Temporal Pattern-Aware QoS Prediction via Biased Non-Negative Latent Factorization of Tensors

Xin Luo, *Senior Member, IEEE*, Hao Wu, Huaqiang Yuan, and MengChu Zhou, *Fellow, IEEE*

***Abstract*—Quality-of-service (QoS) data vary over time, making it vital to capture the temporal patterns hidden in such dynamic data for predicting missing ones with high accuracy. However, currently latent factor (LF) analysis-based QoS-predictors are mostly defined on static QoS data without the consideration of such temporal dynamics. To address this issue, this paper presents a biased non-negative latent factorization of tensors (BNLFTs) model for temporal pattern-aware QoS prediction. Its main idea is fourfold: 1) incorporating linear biases into the model for describing QoS fluctuations; 2) constraining the model to be non-negative for describing QoS non-negativity; 3) deducing a single LF-dependent, non-negative, and multiplicative update scheme for training the model; and 4) incorporating an alternating direction method into the model for faster convergence. The empirical studies on two dynamic QoS datasets from real applications show that compared with the state-of-the-art QoS-predictors, BNLFT represents temporal patterns more precisely with high computational efficiency, thereby achieving the most accurate predictions for missing QoS data.**

***Index Terms*—Latent factor analysis (LFA), latent factorization of tensor, learning temporal pattern, linear bias (LB), non-negative latent factorization of tensor, non-negativity constraint, quality-of-service (QoS) prediction.**

Manuscript received January 13, 2019; revised February 24, 2019; accepted March 2, 2019. Date of publication April 4, 2019; date of current version April 15, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61772493, Grant 91646114, and Grant 61602352, in part by the Chongqing Research Program of Technology Innovation and Application under Grant cstc2017rgzn-zdyfX0020, Grant cstc2017zdcy-zdyf0554, and Grant cstc2017rgzn-zdyf0118, in part by the Chongqing Cultivation Program of Innovation and Entrepreneurship Demonstration Group under Grant cstc2017kjrc-cxcytd0149, in part by the Chongqing Overseas Scholars Innovation Program under Grant cx2017012 and Grant cx2018011, and in part by the Pioneer Hundred Talents Program of Chinese Academy of Sciences. This paper was recommended by Associate Editor D. Tao. *(Xin Luo and Hao Wu are co-first authors.) (Corresponding authors: Huaqiang Yuan; MengChu Zhou.)*

X. Luo is with the School of Computer Science and Technology, Dongguan University of Technology, Dongguan, Guangdong 523808, China, also with the Chongqing Engineering Research Center of Big Data Application for Smart Cities, Chinese Academy of Sciences, Chongqing 400714, China, and also with the Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China (e-mail: luoxin21@gmail.com).

H. Wu is with the Chongqing Engineering Research Center of Big Data Application for Smart Cities, Chinese Academy of Sciences, Chongqing 400714, China, and also with the Chongqing Key Laboratory of Big Data and Intelligent Computing, Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing 400714, China (e-mail: wuhaoxy@cigit.ac.cn).

H. Yuan is with the School of Computer Science and Technology, Dongguan University of Technology, Dongguan 523808, China (e-mail: yuanhq@dgut.edu.cn).

M. Zhou is with the Institute of Systems Engineering, Macau University of Science and Technology, Macau 999078, China and also with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2019.2903736

I. Introduction

With the rapid development of cloud computing [1], [2], numerous Web services with functional equivalence are provided by different service providers. Hence, how to select the most suitable service from a huge set of candidates becomes a hot yet thorny issue [3]–[6]. Since a Web service can be judged via its quality-of-service (QoS) and historical QoS, data are vital for users to decide whether a Web service is suitable to invoke for their specific purposes. Therefore, QoS-based approaches to Web service selection were carefully investigated during the last decade [7]–[9].

It is well known that QoS can be measured from either service-providers' perspective like price and invoking ports, or users' perspective like throughput and response time [15], [16]. Commonly, user-perspective QoS data reflect more individual characteristics due to the following reasons: 1) a network environment and service statuses are changing over time, thus leading to temporal variation and 2) invoking locations frequently change, thus resulting in spatial variation. Hence, user-perspective QoS data are vital in Web service selection. A straightforward approach to acquire user-perspective QoS data is taking real-world service evaluations, which suffers two critical issues: 1) most commercial services charge for invocation, making such tests highly expensive and 2) with the explosively growing number of Web services, it costs a lot of time to evaluate all services in a candidate set [5], [10], [11]. Hence, it is vital to predict unknown QoS data from the user perspective accurately.

Various approaches are proposed to address this issue, and latent factor analysis (LFA)-based ones [7], [8], [17]–[19] are becoming increasingly attractive. An LFA-based QoS-predictor takes a user–service QoS matrix as the fundamental data source [18]–[23], where each row denotes a specified user, each column denotes a specified service, and each element denotes the QoS record of a certain metric experienced by a specified user on a specified service. Obviously, a user cannot invoke all candidate services or a service cannot be touched by all users, making this matrix highly sparse with numerous unknown yet desired data. Hence, an LFA-based QoS-predictor tries to handle this sparse user–service matrix, and extract its essential factors for predicting its unobserved data accurately.

To do so, an LFA-based QoS-predictor maps both users and services into the same low-dimensional latent factor (LF) space to build a low-rank approximation to the target user–service matrix for predicting its missing data. Such a process is defined on the known data of a target matrix only and implemented through various optimization techniques [5], [8], [12]–[14], [18]–[21]. The resulting approximation matrix can be full where the missing data are estimated based on the information hidden in the known ones. With a well-established learning objective and carefully chosen learning algorithms, an LFA-based QoS-predictor can predict missing QoS data accurately with high computational efficiency.

Lo *et al.* [18] proposed extending an LFA-based QoS-predictor with the consideration of location information in each historical QoS record. Zheng *et al.* [19] proposed an LFA-based QoS-predictor by considering the neighborhood information. Nonetheless, it costs a lot of memory to store the pair-wise neighborhood factor of each user/service in real applications. Luo *et al.* [7] proposed ensembling a set of diversified non-negative LFA-based QoS-predictors to achieve a highly accurate QoS-predictor. To improve the training efficiency of the resulting model, they further propose an ensemble of alternating direction method (ADM)-based LFA models [5], where the training process of each single model is accelerated by following the principle of an alternating direction method [24].

