi-dimensional QoS datasets in the public domain, this paper also presents a transformational approach to creating a credible multi-dimensional QoS dataset from a measured taxi usage dataset which contains high dimensional time and space information. Comparative experimental evaluation results show that the proposed QoS prediction approach can result in much better accuracy in recommending Web services than several other representative ones.

Index Terms—Web service recommendation, QoS prediction, multi-dimensional spatiotemporal model, tensor

1 Introduction

Web services are API-defined software components whose capabilities can be composed and delivered to the clients on-demand through the network [1][2]. When there are many candidate services to evaluate for a capability delivery request, service recommendation is essential to efficiently selecting the best one. Most service recommendation approaches take into account quality-of-service (QoS) characteristics [2]. When necessary QoS measures are not available for a specific candidate service, the unknown QoS values are predicted.

Advances in mobile Internet technology have enabled the clients of Web services to be able to keep their service sessions alive while they are on the move. Since the services consumed by a mobile client may be different over time due to client location changes, a multi-dimensional spatiotemporal model is necessary for analyzing the service consumption relations [3][4][5][6]. In order to make the best service recommendation, historical multi-dimensional QoS data must be exploited as much as possible when predicting necessary unknown QoS values [7][8].

Fig. 1 illustrates the structure of a three dimensional QoS dataset in terms of user (or client), service, and time period. There are m users and n Web services. u_i and s_j denote the i-th user and the j-th service, respectively. t_k denotes the k-th period of time. If a service is invoked in a specific period of time, the QoS value (e.g., response time) of the invocation is recorded for that time period.

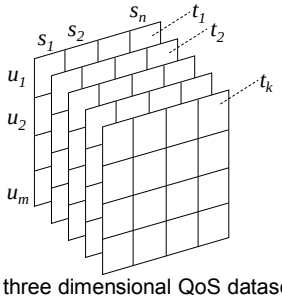


Fig. 1. Structure of a three dimensional QoS dataset in terms of user (or client), service, and time period.

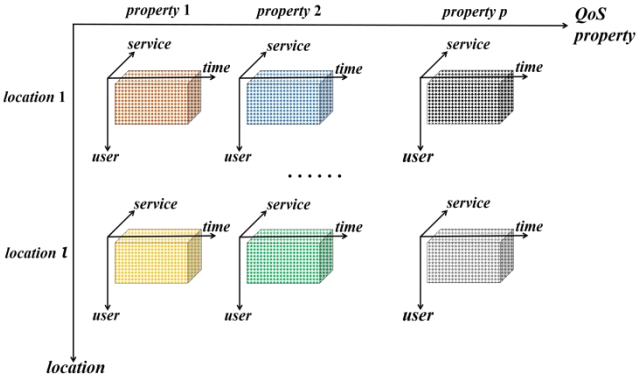


Fig. 2. Structure of a five dimensional QoS data in terms of location, QoS property, user (or client), service, and time period.

- *S. G. Wang, B. Cheng and F. C. Yang are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail: {sgwang; chengbo; fcyang}@bupt.edu.cn.*
- *Y. Ma is with National Satellite Meteorological Center, Beijing 100081, China. E-mail: mayou@cma.gov.cn*
- *R. N. Chang is with IBM Research. E-mail: rong@us.ibm.com.*

Only one QoS property is illustrated in Fig. 1. In practice, several QoS properties can be captured in one QoS dataset with each property modeled as a separate dimension. Moreover, when service requesting location needs be considered, the location data can be modeled via another dimension. Thus, all these QoS data can form a five dimensional space, as shown in Fig. 2 (in which each cuboid is a three dimensional QoS data of Fig. 1).

Although the need of leveraging multi-dimensional QoS data is recognized by many prior studies, the data was not used in an integrated manner. The most common approach is predicting each QoS property respectively and with no dependency considerations regarding time, location and other factors [2][9][10][11]. More specifically, most related work focused on the characteristic of time [8] [12] or the location [7][13], but did not consider the dependencies among all QoS data from the viewpoint of the user (or the service client).

Drawbacks of lacking an integrated approach to analyzing multi-dimensional QoS data are: (1) the integral structure of the multi-dimensional QoS data is neglected for QoS prediction, which prejudices more accurate prediction result; (2) QoS prediction methods using characteristics specific for one dimension are difficult to be expanded to the other dimensions, which requires the suitable prediction methods to be complex to consider the characteristics of all QoS dimensions; and (3) when additional dimensions are needed (e.g., due to deployment of new network computing technologies), design of new prediction methods would be required.

In light of the aforementioned problems, this paper presents an integrated QoS prediction approach (named HDOP) that exploits a high-dimension-oriented QoS prediction method for Web service recommendations. Based on our previous work [1], this approach (1) unifies the modeling of multi-dimensional QoS data via a multi-linear-algebra based concept of tensor, which integrally and wholly considers the contracture of multi-dimensional QoS data; (2) employs efficient optimization algorithms for tensor decomposition and reconstruction; and (3) enables accurate QoS prediction.

The main contributions of this paper are: (1) adopting the mathematical concept of tensor to facilitate structure-aware analysis of QoS data of arbitrary dimensions; (2) creating an efficient tensor decomposition optimization algorithm for efficient QoS value prediction with an in-depth theoretical analysis; (3) allowing the key parameters of the proposed QoS prediction algorithm to be set optimally by theoretical derivations; (4) creating a credible multi-dimensional QoS dataset from a measured taxi usage dataset via a nontrivial data transformational approach; and (5) comparing the accuracy of the proposed QoS prediction approach with that of several other representative ones via a credible multi-dimensional QoS dataset.

The rest of this paper is organized as follows. Section 2 presents the design rationale and algorithms of HDOP. Section 3 describes the process we developed for transforming a measured taxi usage dataset into a multi-dimensional QoS dataset. Section 4 illustrates our comparative experimental evaluation results for HDOP, and how its optimal parameter settings can be deduced. Section 5 presents related work, and Section 6 concludes the paper.

2 HDOP Design Rationale and Algorithms

Given a multi-dimensional QoS dataset, HDOP aims at enabling accurate prediction of the unknown QoS values in any dimension easily and efficiently by considering all QoS dimensions uniformly and in an integrated manner. In order to achieve the goal, we need to decide on (1) how to model multi-dimensional QoS data and (2) how to use the QoS data model to make the predictions.

As illustrated in Fig. 2, multi-dimensional QoS data can be modeled well as a multi-dimensional array. Although predicting QoS values via a two-dimensional matrix is common in Web service recommendation literature, QoS prediction via a multi-dimensional array has not been investigated in depth. We adopt the multi-linear-algebra concept of tensor to model multi-dimensional QoS data because tensors are essentially multi-dimensional arrays and have been widely studied in physics. After illustrating the tensor-based QoS data model, we present HDOP QoS prediction algorithms.

2.1 Multi-Dimensional QoS Data as a Tensor

In this section, we first introduce some key properties of tensor, followed by presenting the tensor decomposition operation, based upon which the unknown QoS can be predicted.

2.1.1 Properties of Tensor

The concept of tensor arose from multi-linear algebra, and was widely used in physics and engineering, e.g., general relativity. Tensors are often denoted by boldface Euler script letters (e.g., $\boldsymbol{\mathcal{X}}$) [14]. $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is an N-dimensional tensor, and the length of its n-th dimension is I_n ($1 \leq n \leq N$). Moreover, as Section 2.1.2 will show, some useful tensor operations make it easy to predict QoS values when tensors are used to model multi-dimensional QoS data. With reference to the QoS data shown in Fig. 2, there are five dimensions: m users, n Web services, k time periods, l locations and p QoS properties. The QoS data can be modeled as a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{m \times n \times k \times l \times p}$. $\boldsymbol{\mathcal{X}}_{i_1 i_2 i_3 i_4 i_5}$ is an entry of $\boldsymbol{\mathcal{X}}$, denoting the value of the i_5-th QoS property for the i_1-th user invoking the i_2-th Web service. This invocation must occur in the i_3-th time period and at the i_4-th location. We note that $\boldsymbol{\mathcal{X}}_{i_1 i_2 i_3 i_4 i_5}$ may not exist (i.e., we may not know its value) since the i_1-th user may not invoke the i_2-th Web service in the i_3-th time period at the i_4-th location. In that case, it would be desirable for the recommender system in use to predict the value of $\boldsymbol{\mathcal{X}}_{i_1 i_2 i_3 i_4 i_5}$ when it comparatively evaluates if the i_2-th Web service is suitable for recommendation for the i_1-th user.