The aforementioned LFA-based QoS-predictors, despite their efficiency, are static models without the consideration of temporal variations of QoS data. Note that in real applications, QoS data of the same service can vary with time. Hence, it becomes a vital task to explore LFA-based QoS-predictors aware of temporal dynamics hidden in the QoS data. Koren [25] proposed the timeSVD++ model, which models the temporal effects as additional LFs based on the time slot of given data. Sahoo *et al.* [26] proposed modeling the dynamic transition of the target data with a hidden Markov model. These models can incorporate part of the temporal information into existing LF models, but the resulting model is still static.

To implement a temporal pattern-aware QoS-predictor that is able to accurately capture temporal dynamics in QoS data, this paper innovatively proposes a biased non-negative latent factorization of tensors (BNLFTs) model. It adopts a tensor to represent time-varying QoS data, thereby naturally taking the temporal features into consideration. It further models the linear bias (LB) on users, services, and time points to address constant fluctuations of QoS data, thereby achieving a robust model with high prediction accuracy for missing QoS data. The main contributions of this paper include the following:

1) a BNLFT model that extracts temporal LFs from dynamic QoS data for predicting missing ones accurately;
2) an effective training scheme for BNLFT, which integrates the principle of single LF-dependent, non-negative, and multiplicative update (SLF-NMU) and ADM into its training process;
3) detailed algorithm design and analysis of a BNLFT model.

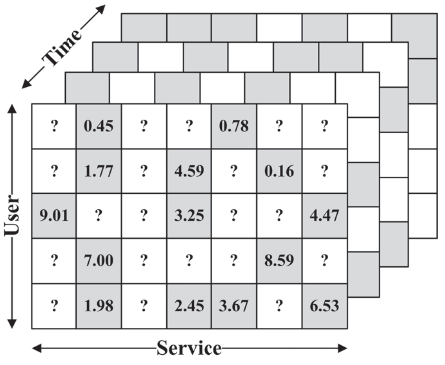


Fig. 1. QoS tensor of user–service–time showing the response time.

Section II gives the preliminaries. Section III presents our methods. Section IV reports the experimental results. Finally, Section V concludes this paper.

II. Preliminaries

A. Notations

When performing temporal pattern-aware QoS analysis and prediction, a user–service–time tensor is taken as a fundamental input data source. Naturally, since QoS data are defined on the non-negative field of real numbers, such a tensor is also filled with non-negative data, as depicted in Fig. 1. Table I gives the necessary symbols for performing latent factorization of a user–service–time tensor. Note that as analyzed in Section I, the target tensor $\mathcal{Y}$ is usually high dimensional and sparse (HiDS), and contains numerous unknown entries due to unobserved QoS data. Thus, we first define our target tensor.

Definition 1 (HiDS User–Service–Time Tensor): Given I, J, and K, $\mathcal{Y}^{|I|\times|J|\times|K|}$ is a user–service–time tensor where element y_{ijk} denotes the QoS experienced by user $i \in I$ on service $j \in J$ at time point $k \in K$. Given $\mathcal{Y}$'s known and unknown element sets Λ and Γ, $\mathcal{Y}$ is HiDS if $|\Lambda| \ll |\Gamma|$.

B. Problem Formulation

Note that this paper performs the latent factorization of tenors (LFTs) on $\mathcal{Y}$ in the way of canonical polyadic factorization (CPF), which is actually a special case of Tucker factorization (TF). Compared to CPF, TF considers the elements of a core tensor; therefore, achieving higher representation ability for a target tensor than CPF does. However, TF consumes much more time than CPF does by noting the additional time cost proportional to the size of a core tensor [27], [28]. Meanwhile, the mathematical form of CPF is much more concise than that of TF [27], [28]. Hence, this paper adopts CPF as a basic scheme to deduce a BNLFT model. However, we plan to try TF in the future.

With CPTF, $\mathcal{Y}$ is decomposed into R rank-one tensors, that is, $\mathcal{A}_1$, $\mathcal{A}_2, \ldots,$ and $\mathcal{A}_R$, where R is the rank of the resulting approximation $\hat{\mathcal{Y}}$ to $\mathcal{Y}$. Note that to identify R on a given tensor is a nondeterministic polynomial problem. Hence, it is commonly predefined.

TABLE I
ADOPTED SYMBOLS AND THEIR DESCRIPTIONS

Symbol	Description
$\boldsymbol{\mathcal{Y}}$	Target three dimensional user-service-time tensor.
$\boldsymbol{\mathcal{A}}$	Rank-one tensor of LFs.
$\boldsymbol{\mathcal{L}}$	Rank-one tensor of LBs.
I, J, K	Sets of users, services and time points.
y_{ijk}, a_{ijk}, l_{ijk}	Single elements in $\boldsymbol{\mathcal{Y}}$, $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{L}}$, respectively.
$\hat{\boldsymbol{\mathcal{Y}}}$	Low-rank approximation to $\boldsymbol{\mathcal{Y}}$.
R	Rank of $\hat{\boldsymbol{\mathcal{Y}}}$; also denotes the dimension of the LF space.
U, **S**, **T**	LF matrices for user, service and time.
D, **E**, **F**	LB matrices for user, service and time.
$\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$	LB vectors for user, service and time obtained by folding LB matrices **D**, **E** and **F**.
$\boldsymbol{u_r}$, $\boldsymbol{s_r}$, $\boldsymbol{t_r}$	LF vectors for the rth rank-one tensor in $\hat{\boldsymbol{\mathcal{Y}}}$.
u_{ir}, s_{ir}, t_{ir}	Single LFs in $\boldsymbol{u_r}$, $\boldsymbol{s_r}$ and $\boldsymbol{t_r}$.
$\mathbb{R}$	Real number domain.
$\circ$	Computes the outer product of two vectors.
$\lVert\cdot\rVert_F$	Computing the Frobenius norm of an enclosed matrix or tensor.
$\lvert\cdot\rvert$	Computing the cardinality of an enclosed set.
Λ	Known entry set of $\boldsymbol{\mathcal{Y}}$.
Γ	Unknown entry set of $\boldsymbol{\mathcal{Y}}$.
n	Training iteration count.

Definition 2 (Rank-One Tensor): $\boldsymbol{\mathcal{A}}_r^{|I|\times|J|\times|K|}$ is a rank-one tensor if it can be written as the outer product of three LF vectors $\boldsymbol{u_r}$, $\boldsymbol{s_r}$, and $\boldsymbol{t_r}$ as $\boldsymbol{\mathcal{A}}_r = \boldsymbol{u}_r \circ \boldsymbol{s}_r \circ \boldsymbol{t}_r$.

Note that $\boldsymbol{u_r}$, $\boldsymbol{s_r}$, and $\boldsymbol{t_r}$ are of length $|I|$, $|J|$, and $|K|$, respectively. By unfolding their outer product in Definition 1, we achieve the detailed expression of each element a_{ijk} in $\boldsymbol{\mathcal{A}}_r$

$$a_{ijk} = u_{ir}s_{jr}t_{kr}. \tag{1}$$

Therefore, LF matrices $\mathbf{U}^{|I|\times R}$, $\mathbf{S}^{|J|\times R}$, and $\mathbf{T}^{|K|\times R}$, respectively, consist of R LF vectors with length $|I|$, $|J|$, and $|K|$, where the rth column vector of **U**, **S**, and **T** is coupled to form the rth rank-one tensor $\boldsymbol{\mathcal{A}}_r$, as shown in Fig. 2. With these R rank-one tensors, that is, $\{\boldsymbol{\mathcal{A}}_r | r \in \{1, 2, \ldots, R\}\}$, we achieve $\boldsymbol{\mathcal{Y}}$'s rank-R approximation $\hat{\boldsymbol{\mathcal{Y}}}$ as follows:

$$\hat{\boldsymbol{\mathcal{Y}}} = \sum_{r=1}^{R} \boldsymbol{\mathcal{A}}_r \tag{2}$$

where each element y_{ijk} is formulated as

$$\hat{y}_{ijk} = \sum_{r=1}^{R} u_{ir}s_{jr}t_{kr}. \tag{3}$$

To achieve the desired LF matrices **U**, **S**, and **T**, we set an objective function measuring the difference between $\boldsymbol{\mathcal{Y}}$ and $\hat{\boldsymbol{\mathcal{Y}}}$. In this paper, we choose the Euclidean distance to model such a difference. But our proposed method is also compatible with other objective functions like L_P-norms or β-distances. With it, the objective function is given by

$$\varepsilon = \left\lVert \boldsymbol{\mathcal{Y}} - \hat{\boldsymbol{\mathcal{Y}}} \right\rVert_F^2 \tag{4}$$

where ε denotes the objective function and the operator $\lVert\cdot\rVert_F$ computes the Frobenius norm of the enclosed tensor. Note

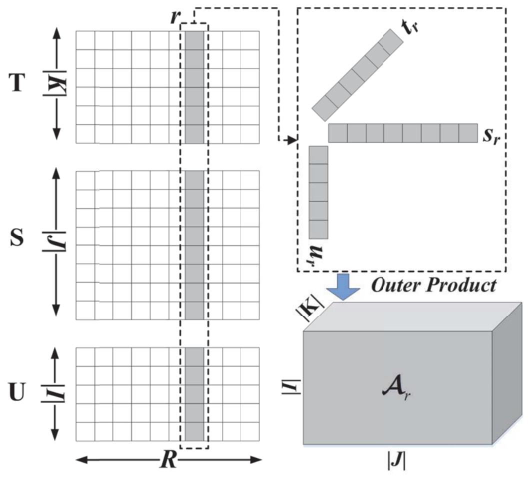


Fig. 2. LF matrices and rank-one tensor of LFs in CP-style TF.

that the Frobenius norm of a tensor $(\boldsymbol{\mathcal{Y}} - \hat{\boldsymbol{\mathcal{Y}}}) \in \mathbb{R}^{|I|\times|J|\times|K|}$ is formulated as [27]

$$\left\lVert \boldsymbol{\mathcal{Y}} - \hat{\boldsymbol{\mathcal{Y}}} \right\rVert_F = \sqrt{\sum_{i=1}^{|I|}\sum_{j=1}^{|J|}\sum_{k=1}^{|K|} \left(y_{ijk} - \hat{y}_{ijk}\right)^2}$$
$$= \sqrt{\sum_{i=1}^{|I|}\sum_{j=1}^{|J|}\sum_{k=1}^{|K|} \left(y_{ijk} - \sum_{r=1}^{R} u_{ir}s_{jr}t_{kr}\right)^2}. \tag{5}$$

Thus, (4) is reformulated into

$$\varepsilon = \sum_{i=1}^{|I|}\sum_{j=1}^{|J|}\sum_{k=1}^{|K|} \left(y_{ijk} - \sum_{r=1}^{R} u_{ir}s_{jr}t_{kr}\right)^2. \tag{6}$$

As analyzed in Section I and shown in Fig. 1, most entries of $\boldsymbol{\mathcal{Y}}$ are unknown. Hence, the objective function (6) should be defined on Λ to describe known information in $\boldsymbol{\mathcal{Y}}$ more precisely. Following such a principle, (6) is reformulated into

$$\varepsilon = \sum_{y_{ijk}\in\Lambda} \left(y_{ijk} - \sum_{r=1}^{R} u_{ir}s_{jr}t_{kr}\right)^2. \tag{7}$$

Naturally, (7) is ill-posed due to the imbalanced distribution of $\boldsymbol{\mathcal{Y}}$'s known data and its sensitivity to initial hypotheses of **U**, **S**, and **T**. Hence, it is essential to incorporate the Tikhonov regularization into it for generalizing the model, yielding

$$\varepsilon = \sum_{y_{ijk}\in\Lambda} \left(\left(y_{ijk} - \sum_{r=1}^{R} u_{ir}s_{jr}t_{kr}\right)^2 + \lambda \sum_{r=1}^{R}\left(u_{ir}^2 + s_{jr}^2 + t_{kr}^2\right)\right). \tag{8}$$

Note that we make the regularization terms connect to each instant loss on a single instance to make them correctly describe the known data density related to each LF in **U**, **S**, and **T**, following the strategies mentioned in [29]–[33]. With such a design, LFs related to more known data are subject to stronger regularization effects, and vice-versa. Finally, since QoS data are defined on the non-negative field of real numbers, it is necessary to constrain **U**, **S**, and **T** to be non-negative

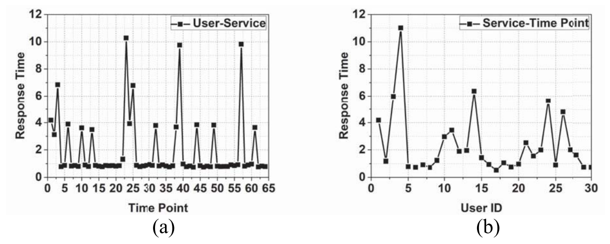


Fig. 3. QoS flucturation. (a) Response time of a specified service experienced by a specified user at different time points. (b) Response time of a specified service experienced by different users at the same time point.