2.1.2 Tensor Decomposition

Although tensor is a very simple concept, its operations can be complicated, such as transforming a multi-dimensional tensor into a two-dimensional matrix, multiplying two tensors together, decomposing a tensor,

and so on [14]. Based upon the concept of *rank one tensor*, HDOP uses only tensor decomposition to predict unknown QoS values.

Definition 1: Rank one tensor—If an N-dimensional tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\boldsymbol{\mathcal{X}} = a^{(1)} \circ a^{(2)} \circ \cdots \circ a^{(N)}$, where $a^{(i)}$ denotes the i-th vector whose length is I_i ($1 \le i \le N$), and symbol "$\circ$" represents the vector outer product, then $\boldsymbol{\mathcal{X}}$ is a *rank one tensor*, and $\boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N} = a_{i_1}^{(1)} a_{i_2}^{(2)} \cdots a_{i_N}^{(N)}$ for all $1 \le i_n \le I_n$ ($1 \le n \le N$).

A rank one tensor has the simplest form of decomposition, i.e., the outer product of N vectors. Since not all tensors are rank one, we decompose a tensor by the following two methods when necessary:

1) A tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is seen as the sum of R rank one tensors.

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} \boldsymbol{\mathcal{X}}_r \tag{1}$$

where each $\boldsymbol{\mathcal{X}}_r$ is a rank one tensor and $\boldsymbol{\mathcal{X}}_r \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. R is called the *rank* of $\boldsymbol{\mathcal{X}}$.

2) Decompose each $\boldsymbol{\mathcal{X}}_r$ as the outer product of N vectors.

$$\boldsymbol{\mathcal{X}}_r = a_r^{(1)} \circ a_r^{(2)} \circ \cdots \circ a_r^{(N)} \tag{2}$$

where $a_r^{(j)}$ denotes a vector whose length is I_j ($1 \le j \le N$) and the subscript r indicates that $a_r^{(j)}$ is $\boldsymbol{\mathcal{X}}_r$ specific, the superscript (j) indicates that $a_r^{(j)}$ is the j-th vector in (2).

Therefore, a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ can be decomposed as:

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} a_r^{(1)} \circ a_r^{(2)} \circ \cdots \circ a_r^{(N)} \tag{3}$$

If we let every $a_r^{(j)}$ in (3) ($1 \le j \le N,\ 1 \le r \le R$) be a column vector, then for a given j ($1 \le j \le N$), the R columns $a_1^{(j)}, a_2^{(j)}, \ldots, a_R^{(j)}$ constitute an $I_j \times R$ matrix which is denoted by $A^{(j)}$, and therefore $a_r^{(j)}$ is the r-th column of $A^{(j)}$. If $a_r^{(j)}$ is written as $A_{:r}^{(j)}$ ($1 \le j \le N,\ 1 \le r \le R$), then (3) is equivalent to:

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} A_{:r}^{(1)} \circ A_{:r}^{(2)} \circ \cdots \circ A_{:r}^{(N)} \tag{4}$$

Equation (4) is the final tensor decomposition formulation we adopt. When decomposing a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, we calculate out each of its ***component matrices*** $A^{(j)}$ ($1 \le j \le N$). Since it is uneasy to do the calculations precisely in practice, the "=" is usually replaced by "≈" to reflect that the right side of (4) is a reconstructed and approximate tensor of the original tensor $\boldsymbol{\mathcal{X}}$. For ease of illustration, we let $\widehat{\boldsymbol{\mathcal{X}}}$ denotes the right side of (4)—the reconstructed and approximate tensor of $\boldsymbol{\mathcal{X}}$.

Even when $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is a sparse tensor, it can be decomposed as (4). All the values of $\boldsymbol{\mathcal{X}}$ including the unknown values can be estimated from (4) as:

$$\boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N} \approx \widehat{\boldsymbol{\mathcal{X}}}_{i_1 i_2 \cdots i_N} = \sum_{r=1}^{R} A_{i_1 r}^{(1)} \cdot A_{i_2 r}^{(2)} \cdot \cdots \cdot A_{i_N r}^{(N)} \tag{5}$$

We call (5) *tensor reconstruction*, as opposed to (4). We decompose a tensor as (4) to find its component matrices, and then we reconstruct this tensor using those component matrices as (5) to estimate the unknown values. If we can calculate out the suitable decomposition of a tensor, its unknown values can be predicted by (5). Thus, finding out the best value of R for (4) is critical for the calculation. Although many studies have been carried out on optimally setting the value of R [14], in practice, R is often manually assigned to several values and tested for the best one. Let ε denotes the precision requirement of tensor decomposition, and if:

$$\left\| \boldsymbol{\mathcal{X}} - \widehat{\boldsymbol{\mathcal{X}}} \right\|^2 = \sum_{\substack{\boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N} \\ \text{is known}}} {e_{i_1 i_2 \cdots i_N}}^2 \ \le \varepsilon \tag{6}$$

where $e_{i_1 i_2 \cdots i_N} = \boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N} - \widehat{\boldsymbol{\mathcal{X}}}_{i_1 i_2 \cdots i_N}$, and ε is often manually assigned, then the smaller R satisfying (6) is better since it can reduce the calculations of QoS tensor decomposition and reconstruction. More detailed explanations about tensor decomposition can be found in Appendix A.

2.2 QoS Prediction Using Tensor Model

In this section, we present how to predict unknown QoS values using the QoS tensor established in Section 2.1. In short, after computing the component matrices satisfying (6), we can predict the unknown QoS values by reconstructing the QoS tensor as (5)

2.2.1 Computing the Component Matrices

Computing the component matrices satisfying (6) is essentially an optimization problem; hence we have to determine two things: (1) our optimization goal; and (2) our optimization algorithm

Our Optimization Goal

If we just adopt (6) as the optimization goal to compute the component matrices, we are likely to get overfitted results since a QoS dataset often contains noises. To avoid the overfitting issue, we determine the optimization goal of the component matrices computing as (7), which is a modification from (6). We call L the loss function of our optimization goal.

$$L = \sum_{\substack{\boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N} \\ \text{is known}}} \Big[{e_{i_1 i_2 \cdots i_N}}^2 + \lambda \left(\left\| A_{i_1:}^{(1)} \right\|^2 + \left\| A_{i_2:}^{(2)} \right\|^2 + \cdots + \left\| A_{i_N:}^{(N)} \right\|^2 \right) \Big] \le \varepsilon \tag{7}$$

where $\lambda \left(\left\| A_{i_1:}^{(1)} \right\|^2 + \left\| A_{i_2:}^{(2)} \right\|^2 + \cdots + \left\| A_{i_N:}^{(N)} \right\|^2 \right)$ is the regularization term to avoid overfitting, and λ is the regularization parameter. The notation $A_{i_j:}^{(j)}$ denotes the i_j-th row of the j-th component matrix $A^{(j)}$.

Our Optimization Algorithm

To compute each component matrix $A^{(j)}$ $(1 \le j \le N)$, we have to compute each entry $a_{i_j r}^{(j)}$ of every $A^{(j)}$ $\left(1 \le j \le N, 1 \le i_j \le I_j, 1 \le r \le R\right)$.

A simple approach to completing the computation task is using the gradient descent method, but this method may encounter the weakness of slow convergence and the difficulty in determining the suitable iteration step size for each entry of every component matrix. To avoid such weaknesses, we adopt iRPROP$^+$ [15] as the optimization algorithm to compute the component matrices. iRPROP$^+$ is a more robust and faster variation of RPROP [27], and the readers can be referred to [28][29][30][31] for the outperformance of RPROP compared with many common optimization algorithms, such as gradient descent, batch gradient descent, Quickprop and so on. iRPROP$^+$ has two significant advantages: (1) different variables are computed with respective iteration step sizes; and (2) for each variable, its iteration step size is adaptively adjusted with the iterative computing to guarantee fast convergence. We use the component matrix entries as the variables. All entries of the component matrices undergo the same iterative computing process. Each iteration of computing the entry $a_{i_j r}^{(j)}$ is performed by the following two steps:

Step 1. Computing the partial derivatives of L with respect to $a_{i_j r}^{(j)}$ as:

$$g_{i_j r}^{(j)} = \frac{\partial L}{\partial a_{i_j r}^{(j)}} = \sum_{\substack{x_{i_1 i_2 \cdots i_N} \\ \text{is known}}} \left(-2 e_{i_1 i_2 \cdots i_N} a_{i_1 r}^{(1)} a_{i_2 r}^{(2)} \cdots a_{i_{j-1} r}^{(j-1)} a_{i_{j+1} r}^{(j+1)} \cdots a_{i_N r}^{(N)} + 2\lambda a_{i_j r}^{(j)}\right) \quad (8)$$

which is worth noting is that the value of $g_{i_j r}^{(j)}$ changes with the iterations of $a_{i_j r}^{(j)}$, and we use $g_{i_j r}^{(j)}|_t$ to denote the value of $g_{i_j r}^{(j)}$ if the (t-1)-th iteration has been just finished but the t-th has not started.

Step 2. Updating the iteration step size and doing iteration. Supposing that the t-th iteration is the coming iteration, if $g_{i_j r}^{(j)}|_t \cdot g_{i_j r}^{(j)}|_{t-1}$ is greater than zero (which means that the coming t-th iteration has the same convergence direction[1] with the (t-1)-th), we increase the iteration step size to accelerate convergence:

$$\delta_{i_j r}^{(j)}|_t = \delta_{i_j r}^{(j)}|_{t-1} \cdot \eta^+ \quad (9)$$

where $\delta_{i_j r}^{(j)}|_t$ is the iteration step size of $a_{i_j r}^{(j)}$ for its t-th iteration. $\delta_{i_j r}^{(j)}$ is increased by $\eta^+ (> 1)$ which is called increase factor. We use $a_{i_j r}^{(j)}|_t$ to denote the value of $a_{i_j r}^{(j)}$ if the (t-1)-th iteration has just been finished but the t-th has not started, then the t-th iteration will be computed as:

$$a_{i_j r}^{(j)}|_{t+1} = a_{i_j r}^{(j)}|_t - \mathbf{sign}\left(g_{i_j r}^{(j)}|_t\right) \cdot \delta_{i_j r}^{(j)}|_t \quad (10)$$

with $\mathbf{sign}(x) = \begin{cases} 1, & if\ x > 0 \\ -1, & if\ x < 0 \\ 0, & \text{otherwise} \end{cases}$.

If $g_{i_j r}^{(j)}|_t \cdot g_{i_j r}^{(j)}|_{t-1}$ is less than zero, then the (t-1)-th iteration step size $\delta_{i_j r}^{(j)}|_{t-1}$ is too large and the (t-1)-th iteration missed the minimum of L. In this case, if:

$$L|_t > L|_{t-1} \quad (11)$$

where $L|_t$ denotes the value of L when the (t-1)-th round iterations of all component matrices entries have just been finished, we abandon $a_{i_j r}^{(j)}|_t$ (since L became bigger which conflicts with our optimization goal), and in the coming t-th iteration $a_{i_j r}^{(j)}|_{t+1}$ is evaluated as $a_{i_j r}^{(j)}|_{t-1}$,

$$a_{i_j r}^{(j)}|_{t+1} = a_{i_j r}^{(j)}|_{t-1} \quad (12)$$

which is actually a rollback to the latest eligible iteration in order to eliminate the error of the (t-1)-th iteration. If (11) is not true (which means that the (t-1)-th iteration is eligible since L became smaller), in the coming t-th iteration $a_{i_j r}^{(j)}|_{t+1}$ is computed with a step size smaller than $\delta_{i_j r}^{(j)}|_{t-1}$. This smaller step size is:

$$\delta_{i_j r}^{(j)}|_t = \delta_{i_j r}^{(j)}|_{t-1} \times \eta^- \quad (13)$$

where $\eta^- (0 < \eta^- < 1)$ is the decrease factor, and $a_{i_j r}^{(j)}|_{t+1}$ is computed as (10).

If $g_{i_j r}^{(j)}|_t \cdot g_{i_j r}^{(j)}|_{t-1}$ equals zero, then the t-th iteration step size remain the same with the (t-1)-th, i.e., $\delta_{i_j r}^{(j)}|_t = \delta_{i_j r}^{(j)}|_{t-1}$, and in the t-th iteration $a_{i_j r}^{(j)}|_{t+1}$ is computed as (10).

After each round of iterations for all component matrix entries, the loss function L is computed, the computing would be finished once $L \le \varepsilon$, and then all component matrices $A^{(j)}$ $(1 \le j \le N)$ are obtained.

Algorithm 1 details the process of computing the component matrices via pseudo code. The notations used in the algorithm presentation are listed in Table 1. We note that the iteration step size in Algorithm 1 is computed as $\delta_{i_j r}^{(j)} = \boldsymbol{min}\left(\delta_{i_j r}^{(j)} \times \eta^+, Max\_Step\_Size\right)$ or $\delta_{i_j r}^{(j)} = \boldsymbol{max}\left(\delta_{i_j r}^{(j)} \times \eta^-, Min\_Step\_Size\right)$, which are slightly different from (9) and (13). This is caused by the computing accuracy limitation of computers. If the iteration step size is too big or too small, $a_{i_j r}^{(j)}$ may become *Infinity* or *–Infinity* in programming languages like Java. For our experiments, we set $Max\_Step\_Size = 0.1$ and $Min\_Step\_Size = 1\text{E} - 8$.

[1] convergence direction of the k-th iteration is -**sign**($g_{i_j r}^{(j)}|_k$), shown as (10)

TABLE 1
DENOTATIONS OF SOME NOTATIONS

notation	denotes	appears in
$\boldsymbol{\mathcal{X}}$	A tensor, which is used to model the multi-dimensional QoS	
L	Loss function of the component matrices computing	(7), (11)
$A_{i_j r}^{(j)}$	The entry which locates at the i_j-th row and the r-th column of the component matrix $A^{(j)}$.	(8),(10),(12)
$g_{i_j r}^{(j)}$	The first order partial derivative of L with respect to $A_{i_j r}^{(j)}$.	(8),(10)
$\delta_{i_j r}^{(j)}$	the iteration step size of $A_{i_j r}^{(j)}$	(9),(10),(13)
η^+	increase factor, which make iteration step sizes bigger	(9)
η^-	decrease factor, which make iteration step sizes smaller	(13)
R	$\boldsymbol{\mathcal{X}}$ can be seen as the sum of R rank one tensors	(1),(3),(4),(5)
ε	Precision requirement of the component matrices computing, namely that $L \leq \varepsilon$ is the optimization goal	(6),(7)
λ	regularization parameter of the optimization goal	(7)

2.2.2 Making Predictions

Once the component matrices of $\boldsymbol{\mathcal{X}}$ have been calculated out by Algorithm 1, each unknown QoS value (e.g., $\boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N}$) can be predicted by (5), i.e., $\boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N} \approx \sum_{r=1}^{R} A_{i_1 r}^{(1)} \cdot A_{i_2 r}^{(2)} \cdot \cdots \cdot A_{i_N r}^{(N)}$.

2.2.3 Computational Complexity Analysis

HDOP works by two steps, i.e., tensor decomposition and tensor construction. According to (5), HDOP predicts an unknown QoS value by tensor reconstruction with $O(1)$ computational complexity, and therefore the computational complexity of HDOP is mainly caused by tensor decomposition which is implemented by algorithm 1. Now we analyze the computational complexity of algorithm 1.

The loop boundary of Algorithm 1 is $1 \leq j \leq N$ && $1 \leq i_j \leq I_j$ && $1 \leq r \leq R$ (as shown on line 10), where N is the dimension of training set $\boldsymbol{\mathcal{X}}$, I_j is the length of the *j-th* dimension of $\boldsymbol{\mathcal{X}}$, and R is the rank of $\boldsymbol{\mathcal{X}}$. On the surface, the computation amount of each loop is $R \cdot \prod_{j=1}^{N} I_j$, where $\prod_{j=1}^{N} I_j$ is the volume of $\boldsymbol{\mathcal{X}}$, i.e., the total number of known and unknown values in $\boldsymbol{\mathcal{X}}$. In reality, Algorithm 1 works only based on the known values of $\boldsymbol{\mathcal{X}}$, which is shown as (7) and (8). Therefore the computational complexity of each loop of algorithm 1 is $O(C \cdot R)$, where C is number of known values of $\boldsymbol{\mathcal{X}}$, and the computational complexity of HDOP is $O(S \cdot C \cdot R)$, where S is the number of iterations required for the convergence of (7) which is the optimization goal of tensor decomposition. Almost all the optimization algorithms including HDOP cannot converge in definite iterations, namely that we cannot determine S exactly, whereas iRPROP+—the core algorithm of HDOP is one of the fastest optimization mechanisms [28][29][30][31]. The reasons for the fast convergence of iRPROP+ are given in Appendix B.