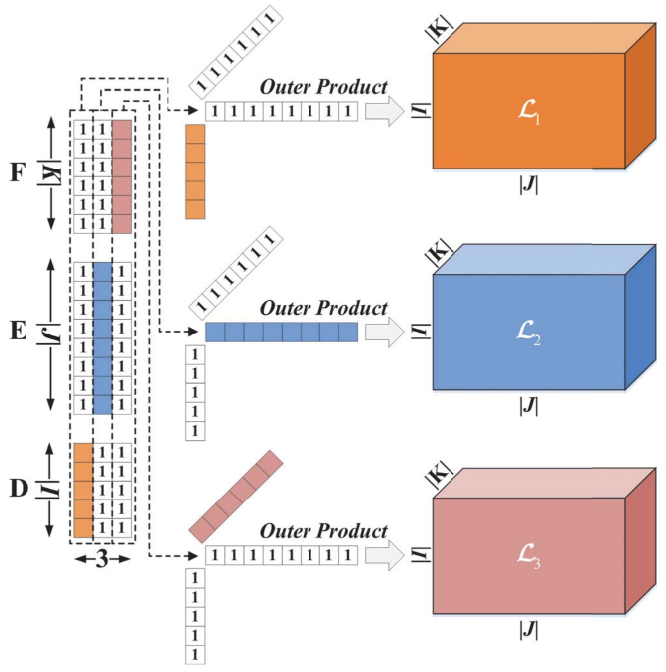


Fig. 4. LB matrices and rank-one tensor of LBs in CP-style TF.

to correctly describe such non-negative data. Thus, we achieve our problem formulation as follows:

$$\varepsilon = \sum_{y_{ijk}\in\Lambda}\left(\left(y_{ijk} - \sum_{r=1}^{R} u_{ir}s_{jr}t_{kr}\right)^2 + \lambda\sum_{r=1}^{R}\left(u_{ir}^2 + s_{jr}^2 + t_{kr}^2\right)\right)$$
$$\text{s.t.}\quad \forall i \in I, j \in J, k \in K, r \in \{1,2,\ldots,R\}:$$
$$u_{ir} \geq 0, s_{jr} \geq 0, t_{kr} \geq 0. \tag{9}$$

III. Temporal Pattern-Aware QoS Prediction

A. Linear Bias Modeling in Non-Negative LFT

QoS data fluctuate with time, as shown in Fig. 3. According to prior research [34]–[37], when analyzing such fluctuating data, it is essential to incorporate LBs into a learning model to enhance its stability and representation ability. Although LFT on an HiDS tensor is defined on a higher-dimensional solution space than LFA on an HiDS matrix [38]–[43], it is promising to improve an LFT model's performance by adopting similar strategies. Following this, we, respectively, introduce LBs corresponding to users, services, and time points into the problem. Fig. 4 illustrates such designs.

As shown in Fig. 4, in our context, LBs in LFT rely on three LB matrices, that is, $D^{|I|\times 3}$ for users, $E^{|J|\times 3}$ for services, and $F^{|K|\times 3}$ for time points. For each matrix, it has only one column filled with valued LBs but the other two are filled with the constant 1's. Similar to the original CPTF as shown in Fig. 2, we also take the LB vectors at the same dimension to form rank-one tensors of LBs. However, elements in each rank-one tensor of LBs only vary with the "active" dimension, that is, the dimension corresponding to valued LBs. For instance, in $\mathcal{L}_1$, for a specified user $i \in I$, we have $\forall j \in J, k \in K : l_{ijk}^1 = d_{i1}$. The same rule is also applied to $\mathcal{L}_2$ and $\mathcal{L}_3$.

By combining LBs shown in Fig. 4 with LFs shown in Fig. 2, we achieve $\mathcal{Y}$'s rank-R approximation $\hat{\mathcal{Y}}$ with LBs as follows:

$$\hat{\mathcal{Y}} = \sum_{r=1}^{R}\mathcal{A}_r + \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3. \tag{10}$$

Note that each element $\hat{y}_{ijk}$ in (10) is formulated as

$$\hat{y}_{ijk} = \sum_{r=1}^{R} u_{ir}s_{jr}t_{kr} + d_{i1} + e_{j2} + f_{k3}. \tag{11}$$

Since D, E, and F in (11) all have one active dimension only with the other dimension fixed, we fold LB matrices $\mathbf{D}$, $\mathbf{E}$, and $\mathbf{F}$ into LB vectors $\boldsymbol{a}^{|I|}$ for $\mathbf{D}$, $\boldsymbol{b}^{|J|}$ for $\mathbf{E}$, and $\boldsymbol{c}^{|K|}$ for $\mathbf{F}$ with the following mapping rule:

$$\forall i \in I : a_i = d_{i1}, \forall j \in J : b_j = e_{j2}, \forall k \in K : c_k = f_{k3}. \tag{12}$$

Thus, (11) is reformulated into

$$\hat{y}_{ijk} = \sum_{r=1}^{R} u_{ir}s_{jr}t_{kr} + a_i + b_j + c_k. \tag{13}$$

By substituting (13) into (9), we have the following objective function for performing LFT on $\mathcal{Y}$:

$$\varepsilon = \sum_{y_{ijk}\in\Lambda}\left(\left(y_{ijk} - \sum_{r=1}^{R} u_{ir}s_{jr}t_{kr} - a_i - b_j - c_k\right)^2 + \lambda\sum_{r=1}^{R}\left(u_{ir}^2 + s_{jr}^2 + t_{kr}^2\right) + \lambda_b\left(a_i^2 + b_j^2 + c_k^2\right)\right)$$
$$\text{s.t. } \forall i \in I, j \in J, k \in K, r \in \{1,2,\ldots,R\}:$$
$$u_{ir} \geq 0, s_{jr} \geq 0, t_{kr} \geq 0, a_i \geq 0, b_j \geq 0, c_k \geq 0 \tag{14}$$

where λ_b is the regularizing constant for LBs. Note that the regularization on LBs in (14) is also connected to each entry $y_{ijk} \in \Lambda$ to illustrate the sparsity of $\mathcal{Y}$.