Algorithm 1: component matrices computing

Input: $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}, \varepsilon, \lambda, \delta, R$

Output: $A^{(j)}s$ for $j = 1$ to N

1 **Initialize** all $A^{(j)}s$ //which can be seen as the 0^{th} round iterations;
2 $l'' = L$ //if we need to judge whether (11) is true then l'' denotes $L|_{t-1}$
3 **for** *each* $A_{i_j r}^{j}(1 \leq j \leq N, 1 \leq i_j \leq I_j, 1 \leq r \leq R)$ **do**
4 //1^{st} round iterations;
5 $g_{i_j r}^{(j)'} = g_{i_j r}^{(j)}$;
6 $A_{i_j r}^{(j)'} = A_{i_j r}^{(j)}$//if the rollback shown as (12) is needed,$A_{i_j r}^{(j)'}$ denotes $A_{i_j r}^{(j)}|_{t-1}$;
7 $A_{i_j r}^{(j)} = A_{i_j r}^{(j)} - \mathbf{sign}\left(g_{i_j r}^{(j)}\right) \cdot \delta_{i_j r}^{(j)}$
8 **repeat** //other rounds of iterations for computing component matrices
9 $l' = L$ //if we need to judge whether (11) is true then l' denotes $L|_t$;
10 **for** *each* $A_{i_j r}^{j}(1 \leq j \leq N, 1 \leq i_j \leq I_j, 1 \leq r \leq R)$ **do**
11 **if** $g_{i_j r}^{(j)} \cdot g_{i_j r}^{(j)'} > 0$ **then**
12 $A_{i_j r}^{(j)'} = A_{i_j r}^{(j)}$;
13 $g_{i_j r}^{(j)'} = g_{i_j r}^{(j)}$;
14 $\delta_{i_j r}^{(j)} = \mathbf{min}\left(\delta_{i_j r}^{(j)} \cdot \eta^+, Max\_Step\_Size\right)$;
15 $A_{i_j r}^{(j)} = A_{i_j r}^{(j)} - \mathbf{sign}\left(g_{i_j r}^{(j)}\right) \cdot \delta_{i_j r}^{(j)}$
16 **else if** $g_{i_j r}^{(j)} \cdot g_{i_j r}^{(j)'} < 0$ **then**
17 **if** $l' > l''$ **then**
18 $g_{i_j r}^{(j)'} = g_{i_j r}^{(j)}$;
19 $A_{i_j r}^{(j)} = A_{i_j r}^{(j)'}$// if (11) is true then rollback as (12);
20 $\delta_{i_j r}^{(j)} = \mathbf{max}\left(\delta_{i_j r}^{(j)} \times \eta^-, Min\_Step\_Size\right)$;
21 **else**
22 $A_{i_j r}^{(j)'} = A_{i_j r}^{(j)}$;
23 $g_{i_j r}^{(j)'} = g_{i_j r}^{(j)}$;
24 $\delta_{i_j r}^{(j)} = \mathbf{max}\left(\delta_{i_j r}^{(j)} \cdot \eta^-, Min\_Step\_Size\right)$;
25 $A_{i_j r}^{(j)} = A_{i_j r}^{(j)} - \mathbf{sign}\left(g_{i_j r}^{(j)}\right) \cdot \delta_{i_j r}^{(j)}$
26 **else**
27 $A_{i_j r}^{(j)'} = A_{i_j r}^{(j)}$;
28 $g_{i_j r}^{(j)'} = g_{i_j r}^{(j)}$;
29 $A_{i_j r}^{(j)} = A_{i_j r}^{(j)} - \mathbf{sign}\left(g_{i_j r}^{(j)}\right) \cdot \delta_{i_j r}^{(j)}$
30 $l'' = l'$;
31 **until** *if* $\boldsymbol{L \leq \varepsilon}$ *or maximum iterations exhausted*;

3 BTP: A CREDIBLE HIGH DIMENSIONAL QOS DATASET

In this section, we proposed a new dataset for QoS prediction. The dataset contains multiple properties such as time, location, and others.

3.1 Rationale for the Use of the BTP Dataset

There are many QoS datasets used for Web service related researches, such as WSDream[2] proposed by Zheng, QWS[3] proposed by Al-Masri and Mahmoud and so on. However, none of them contain both time and location information,

[2] www.wsdream.net/dataset.html

[3] http://www.uoguelph.ca/~qmahmoud/qws/

which is crucial for researching on service recommendation over high dimensional space-time. We decided to create one credible multi-dimensional QoS dataset via the Beijing taxi passengers (BTP) dataset which contains information on the number of people called and got off taxies at different time and locations.

3.2 Transformation of the BTP Dataset

BTP is compiled from the Beijing taxi GPS raw data of November 2012, which is available from datatang[4]—a popular data sharing platform in China. As shown in Fig. 3, the raw data contains about one billion records, each of which consists of taxi ID, longitude, latitude, speed, travel direction, carrying passengers or not, sampling time and so on. We extracted one-week BTP data records (from 2012.11.05 to 2012.11.11) to create the multi-dimensional QoS dataset we need.

taxi ID	longitude	latitude	speed	driving direction	carrying passenger or not	sampling time

Fig .3 Record structure for the Beijing taxi GPS raw data

For a specific taxi, value change of the field *carrying passenger or not* to *Yes* from *No* means a passenger called the taxi at the recorded sampling time. Similarly, changing to *No* from *Yes* of the field means a passenger got off the taxi at the recorded sampling time.

Below is a summary of the data transformation steps we used to create our BTP-based multi-dimensional QoS dataset:

(1) We divided the zone of Beijing within Third Ring Road into 3600 location grids, each of which is 240m×240m in size, as shown in Fig. 4.
(2) We also divided each day of the extracted week into 144 10-minute time intervals.
(3) We sorted all the records of each taxi by sampling time in ascending order.
(4) For each record of a specific taxi, if the value of the field *carrying_passenger_or_not* changes to *No* from *Yes*, we can find out in which location grid and at which time interval a passenger called the taxi via the record's values for *longitude, latitude,* and *sampling_time*.
(5) When the traversal of all the records of all the taxies is finished, we can find out the number of people who have called taxies in a specific location grid and/or at a specific time interval.
(6) Similarly, we can derive the number of people who have got off taxies in a specific location grid and/or at a specific time interval.

Thus, for every location grid, the generated dataset can provide two values for every time interval: how many people have called taxies (*denoted as CT*) and how many people have got off taxies (*denoted as GT*).

3.3 Exploitation of the BTP Dataset

Strictly, BTP is not a traditional Web service QoS dataset as WSDream and QWS, etc., but it can still be used in Web-based service researches, e.g., taxi recommendation. CT and GT can be seen as QoS properties of each grid by which recommender systems can suggest taxi drivers where are apt to get passengers (according to CT), and where should circumvent since there are already many empty taxis (according to GT). A big advantage of BTP is that it contains high dimensional time and space information together.

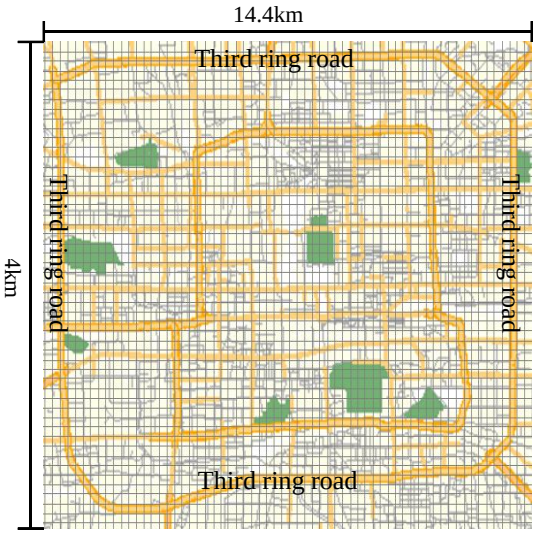


Fig. 4. For each location grid, the transformed BTP dataset surveyed on two things for every time interval: how many people called taxi and how many people got off taxi. We divided the city zone of Beijing within the third ring road into 3600 location grids, each of which is 240m×240m in size. We also divided each day into 144 10-minute time intervals.

4 Comparative Experimental Evaluations

This section presents the experimental evaluations we performed (via the WSDream and the GTP datasets) for validating HDOP's approach to service recommendation and for comparing the QoS prediction accuracy of HDOP with three other representative ones. WSDream is a real Web service QoS dataset which has been widely studied. It contains two types of QoS properties (response time and throughput) collected from 4,532 Web services for 142 users in 64 time intervals.

We will explain how to set the parameters of HDOP in Section 4.3. We note that for a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, its headmost entry is denoted as $\boldsymbol{\mathcal{X}}_{1,1,\ldots,1}$ in Section 2 for ease of illustration, but will be denoted as $\boldsymbol{\mathcal{X}}_{0,0,\ldots,0}$ in this section to conform the common programming practices.

4.1 Experiments Setup

In this section, we configure the experimental environment,

[4] www.datatang.com/data/44502

establish the tensor, and definite the performance metric.

4.1.1 Hardware and Software

We implemented the experiments via JDK 7.0 and eclipse 4.3 on an IBM server with Inter Xeon E5-2670 eight-core 2.60 GHz CPU and 32G RAM.

4.1.2 HDOP Models WSDream and BTP as Tensors

When performing the experimental evaluations, we noticed the large data volumes of WSDream and BTP caused long execution time. WSDream contains $142 \times 4532 \times 64 \times 2$ QoS values, and BTP contains $3600 \times 144 \times 7 \times 2$ QoS values. If we model the whole WSDream or BTP as a tensor, then for most existing optimization algorithms, one round iterations of so many values would take a long time to complete. Therefore we model the WSDream as a tensor set $\boldsymbol{\mathcal{Y}} = \{\boldsymbol{\mathcal{Y}}^{(k)} | k = 0,1,..,21\}$, each tensor $\boldsymbol{\mathcal{Y}}^{(k)} \in \mathbb{R}^{2\times64\times142\times206}$, namely that we divided the 4532 Web services into 22 groups, and each group contains 206 Web services. Based on each Web service group, we establish a tensor $\boldsymbol{\mathcal{Y}}^{(k)}$. Similarly, for BTP, we divided the 3600 grids into 2 groups, and each group contains 1800 grids. We established a tensor $\boldsymbol{\mathcal{Z}}^{(k)}$ for each of the grid groups and modeled BTP as a tensor set $\boldsymbol{\mathcal{Z}} = \{\boldsymbol{\mathcal{Z}}^{(k)} | k = 0,1\}$ such that each tensor $\boldsymbol{\mathcal{Z}}^{(k)} \in \mathbb{R}^{2\times7\times144\times1800}$. $\boldsymbol{\mathcal{Y}}^{(k)}$ and $\boldsymbol{\mathcal{Z}}^{(k)}$ are illustrated as Fig. 5. A big benefit of dividing WSDream (or BTP) into relative smaller tensors is that all tensors in $\boldsymbol{\mathcal{Y}}$ (or $\boldsymbol{\mathcal{Z}}$) share the same rank (as defined in (1)), which will be elaborated in Section 4.3.3.

For example, $\boldsymbol{\mathcal{Y}}^{(0)}{}_{0,1,2,3}$ is a QoS value which only exists in tensor $\boldsymbol{\mathcal{Y}}^{(0)}$, and this value is of the 0th QoS property (*the 0th QoS property is response time and the 1st is throughput*) of the 3rd Web service of $\boldsymbol{\mathcal{Y}}^{(0)}$ for the 2nd user at the 1st time interval. $\boldsymbol{\mathcal{Z}}^{(1)}{}_{1,2,47,99}$ is a QoS value which only exists in tensor $\boldsymbol{\mathcal{Z}}^{(1)}$ and this value is of the 1st QoS property (*the 0th QoS property is CT and the 1st is GT*) of the 99th grid of $\boldsymbol{\mathcal{Z}}^{(1)}$ in the 47th time interval of the 2nd day (*Wednesday*).

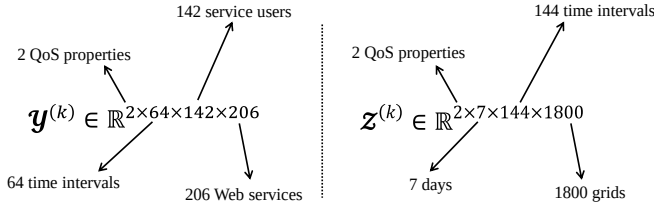


Fig. 5. Dimensions of $\boldsymbol{\mathcal{Y}}^{(k)}$ and $\boldsymbol{\mathcal{Z}}^{(k)}$

4.1.3 Accuracy Metrics

We adopt the widely used *Mean Absolute Error* (MAE) and *Root Mean Square Deviation* (RMSE) as the accuracy metrics for our experiments. MAE and RMSE are frequently used to measure the difference between values predicted by a model or estimator and observed values. The definition of the two metrics is given below:

$$MAE = \frac{\sum \left| R_{ij} - R_{ij} \right|}{N} \tag{14}$$

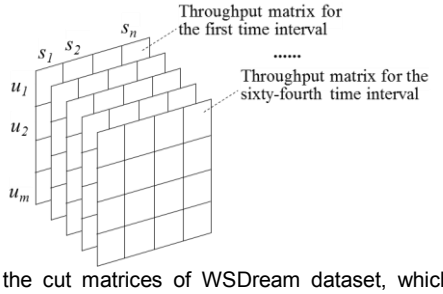
$$RMSE = \sqrt{\frac{\sum \left(R_{ij} - R_{ij} \right)^2}{N}} \tag{15}$$

where R_{ij} denotes the QoS value of service j observed by user i, R_{ij} is the predicted QoS value, and N is the number of predicted values.

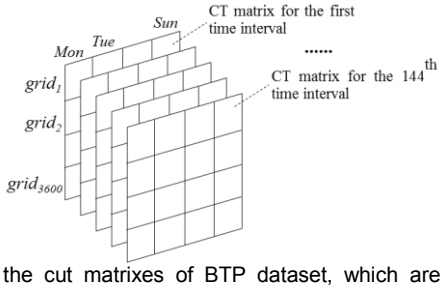
4.2 Performance Comparisons

We compare the multi-dimensional QoS prediction performance of HDOP with three other representative QoS prediction methods. Two of them are special characteristic based methods, which make prediction based on some special characteristics, such as context-aware, time-aware, and location-aware and so on. One of them is a time-aware method (TA), which was proposed in reference [12]. We compared HDOP with this TA method via the WSDream dataset. The other is a location-aware method (LA), which is proposed in reference [7]. We compared HDOP with this LA method via the BTP dataset.

The third compared method is matrix factorization based method (MF), which makes prediction by factorizing the user-service matrix wherein each entry is the QoS value of a user-service invocation. We selected the method proposed in reference [16] to compare with. We compared HDOP with this MF method on both WSDream and BTP datasets. Traditionally, this type of methods mostly works on two-dimensional matrices. For comparison, we cut the datasets of WSDream and BTP into matrices as shown in Fig. 6, and make prediction on each matrix by the selected MF method.



(a) Parts of the cut matrices of WSDream dataset, which are 64 throughput matrices, and there are also 64 such matrices for response time.



(b) Parts of the cut matrixes of BTP dataset, which are 144 CT matrixes, and there are also 144 such matrixes for GT.

Fig. 6. For comparison, the high dimensional datasets are segmented into matrixes. Predictions are made by the compared MF method on each matrix.

TABLE 2
ACCURACY COMPARISON

	Methods	Traing Data=5%		Traing Data=10%		Traing Data=20%		Traing Data=50%	
		MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
Response Time of **WSDream**	TA	0.6179	1.5798	0.6013	1.5487	0.5994	1.5238	0.4877	1.4856
	MF	0.5690	1.4794	0.4987	1.2843	0.4495	1.1862	0.4013	1.0823
	HDOP	**0.3825**	**0.9476**	**0.3076**	**0.7602**	**0.2276**	**0.5628**	**0.1237**	**0.3156**
Throughput of **WSDream**	TA	27.5612	76.2975	17.2365	52.2187	15.0994	51.0288	14.9870	47.8876
	MF	20.4132	55.3479	16.3214	47.2143	14.1478	42.1162	14.9013	41.8863
	HDOP	**18.0112**	**52.2345**	**13.2578**	**42.7602**	**10.1276**	**36.5748**	**10.0037**	**35.4379**
CT and GT of **BTP** are validated together	LA	20.2876	60.4207	15.3286	46.0236	12.0216	37.5143	10.5643	29.8163
	MF	16.6541	50.6571	13.0247	37.1258	11.0291	35.1490	9.9531	25.3654
	HDOP	**8.3732**	**25.0023**	**6.3602**	**16.3602**	**5.0512**	**14.2386**	**3.6123**	**11.2563**

As presented in Section 4.1.2, for our HDOP, WSDream and BTP were molded as tensor sets $\boldsymbol{\mathcal{Y}} = \{\boldsymbol{\mathcal{Y}}^{(k)} | k = 0,1,..,21\}$ and $\boldsymbol{\mathcal{Z}} = \{\boldsymbol{\mathcal{Z}}^{(k)} | k = 0,1\}$, respectively. For each tensor of $\boldsymbol{\mathcal{Y}}$ and $\boldsymbol{\mathcal{Z}}$, we selected 5%, 10%, 20% and 50% data as training data respectively, and the remainders were test data. Training data were used to compute the component matrices and reconstruct the tensors. The test data were used to verify the prediction accuracy of the reconstructed tensors.