B. Parameter Learning via SLF-NMU

As mentioned in previous sections, non-negativity is an essential characteristic of QoS data. Therefore, it is vital to constrain an LFT model's parameters to be non-negative to correctly describe non-negative QoS data. As indicated in prior research [9], [30], [44], [45], SLF-NMU is a highly efficient solver to implement non-negative LFA on an HiDS matrix. However, on an HiDS tensor, the parameter learning rule should be carefully redesigned.

Following the principle of SLF-NMU, we first apply the additive gradient descent (AGD) to (14) with respect to each single LF and LB to achieve the following learning rule:

$$(\mathbf{U},\mathbf{S},\mathbf{T},\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})=\underset{\mathbf{U},\mathbf{S},\mathbf{T},\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}}{\operatorname{argmin}}\ \varepsilon \overset{\text{AGD}}{\Rightarrow}\begin{cases} u_{ir}\leftarrow u_{ir}-\eta_{ir}\frac{\partial f}{\partial u_{ir}}=u_{ir}\\ \quad -\eta_{ir}\Big(\sum_{y_{ijk}\in\Lambda(i)}\big((y_{ijk}-\hat{y}_{ijk})(-s_{jr}t_{kr})+\lambda u_{ir}\big)\Big)\\ s_{jr}\leftarrow s_{jr}-\eta_{jr}\Big(\sum_{y_{ijk}\in\Lambda(j)}\big((y_{ijk}-\hat{y}_{ijk})(-u_{ir}t_{kr})+\lambda s_{jr}\big)\Big)\\ t_{kr}\leftarrow t_{kr}-\eta_{kr}\Big(\sum_{y_{ijk}\in\Lambda(k)}\big((y_{ijk}-\hat{y}_{ijk})(-u_{ir}s_{jr})+\lambda t_{kr}\big)\Big)\\ a_i\leftarrow a_i-\eta_i\frac{\partial f}{\partial a_i}=a_i-\eta_i\Big(\sum_{y_{ijk}\in\Lambda(i)}\big(-(y_{ijk}-\hat{y}_{ijk})+\lambda_b a_i\big)\Big)\\ b_j\leftarrow b_j-\eta_j\Big(\sum_{y_{ijk}\in\Lambda(j)}\big(-(y_{ijk}-\hat{y}_{ijk})+\lambda_b b_j\big)\Big)\\ c_k\leftarrow c_k-\eta_k\Big(\sum_{y_{ijk}\in\Lambda(k)}\big(-(y_{ijk}-\hat{y}_{ijk})+\lambda_b c_k\big)\Big)\end{cases}\tag{15}$$

where η_i, η_j, η_k, η_{ir}, η_{jr}, and η_{kr}, denote the learning rates for a_i, b_j, c_k, u_{ir}, s_{jr}, and t_{kr}, $\Lambda(i)$, $\Lambda(j)$, and $\Lambda(k)$ denote the subsets of Λ linked with entities $i\in I$, $j\in J$, and $k\in K$, respectively. To identify the negative components in the learning rule for each parameter, we reformulate (15) into

$$(\mathbf{U},\mathbf{S},\mathbf{T},\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})=\underset{\mathbf{U},\mathbf{S},\mathbf{T},\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}}{\operatorname{argmin}}\ \varepsilon \overset{\text{AGD}}{\Rightarrow}\begin{cases} u_{ir}\leftarrow u_{ir}+\eta_{ir}\sum\limits_{y_{ijk}\in\Lambda(i)}y_{ijk}s_{jr}t_{kr}\\ \quad -\eta_{ir}\sum\limits_{y_{ijk}\in\Lambda(i)}\big(\hat{y}_{ijk}s_{jr}t_{kr}+\lambda u_{ir}\big)\\ s_{jr}\leftarrow s_{jr}+\eta_{jr}\sum\limits_{y_{ijk}\in\Lambda(j)}y_{ijk}u_{ir}t_{kr}\\ \quad -\eta_{jr}\sum\limits_{y_{ijk}\in\Lambda(j)}\big(\hat{y}_{ijk}u_{ir}t_{kr}+\lambda s_{jr}\big)\\ t_{kr}\leftarrow t_{kr}+\eta_{kr}\sum\limits_{y_{ijk}\in\Lambda(k)}y_{ijk}u_{ir}s_{jr}\\ \quad -\eta_{kr}\sum\limits_{y_{ijk}\in\Lambda(k)}\big(\hat{y}_{ijk}u_{ir}s_{jr}+\lambda t_{kr}\big)\\ a_i\leftarrow a_i+\eta_i\sum\limits_{y_{ijk}\in\Lambda(i)}y_{ijk}-\eta_i\sum\limits_{y_{ijk}\in\Lambda(i)}\big(\hat{y}_{ijk}+\lambda_b a_i\big)\\ b_j\leftarrow b_j+\eta_j\sum\limits_{y_{ijk}\in\Lambda(j)}y_{ijk}-\eta_j\sum\limits_{y_{ijk}\in\Lambda(j)}\big(\hat{y}_{ijk}+\lambda_b b_j\big)\\ c_k\leftarrow c_k+\eta_k\sum\limits_{y_{ijk}\in\Lambda(k)}y_{ijk}-\eta_k\sum\limits_{y_{ijk}\in\Lambda(k)}\big(\hat{y}_{ijk}+\lambda_b c_k\big).\end{cases}\tag{16}$$

With (16), we clearly identify the negative components in the learning rule of each parameter, which can make the training results negative. Thus, following the principle of SLF-NMU, we manipulate the learning rates to cancel the negative terms in (16) with the initial status of the corresponding parameter. More specifically, we set η_i, η_j, η_k, η_{ir}, η_{jr}, and η_{kr} as follows:

$$\begin{cases}\eta_{ir}=u_{ir}/\sum_{y_{ijk}\in\Lambda(i)}\big(\hat{y}_{ijk}s_{jr}t_{kr}+\lambda u_{ir}\big)\\ \eta_{jr}=s_{jr}/\sum_{y_{ijk}\in\Lambda(j)}\big(\hat{y}_{ijk}u_{ir}t_{kr}+\lambda s_{jr}\big)\\ \eta_{kr}=t_{kr}/\sum_{y_{ijk}\in\Lambda(k)}\big(\hat{y}_{ijk}u_{ir}s_{jr}+\lambda t_{kr}\big)\\ \eta_i=a_i/\sum_{y_{ijk}\in\Lambda(i)}\big(\hat{y}_{ijk}+\lambda_b a_i\big)\\ \eta_j=b_j/\sum_{y_{ijk}\in\Lambda(j)}\big(\hat{y}_{ijk}+\lambda_b b_j\big)\\ \eta_k=c_k/\sum_{y_{ijk}\in\Lambda(k)}\big(\hat{y}_{ijk}+\lambda_b c_k\big).\end{cases}\tag{17}$$