The prediction accuracies of HDOP and the comparisons with other methods are shown in Table 2. With reference to Table 2, we can see that HDOP is more accurate than all of the other methods for the two chosen datasets. As the training data increases from 5% to 50%, the MAE and RMSE values become smaller. Since the two QoS properties of WSDream (i.e., response time and throughput) have different value ranges, we show the comparisons on these two properties separately in Table 2.

4.3 Setting HDOP Parameters

When compared with other QoS prediction methods in Section 4.2, HDOP was set by the optimized parameters, i.e., ε, λ, R, η^+, η^- and the initial step size (see Table 1 for their detailed denotations). This section presents their settings.

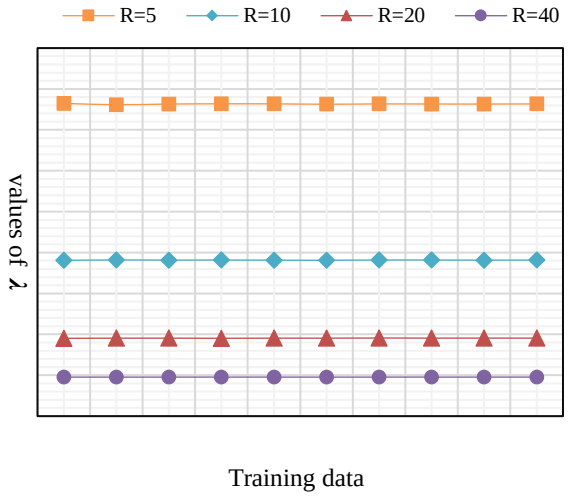
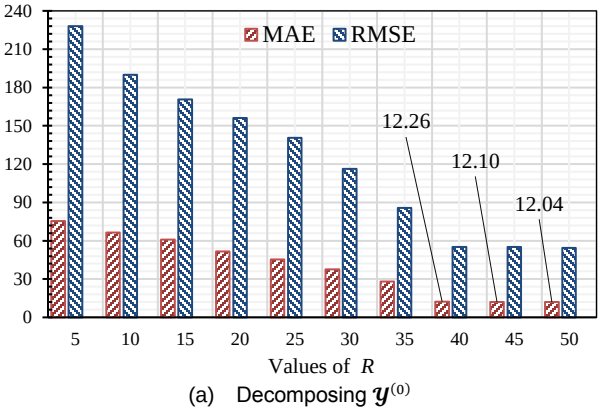
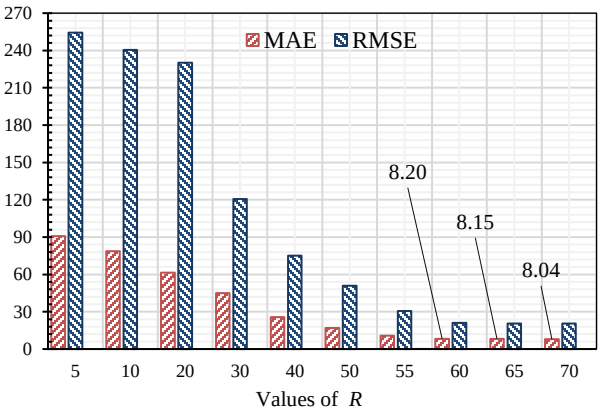


Fig. 7. For a given R, λ stay stable with training data increasing. For a given training data, λ becomes smaller with R increasing.



(a) Decomposing $\boldsymbol{\mathcal{Y}}^{(0)}$



(b) Decomposing $\boldsymbol{\mathcal{Z}}^{(0)}$

Fig. 8. Setting $AvePrec$ to 0.01, we decomposed $\boldsymbol{\mathcal{Y}}^{(0)}$and $\boldsymbol{\mathcal{Z}}^{(0)}$ with 5% training data under different values of *R*. With the value of *R* increasing, MAE and RMSE became smaller meaning that the prediction accuracies became higher. For $\boldsymbol{\mathcal{Y}}^{(0)}$, the prediction accuracies are very close when *R* = 40, 45 and 50, therefore we select 40 as the best value of *R* for WSDream since a smaller *R* can reduce the calculations of tensor decomposition and reconstruction, as the similar reason we select 60 as the best value of *R* for BTP.

4.3.1 Setting of ε

According to (7), ε controls the tensor decomposition precision and is easy to determine. If we let:

$$AvePrec = \frac{\sum_{\substack{x_{i_1 i_2 \cdots i_N} \\ \text{is known}}} e_{i_1 i_2 \cdots i_N}{}^2}{number\ of\ known \boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N}} \tag{16}$$

and if we let the two parts of L equals to each other,

i.e.,$\sum {e_{i_1 i_2 \cdots i_N}}^2 = \sum \lambda \left(\left\| A_{i_1:}^{(1)} \right\|^2 + \left\| A_{i_2:}^{(2)} \right\|^2 + \cdots + \left\| A_{i_N:}^{(N)} \right\|^2 \right)$ (the next section—"*Setting of* λ" shows why we do this), then:

$$\varepsilon = 2 \cdot AvePrec \cdot \left(number\ of\ known\ \boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N} \right) \quad (17)$$

4.3.2 Setting of λ

According to (7), λ controls the proportions of the two parts of loss function L. In previous studies [17][18], the two parts are in direct proportion, shown as:

$$k = \frac{\sum_{\text{is known}} \boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N} {e_{i_1 i_2 \cdots i_N}}^2}{\sum_{\text{is known}} \boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N} \lambda \left(\left\| A_{i_1:}^{(1)} \right\|^2 + \left\| A_{i_2:}^{(2)} \right\|^2 + \cdots + \left\| A_{i_N:}^{(N)} \right\|^2 \right)} \quad (18)$$

As per Miller's suggestion [19], we set k equal to 1 for our proposed HDOP. Thus, λ should be small enough to guarantee the loss function, $L = \sum_{\text{is known}} \boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N} \left[{e_{i_1 i_2 \cdots i_N}}^2 + \lambda \left(\left\| A_{i_1:}^{(1)} \right\|^2 + \left\| A_{i_2:}^{(2)} \right\|^2 + \cdots + \left\| A_{i_N:}^{(N)} \right\|^2 \right) \right] \leq \varepsilon$ can converge to a small ε. Supposing that tensor $\boldsymbol{\mathcal{X}}$ has been decomposed into component matrices precisely, then for a given $\boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N}$, following (5), $\boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N} \approx \sum_{r=1}^{R} A_{i_1 r}^{(1)} \cdot A_{i_2 r}^{(2)} \cdot \cdots \cdot A_{i_N r}^{(N)}$, $A_{i_j r}^{(j)}$ can be approximately evaluated as $A_{i_j r}^{(j)} \approx \sqrt[N]{\frac{\boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N}}{R}}$, therefore for each $\boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N} \in \boldsymbol{\mathcal{X}}$, we have the following equation:

$$\left\| A_{i_1:}^{(1)} \right\|^2 + \left\| A_{i_2:}^{(2)} \right\|^2 + \cdots + \left\| A_{i_N:}^{(N)} \right\|^2 \approx NR \left(\frac{\boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N}}{R} \right)^{\frac{2}{N}} \quad (19)$$

If a small ε is given, which means that tensor $\boldsymbol{\mathcal{X}}$ can be decomposed precisely, then since we have set k equal to 1 in (18), λ is computed by:

$$\sum_{\substack{\boldsymbol{\mathcal{X}}_{i_1 i_2 \cdots i_N} \\ \text{is known}}} \lambda \left(\left\| A_{i_1:}^{(1)} \right\|^2 + \left\| A_{i_2:}^{(2)} \right\|^2 + \cdots + \left\| A_{i_N:}^{(N)} \right\|^2 \right) = \frac{\varepsilon}{2} \quad (20)$$

Taking $\boldsymbol{\mathcal{Y}}^{(0)}$ as an example, the training data for $\boldsymbol{\mathcal{Y}}^{(0)}$ is randomly selected. As the training data increases, the two sides of (20), $\frac{\varepsilon}{2}$ and $\sum \lambda \left(\left\| A_{i_1:}^{(1)} \right\|^2 + \left\| A_{i_2:}^{(2)} \right\|^2 + \cdots + \left\| A_{i_N:}^{(N)} \right\|^2 \right)$ increase synchronously. Thus, λ should stay stable, which can be verified via Fig. 7. If we set $AvePrec$ equal to 0.01 and R is given, then different values of λ for $\boldsymbol{\mathcal{Y}}^{(0)}$ under different training data are shown as Fig.7.

4.3.3 Setting of R

According to (1), R is the rank of a tensor, and it is difficult to exactly determine R [14]. For a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, its rank is related to $I_1, I_2, \cdots$, and I_N [20]. For the tensor set $\boldsymbol{\mathcal{Y}} = \{\boldsymbol{\mathcal{Y}}^{(k)} | k = 0,1,..,21\}$ and $\boldsymbol{\mathcal{Z}} = \{\boldsymbol{\mathcal{Z}}^{(k)} | k = 0,1\}$, each tensor $\boldsymbol{\mathcal{Y}}^{(k)} \in \mathbb{R}^{2 \times 64 \times 142 \times 206}$ and each tensor $\boldsymbol{\mathcal{Z}}^{(k)} \in \mathbb{R}^{2 \times 7 \times 144 \times 1800}$, therefore the ranks of the tensors in a set are the same. We only need to find the best Rs for $\boldsymbol{\mathcal{Y}}^{(0)}$ and $\boldsymbol{\mathcal{Z}}^{(0)}$. Under $AvePrec = 0.01$ and different Rs, we find the best prediction accuracy by experiments, and then we determine the best R. The prediction accuracies of $\boldsymbol{\mathcal{Y}}^{(0)}$ and $\boldsymbol{\mathcal{Z}}^{(0)}$ under different values of R are shown as Fig. 8.

As Fig. 8 shows, we try to select a small R with high prediction accuracy since a smaller R can reduce the calculations of tensor decomposition and reconstruction.

4.3.4 Other Settings

We set $\eta^+ = 1.0001$ and $\eta^- = 0.5$ and find the loss function can get a fast convergence speed.

The initial step sizes of all entries of component matrices should be set to a relatively small value to guarantee the loss function L can develop to convergence early. We set the initial step sizes of all entries of component matrices to 1E-6. We need not worry about the initial step sizes are too small, since the increase factor η^+ can adjust them dynamically.

4.4 Limitations of Our Approach

Although HDOP is very efficient and accurate, it has the following limitations in practice:

(1) HDOP models the whole QoS data as a tensor and hence it is uneasy to have a parallel implementation. Traditional matrix factorization method can model the multi-dimensional QoS as multiple matrices and handle each matrix concurrently.

(2) When QoS data is extremely sparse, QoS tensor may require more storage or memory space than multiple QoS matrices.

5 Related Work

As a core research topic of Web service recommendation, predicting unknown QoS values has been widely studied by many researchers [2][7][8][9][10][11][12][13][16][21]. Since current recommender systems often recommend services in a high dimensional spatiotemporal space [3][4][5][6], the presented HDOP work focuses on how to accurately predict the unknown QoS values in a high dimensional spatiotemporal space.

CF-based methods have been widely used in QoS prediction [2][7][9][13]. There are two types of collaborative filtering approaches—model based vs. neighborhood based. The neighborhood based collaborative filtering approach was implemented mainly by user based methods, item-based methods, or a combination of both types of methods. In user-based method, the unknown values are predicted by employing the values of similar users. In item-based methods, the unknown values are predicted by employing the values of similar items. Similarity calculation between items or users is an important part of neighborhood-based collaborative filtering. Multiple mechanisms such as Pearson Correlation and Vector Cosine based similarity are used for this purpose.

Model-based CF employs training data to find the user interest pattern and then predict the user's interest in the

items that have not been accessed [22][23][24][25]. These approaches are mainly implemented by matrix factorization. For example, the work [25] detailed some important matrix factorization methods, however it can be only used for two-dimensional matrixes. This kind of CF handles the sparse data better than memory based ones. It is good for processing large scale data set and improves the prediction performance. It can present intuitive rationale for recommendations. But model building in model-based CF is very time-consuming.

Although CF based prediction methods have achieved great success, they are used mainly for the traditional user-service two dimensional QoS data, which means that user-context, time, and location are not considered. Therefore, many methods using some special characteristics of these three factors have been proposed [4][5][7][8][12][26]. However, all of the above methods do not consider all the dimensions of QoS data in an integrated manner, hence the integral structure of the high dimensional QoS data is neglected, which results in some disadvantages: (1) the integral structure of the high dimensional QoS data can be used for QoS prediction, while It is a pity that the integral structure is not employed so far; (2) the kind of QoS prediction methods using some characteristics special for one dimension are difficult to be reused for the other dimensions, therefore the prediction methods have to be complex due to considering the characteristics of all QoS dimensions; and (3) if some new dimensions occur owing to the development of network technology, we have to design new prediction methods. Therefore we need a new QoS prediction approach considering all QoS dimensions wholly and uniformly to predict multi-dimensional QoS accurately and easily, which was proposed in this paper.

One main hindrance to validating prediction methods is the lack of real-world QoS datasets, especially the ones containing time and location data. Therefore, we have utilized the dataset released by Zheng et al. [8] containing time information, and proposed a new BTP-based credible QoS dataset containing both time and location data.

6 Conclusion and Future Work

We have utilized tensor to predict unknown QoS values in a high dimensional spatiotemporal space, considering all QoS dimensions in an integrated manner to predict multi-dimensional QoS accurately and easily. Our proposed approach first models multi-dimensional QoS data as a tensor, then finds out its component matrices by decomposing this QoS tensor. These component matrices allow us to reconstruct the QoS tensor accurately. Finally, the reconstructed tensor tells us the prediction of the unknown values of QoS data. Different from related work, our proposed methods make full use of the integral structure of multi-dimensional QoS data, which can be used to predict any dimensional QoS data easily and accurately.

The proposed HDOP approach is essentially a multi-dimensional information oriented model-based CF (collaborative filtering), since it works based on tensor decomposition instead of matrix decomposition. Using another kind of CF, e.g., neighborhood-based CF, to make QoS predictions has been studied in our previous work [32][33] in depth; however the method proposed in that previous work is not multi-dimensional information oriented.

For the development of Internetware [34], our proposed approach can be encapsulated as a API that is invoked by the Internetware system [35][36]. Described as Algorithm 1 in Section 2.2, the inputs of our proposed approach are tensors which are essentially multi-dimensional arrays. In fact, a multi-dimensional array is a basic data structure for many program language such as Java and python. Hence, after being coded, the proposed HDOP algorithm can be directly used for QoS prediction in Internetware systems. In our future work, we will establish a big distributed computation infrastructure to enhance the prediction performance in term of accuracy or time cost. Moreover, we will devote ourselves to the problem of high dimensional information oriented neighborhood-based CF, which will require us to study high dimensional information oriented similarity computing as well as missing value prediction based on such type of similarities.

Acknowledgments

The work was supported by the NSFC (61472047), the Fundamental Research Funds for the Central Universities (2014ZD01), and the Open Research Fund Program of Beijing Key Laboratory on Integration and Analysis of Large-scale Stream Data.

References

[1] Y. Ma, S. Wang, F. Yang, and R. N. Chang, "Predicting QoS Values via Multi-dimensional QoS Data for Web Service Recommendations," *in Proc. 22th Int. Conf. Web Services (ICWS'15)*, pp. 249-256, 2015.

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

[3] A. Jaiswal, W. Peng, and T. Sun, "Predicting time-sensitive user locations from social media," *in Proc. Int. Conf. Advances in Social Networks Analysis and Mining (ASONAM'13),* pp. 870-877, 2013.

[4] J. Wang, C. Zeng, C. He, L. Hong, L. Zhou, R. K. Wong, and J. Tian, "Context-aware role mining for mobile service recommendation," *in Proc. 27th Annual ACM Symposium on Applied Computing (SAC'12),* pp. 173-178, 2012.

[5] R. K. Wong, V. W. Chu, and T. Hao, "Online role mining for context-aware mobile service recommendation," *Personal and Ubiquitous Computing*, vol. 18, no.5, pp. 1029-1046, 2014.