By substituting (17) into (16), we obtain the multiplicative learning rule for each desired parameter in (14) as follows:

$$(\mathbf{U},\mathbf{S},\mathbf{T},\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})=\underset{\mathbf{U},\mathbf{S},\mathbf{T},\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}}{\operatorname{argmin}}\ \varepsilon \overset{\text{SLF-NMU}}{\Rightarrow}\begin{cases} u_{ir}\leftarrow u_{ir}\Big(\sum_{y_{ijk}\in\Lambda(i)}y_{ijk}s_{jr}t_{kr}\Big/\Big(\sum_{y_{ijk}\in\Lambda(i)}\hat{y}_{ijk}s_{jr}t_{kr}\\ \quad +\lambda|\Lambda(i)|u_{ir}\Big)\Big)\\ s_{jr}\leftarrow s_{jr}\Big(\sum_{y_{ijk}\in\Lambda(j)}y_{ijk}u_{ir}t_{kr}\Big/\Big(\sum_{y_{ijk}\in\Lambda(j)}\hat{y}_{ijk}u_{ir}t_{kr}\\ \quad +\lambda|\Lambda(j)|s_{jr}\Big)\Big)\\ t_{kr}\leftarrow t_{kr}\Big(\sum_{y_{ijk}\in\Lambda(k)}y_{ijk}u_{ir}s_{jr}\Big/\Big(\sum_{y_{ijk}\in\Lambda(k)}\hat{y}_{ijk}u_{ir}s_{jr}\\ \quad +\lambda|\Lambda(k)|t_{kr}\Big)\Big)\\ a_i\leftarrow a_i\Big(\sum_{y_{ijk}\in\Lambda(i)}y_{ijk}\Big/\Big(\sum_{y_{ijk}\in\Lambda(i)}\hat{y}_{ijk}+\lambda_b|\Lambda(i)|a_i\Big)\Big)\\ b_j\leftarrow b_j\Big(\sum_{y_{ijk}\in\Lambda(j)}y_{ijk}\Big/\Big(\sum_{y_{ijk}\in\Lambda(j)}\hat{y}_{ijk}+\lambda_b|\Lambda(j)|b_j\Big)\Big)\\ c_k\leftarrow c_k\Big(\sum_{y_{ijk}\in\Lambda(k)}y_{ijk}\Big/\Big(\sum_{y_{ijk}\in\Lambda(k)}\hat{y}_{ijk}+\lambda_b|\Lambda(k)|c_k\Big)\Big).\end{cases}\tag{18}$$

Note that if $\mathbf{U}$, $\mathbf{S}$, $\mathbf{T}$, $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ are initially non-negative, their non-negativity remains with the multiplicative learning rule given in (18).

C. Training Scheme With ADM-Principle

Performing LFT on an HiDS tensor like our user–service–time tensor is much more complex than LFA on a 2-D HiDS matrix [5], [46]: our learning space is much larger with many more decision parameters. Therefore, a solver can be easily trapped at some saddle points, making the model converge slowly or even fail to achieve any local optimum. As demonstrated in our prior research [5], [24], [46], a learning model's convergence rate can be improved by incorporating ADM into its training process with the following procedure: 1) decomposing the original task into multiple subtasks and 2) solving each subtask sequentially, where the solution to the current subtask depends on the solutions of previously addressed ones.

To achieve a highly efficient ADM-based training scheme, the original optimization task should be split into interdependent subtasks, that is, the status of decision parameters involved in one subtask can affect solutions for the others [24], [46]. By observing the learning objective (14) and parameter update rule (17), we have the following findings: 1) in each training iteration, LFs in the same LF matrices or LBs in the same LB vectors are inner-independent, for example, $\forall i, m\in I$, $r\in\{1,\ldots,R\}$, and the status of u_{ir} does not affect that of u_{lr} in the same training iteration with the parameter update rule (18) and 2) LFs/LBs in different LF matrices/LB vectors are interdependent, for example, the status of $\mathbf{U}$ affects those of $\mathbf{S}$, $\mathbf{T}$, $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$. Thus, to achieve a task sequence where one is solved based on the solution of the previously solved ones, we, respectively, find the solution process to $\mathbf{U}$, $\mathbf{S}$, $\mathbf{T}$, $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$. as separate subtasks in the nth training iteration

$$\forall i\in I, r=1\sim R, \mathbf{U}^{(n+1)}\overset{\text{SLF-NMU}}{\leftarrow}\underset{\mathbf{U}}{\operatorname{argmin}}\,\varepsilon \times\Big(\boldsymbol{a}^{(n)},\boldsymbol{b}^{(n)},\boldsymbol{c}^{(n)},\mathbf{U}^{(n)},\mathbf{S}^{(n)},\mathbf{T}^{(n)}\Big)$$