[6] H. Zhu, E. Chen, H. Xiong, K. Yu, H. Cao, and J. Tian, "Mining Mobile User Preferences for Personalized Context-Aware

Recommendation," *ACM Trans, Intelligent Systems and Technology*, vol. 5, article no. 58, 2014.
[7] M. Tang, Y. Jiang, J. Liu, and X. Liu, "Location-aware collaborative filtering for QoS-based service recommendation," *in Proc. 19th Int. Conf. Web Services (ICWS'12),* pp. 202-209, 2012.
[8] Y. Zhang, Z. Zheng, and M. R. Lyu, "WSPred: A time-aware personalized QoS prediction framework for Web services," *in Proc. 22nd Symposium onSoftware Reliability Engineering (ISRE'11),* pp. 210-219, 2011.
[9] X. Chen, Z. Zheng, and M. R. Lyu, "QoS-Aware Web Service Recommendation via Collaborative Filtering," Web Services Foundations, ed: Springer, pp. 563-588, 2014
[10] S. Ran, "A model for Web services discovery with QoS," *ACM Sigecom exchanges*, vol. 4, no.1, pp. 1-10, 2003.
[11] M. Zhang, X. Liu, R. Zhang, and H. Sun, "A Web service recommendation approach based on qos prediction using fuzzy clustering," *in Proc. 9th Int. Conf. Services Computing (SCC'12),* pp. 138-145, 2012.
[12] A. Amin, A. Colman, and L. Grunske, "An approach to forecasting QoS attributes of Web services based on ARIMA and GARCH models," *in Proc. 19th Int. Conf.* Web Services *(ICWS'12)*, pp. 74-81, 2012.
[13] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative Web service qos prediction via neighborhood integrated matrix factorization," *IEEE Trans. Services Computing*, vol. 6, no.3, pp. 289-299, 2013.
[14] T. G. Kolda and B. W. Bader, "*Tensor Decompositions and Applications,*" SIAM review, vol. 51, no. 3, pp. 455-500, 2009.
[15] C. Igel and M. Hüsken, "Improving the Rprop learning algorithm," *in Proc. 2nd Int. ICSC symposium on neural computation (NC'2000)*, pp. 115-121, 2000.
[16] W. Lo, J. Yin, S. Deng, Y. Li, and Z. Wu, "An extended matrix factorization approach for qos prediction in service selection," *in Proc. 9th Int. Conf. Services Computing (SCC'12)*, pp. 162-169, 2012.
[17] M. G. Kang and A. K. Katsaggelos, "General choice of the regularization functional in regularized image restoration," *IEEE Trans. Image Processing*, vol. 4, no. 5, pp. 594-602, 1995.
[18] A. K. Katsaggelos, J. Biemond, R. W. Schafer, and R. M. Mersereau, "A regularized iterative image restoration algorithm," *IEEE Trans. Signal Processing*, vol. 39, no. 4, pp. 914-929, 1991.
[19] K. Miller, "Least squares methods for ill-posed problems with a prescribed bound," *SIAM Journal on Mathematical Analysis*, vol. 1, no. 1, pp. 52-74, 1970.
[20] P. Comon, J. M. ten Berge, L. De Lathauwer, and J. Castaing, "Generic and typical ranks of multi-way arrays," *Linear Algebra and its Applications*, vol. 430, no. 11, pp. 2997-3007, 2009.
[21] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun, "Personalized qos-aware Web service recommendation and visualization," *IEEE Trans. Services Computing*, vol. 6, no. 1, pp. 35-47, 2013.
[22] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Trans. Information Systems*, vol. 22, no. 1, pp. 89-115, 2004.
[23] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," Advances in artificial intelligence, vol. 2009, article no. 4, 2009.
[24] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel, "Probabilistic memory-based collaborative filtering," *IEEE Trans. Knowledge and Data Engineering*, vol. 16, no. 1, pp. 56-69, 2004.
[25] S. Rendle, "Factorization Machines with libFM," ACM Trans. Intell. Syst. Technol., 3, vol. 3, pp. 1-22, 2012.
[26] Y. Xu, J. Yin, and W. Lo, "A Unified Framework of QoS-Based Web Service Recommendation with Neighborhood-Extended Matrix Factorization," *in Proc. 6th Int. Conf. Service-Oriented Computing and Applications (SOCA'13)*, pp. 198-205, 2013.
[27] Igel C, Hüsken M. "Empirical evaluation of the improved Rprop learning algorithms". *Neurocomputing*, vol. 50, pp. 105-123, January, 2003.
[28] M. Riedmiller and H. Braun. "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," *in Proc. Int. Conf. Neural Networks*, pp. 586–591, 1993.
[29] M. Riedmiller, "Advanced supervised learning in multi-layer perceptrons – from backpropagation to adaptive learning algorithms," *Computer Standards and Interfaces*, vol. 16, no. 5, pp. 265–278, 1994.
[30] W. Schiffmann, M. Joost, and R. Werner. "Comparison of optimized backpropagation algorithms," *in Proc. European Symposium on Artificial Neural Networks (ESANN'93)*, pp. 97–104, 1993.
[31] M. Joost and W. Schiffmann, "Speeding up backpropagation algorithms by using cross–entropy combined with pattern normalization," *International Journal of Uncertainty, Fuzziness and Knowledge–based Systems*, vol. 6, no. 2, pp. 117–126, 1998.
[32] Ma Y, Wang S, Hung PCK, Hsu C-H, Sun Q, Yang F., "A Highly Accurate Prediction Algorithm for Unknown Web Service QoS Value," *IEEE Transactions on Services Computing*, 2015,PP(99):1-14. DOI: 10.1109/TSC.2015.2407877.
[33] S. Wang, Z. Zheng, Z. Wu, M. Lyu, and F. Yang, "Reputation Measurement and Malicious Feedback Rating Prevention in Web Service Recommendation Systems," IEEE Transactions on Services Computing, vol. 8, no. 6, pp. 755 - 767, 2015.
[34] Hong Mei, Gang Huang, Tao Xie, "Internetware: A Software Paradigm for Internet Computing," *IEEE Computer*, vol. 45, no. 6, pp. 26-31, 2012.
[35] Xuanzhe Liu, Yun Ma, Gang Huang, Junfeng Zhao, Hong Mei, Yunxin Liu, "Data-Driven Composition for Service-Oriented Situational Web Applications," *IEEE Trans. Services Computing*, vol. 8, no. 1, pp. 2-16, 2015.
[36] Yun Ma, Xuan Lu, Xuanzhe Liu, Xudong Wang, M. Brian Blake, "Data-driven synthesis of multiple recommendation patterns to create situational Web mashups," *SCIENCE CHINA Information Sciences*, vol. 56, no. 8, pp. 1-16, 2013.

Shangguang Wang is an associate professor at the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications (BUPT). He received his Ph.D. degree at BUPT in 2011. He is 2015-2016 President of the Service Society Young Scientist Forum in China, General Chair of CollaborateCom 2016, General Chair of ICCSA 2016, Application Track Chair of IEEE SCC 2015, Workshop Chair of AMC/IEEE UCC 2016, Program Chair of IOV 2014, and Program Chair of SC2 2014. He has published more than 50 journal/conference papers such as IEEE TSC, ACM TOMM, IEEE TCC, IEEE TETC, and IEEE TASE. His research interests include Service Computing, Cloud Services, and QoS Management. He is a Senior Member of the IEEE..

You Ma is an assistant research scientist at the National Satellite Meteorological Center (NSMC), Beijing, China. He received his Ph.D. degree at BUPT in 2015. His research interests are in service computing, recommender systems, and satellite data assimilation

Bo Cheng received the PhD degree in computer science and technology from the University of Electronics Science and Technology of China in 2006. His research interests include multimedia communications, and services computing. Currently, he is an associate professor of state key laboratory of networking and switching technology of Beijing University of posts and telecommunications.

Fangchun Yang received his Ph.D. degree in communication and electronic system from the Beijing University of Posts and Telecommunication in 1990. He is currently a professor at the Beijing University of Posts and Telecommunication, China. His research interests include network intelligenceand communications software.He is a fellow of the IET.

Rong Chang received his PhD degree in computer science and engineering from the University of Michigan in 1990. He is with IBM Research leading a global team creating innovative IoT cloud services technologies. He holds 30+ patents and has published 40+ papers. He is Member of IBM Academy of Technology, ACM Distinguished Engineer, Chair of IEEE Computer Society Technical Committee on Services Computing, Editor-in-Chief of the International Journal of Cloud Computing and Associate Editor-in-Chief of the IEEE Transactions on Services Computing.