Algorithm 1 BNLFT

```
Input: $\Lambda, I, J, K, R, \lambda, \lambda_b$
Operation                                                                      Cost
1. Initialize $\mathbf{U}^{|I|\times R}$, $\mathbf{S}^{|J|\times R}$, $\mathbf{T}^{|K|\times R}$, $\boldsymbol{a}^{|I|}$, $\boldsymbol{b}^{|J|}$,        $\Theta((|I|+|J|+|K|)\times R)$
   $\boldsymbol{c}^{|K|}$ with non-negative random values in [0, 0.05].
   Initialize $n = 1$, $T = max\_itearation\_count$                            $\Theta(1)$
   while $n \le T$ and not converge do                                         $\times n$
2.   $\mathbf{U}^{(n+1)}$ = Update_U($\Lambda, I, R, \lambda, \mathbf{U}^{(n)}, \mathbf{S}^{(n)}$,              $\Theta$(Update_U)
       $\mathbf{T}^{(n)}, \boldsymbol{a}^{(n)}, \boldsymbol{b}^{(n)}, \boldsymbol{c}^{(n)}$)
3.   $\mathbf{S}^{(n+1)}$ = Update_S($\Lambda, J, R, \lambda, \mathbf{U}^{(n+1)}, \mathbf{S}^{(n)}$,            $\Theta$(Update_S)
       $\mathbf{T}^{(n)}, \boldsymbol{a}^{(n)}, \boldsymbol{b}^{(n)}, \boldsymbol{c}^{(n)}$)
4.   $\mathbf{T}^{(n+1)}$ = Update_T($\Lambda, K, R, \lambda, \mathbf{U}^{(n+1)}$,                     $\Theta$(Update_T)
       $\mathbf{S}^{(n+1)}, \mathbf{T}^{(n)}, \boldsymbol{a}^{(n)}, \boldsymbol{b}^{(n)}, \boldsymbol{c}^{(n)}$)
5.   $\boldsymbol{a}^{(n+1)}$ = Update_a($\Lambda, I, R, \lambda, \mathbf{U}^{(n+1)}, \mathbf{S}^{(n+1)}$,        $\Theta$(Update_a)
       $\mathbf{T}^{(n+1)}, \boldsymbol{a}^{(n)}, \boldsymbol{b}^{(n)}, \boldsymbol{c}^{(n)}$)
6.   $\boldsymbol{b}^{(n+1)}$ = Update_b($\Lambda, J, R, \lambda, \mathbf{U}^{(n+1)}, \mathbf{S}^{(n+)}$,         $\Theta$(Update_b)
       $\mathbf{T}^{(n+1)}, \boldsymbol{a}^{(n+1)}, \boldsymbol{b}^{(n)}, \boldsymbol{c}^{(n)}$)
7.   $\boldsymbol{c}^{(n+1)}$ = Update_c($\Lambda, K, R, \lambda, \mathbf{U}^{(n+1)}, \mathbf{S}^{(n+1)}$,        $\Theta$(Update_c)
       $\mathbf{T}^{(n+1)}, \boldsymbol{a}^{(n+1)}, \boldsymbol{b}^{(n+1)}, \boldsymbol{c}^{(n)}$)
   end while                                                                   --
Output: U, S, T, a, b, c
```

$$\forall j \in J, r = 1 \sim R, \mathbf{S}^{(n+1)} \overset{\text{SLF-NMU}}{\longleftarrow} \underset{\mathbf{S}}{\operatorname{argmin}}\, \varepsilon \times \left(\boldsymbol{a}^{(n)}, \boldsymbol{b}^{(n)}, \boldsymbol{c}^{(n)}, \mathbf{U}^{(n+1)}, \mathbf{S}^{(n)}, \mathbf{T}^{(n)}\right)$$
$$\forall k \in K, r = 1 \sim R, \mathbf{T}^{(n+1)} \overset{\text{SLF-NMU}}{\longleftarrow} \underset{\mathbf{T}}{\operatorname{argmin}}\, \varepsilon \times \left(\boldsymbol{a}^{(n)}, \boldsymbol{b}^{(n)}, \boldsymbol{c}^{(n)}, \mathbf{U}^{(n+1)}, \mathbf{S}^{(n+1)}, \mathbf{T}^{(n)}\right)$$
$$\forall i \in I, \boldsymbol{a}^{(n+1)} \overset{\text{SLF-NMU}}{\longleftarrow} \underset{\boldsymbol{a}}{\operatorname{argmin}}\, \varepsilon \times \left(\boldsymbol{a}^{(n)}, \boldsymbol{b}^{(n)}, \boldsymbol{c}^{(n)}, \mathbf{U}^{(n+1)}, \mathbf{S}^{(n+1)}, \mathbf{T}^{(n+1)}\right)$$
$$\forall j \in J, \boldsymbol{b}^{(n+1)} \overset{\text{SLF-NMU}}{\longleftarrow} \underset{\boldsymbol{b}}{\operatorname{argmin}}\, \varepsilon \times \left(\boldsymbol{a}^{(n+1)}, \boldsymbol{b}^{(n)}, \boldsymbol{c}^{(n)}, \mathbf{U}^{(n+1)}, \mathbf{S}^{(n+1)}, \mathbf{T}^{(n+1)}\right)$$
$$\forall k \in K, \boldsymbol{c}^{(n+1)} \overset{\text{SLF-NMU}}{\longleftarrow} \underset{\boldsymbol{c}}{\operatorname{argmin}}\, \varepsilon \times \left(\boldsymbol{a}^{(n+1)}, \boldsymbol{b}^{(n+1)}, \boldsymbol{c}^{(n)}, \mathbf{U}^{(n+1)}, \mathbf{S}^{(n+1)}, \mathbf{T}^{(n+1)}\right) \quad (19)$$

where $\mathbf{U}^{(n)}$, $\mathbf{S}^{(n)}$, $\mathbf{T}^{(n)}$, $\boldsymbol{a}^{(n)}$, $\boldsymbol{b}^{(n)}$, $\boldsymbol{c}^{(n)}$, $\mathbf{U}^{(n+1)}$, $\mathbf{S}^{(n+1)}$, $\mathbf{T}^{(n+1)}$, $\boldsymbol{a}^{(n+1)}$, $\boldsymbol{b}^{(n+1)}$, and $\boldsymbol{c}^{(n+1)}$ denote their initial status at the nth and $(n+1)$th iterations, respectively.

In this paper, we choose to fix the training sequence as shown in (19). However, prior research indicates that the performance of a learning model can evolve with evolutionary computation frameworks like particle swarm optimization [47]–[50].

D. Algorithm Design and Analysis

Based on the above inferences, we design the algorithm of a BNLFT model, as shown in Algorithm 1. Note that BNLFT relies on six procedures, that is, **Update_U**, **Update_S**, and **Update_T** for training LFs in **U**, **S**, and **T**, and ***Update_a***, ***Update_b***, and ***Update_c*** for training LBs in ***a***, ***b***, and ***c***, respectively. Since the pseudocodes for them are highly similar, we only present Procedures **Update_U** as in Algorithm 2 and ***Update_a*** as in Algorithm 3 here. From BNLFT, we clearly see the ADM-based training scheme as given by (19):

Algorithm 2 Procedure Update_U

```
Input: $\Lambda, I, R, \lambda, \mathbf{U}^{(n)}, \mathbf{S}^{(n)}, \mathbf{T}^{(n)}, \boldsymbol{a}^{(n)}, \boldsymbol{b}^{(n)}, \boldsymbol{c}^{(n)}$
Operation                                                                      Cost
1. Initialize U_U$^{|I|\times R}$, U_D$^{|I|\times R}$ = 0.                    $\Theta(|I|\times R)$
2. for each $y_{ijk}$ in $\Lambda$                                             $\times|\Lambda|$
     $\hat{y}_{ijk} = \sum_{r=1}^{R} u_{ir}^{(n)} s_{jr}^{(n)} t_{kr}^{(n)} + a_i^{(n)} + b_j^{(n)} + c_k^{(n)}$    $\Theta(R)$
     for $r = 1$ to $R$ do                                                     $\times R$
       U_U$_{ir}$ = U_U$_{ir}$ + $y_{ijk} \cdot s_{jr}^{(n)} \cdot t_{kr}^{(n)}$                          $\Theta(1)$
       U_D$_{ir}$ = U_D$_{ir}$ + $y_{ijk} \cdot s_{jr}^{(n)} \cdot t_{kr}^{(n)} + \lambda \cdot u_{ir}^{(n)}$       $\Theta(1)$
     end for                                                                   --
   end for                                                                     --
3. for $i = 1$ to $|I|$ do                                                     $\times|I|$
     for $r = 1$ to $R$ do                                                     $\times R$
       $u_{ir}^{(n+1)} = u_{ir}^{(n)} \cdot$ (U_U$_{ir}$/U_D$_{ir}$)                                  $\Theta(1)$
     end for                                                                   --
   end for                                                                     --
   Output: $\mathbf{U}^{(n+1)}$
```

Algorithm 3 Procedure Update_*a*

```
Input: $\Lambda, I, R, \lambda, \mathbf{U}^{(n+1)}, \mathbf{S}^{(n+1)}, \mathbf{T}^{(n+1)}, \boldsymbol{a}^{(n)}, \boldsymbol{b}^{(n)}, \boldsymbol{c}^{(n)}$
Operation                                                                      Cost
1. Initialize a_U$^{|I|}$, a_D$^{|I|}$ = 0.                                    $\Theta(|I|)$
2. for each $y_{ijk}$ in $\Lambda$                                             $\times|\Lambda|$
     $\hat{y}_{ijk} = \sum_{r=1}^{R} u_{ir}^{(n+1)} s_{jr}^{(n+1)} t_{kr}^{(n+1)} + a_i^{(n)} + b_j^{(n)} + c_k^{(n)}$    $\Theta(R)$
     a_U$_i$ = a_U$_i$ + $y_{ijk}$                                              $\Theta(1)$
     a_D$_i$ = a_D$_i$ + $y_{ijk} + \lambda_b \cdot a_i^{(n)}$                    $\Theta(1)$
   end for                                                                     --
3. for $i = 1$ to $|I|$ do                                                     $\times|I|$
     $a_i^{(n+1)} = a_i^{(n)} \cdot$(a_U$_i$/a_D$_i$)                                 $\Theta(1)$
   end for                                                                     --
   Output: $\boldsymbol{a}^{(n+1)}$
```

U, **S**, **T**, ***a***, ***b***, and ***c***. are updated sequentially, where the status of LFs/LBs in each LF matix/LB vector relies on the status of previously trained ones.

Considering the computational complexity of BNLFT, we see that it relies on the complexity of six procedures. First, Procedure Update_U has the complexity

$$\Theta(\text{Update\_}U) = \Theta(|I| \times R + |\Lambda| \times R + |I| \times R) \approx \Theta(|\Lambda| \times R). \quad (20)$$

Note that in an HiDS tensor, we commonly have $|\Lambda| \gg \max\{|I|, |J|, |K|\}$; therefore, we achieve the last result of (20) by reasonably omitting the constant coefficients and lower order terms. By analogy, the complexity of Procedures Update_S and Update_T is also $\Theta(|\Lambda| \times R)$ since they share nearly the same processing flow as that of Procedure Update_U. The complexity of Procedure Update_*a* is

$$\Theta(\text{Update\_U}) = \Theta(|I| + |\Lambda| \times R + |I|) \approx \Theta(|\Lambda| \times R) \quad (21)$$

whose last result is also achieved by reasonably omitting the constant coefficients and lower order terms. From (21), we infer that the complexity of Procedures Update_*b* and Update_*c* is also $\Theta(|\Lambda| \times R)$.

Thus, the computational complexity of the BNLFT is

$$C_{\text{BNLFT}} = \Theta((|I| + |J| + |K|) \times R + 6 \times n \times |\Lambda| \times R) \approx \Theta(n \times |\Lambda| \times R). \quad (22)$$

TABLE II
DATASET DETAILS

Dataset	D1	D2
Data Type	Response-Time	Throughput
Scale	0-20s	0-1,000kbps
Mean	3.165s	9.609kbps
User Count	142	142
Service Count	4,532	4,532
Time Point Count	64	64
Element Count	30,287,611	30,287,611

Note that n and R are positive constants in practice. Thus, the complexity of a BNLFT model is linear with the number of observed elements in an HiDS tensor.

Its storage complexity depends on three factors: 1) it caches Λ, I, J, K, $\mathbf{U}$, $\mathbf{S}$, $\mathbf{T}$, $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$, whose storage cost comes to $\Theta((|I| + |J| + |K|) \times R + |\Lambda|)$; 2) Procedures Update_U, Update_S, and Update_T all rely on auxiliary matrices for updating $\mathbf{U}$, $\mathbf{S}$, and $\mathbf{T}$, whose storage cost comes to $\Theta((|I| + |J| + |K|) \times R)$; and 3) Procedures Update_a, Update_b and Update_c adopt auxiliary arrays for updating $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$, which cost $\Theta(|I| + |J| + |K|)$ in space. Thus, by reasonably ignoring the constant coefficients and lower-order terms, we formulate the storage complexity of BNLFT as

$$S_{\text{BNLFT}} \approx \Theta((|I| + |J| + |K|) \times R + |\Lambda|). \tag{23}$$

In other words, its storage complexity is linear with the number of observed elements and involved entities in an HiDS tensor.

To summarize, with algorithms presented in this section, a BNLFT model is highly efficient in both computation and storage. Next, we validate its performance on user–service–time QoS tensors from real applications.

IV. EMPIRICAL STUDIES

A. General Settings

1) Datasets: The experiments are conducted on two datasets collected by WSMonitor [11], whose details are summarized in Table II. As shown, they describe response time and throughput on 4532 real-world WSs experienced by 142 users during 64 different time points. Either dataset contains 30 287 611 QoS records. With them, we actually have two user–service–time QoS tensors with the size of $142 \times 4532 \times 64$ regarding different QoS properties. Note that the data density of both tensors is 73.53%, which is rather high compared with real cases. Hence, in the experiments, we adopt a small part of them, that is, 5%–40% to train each model, to predict the remaining 95%–60% data for evaluating their performance.

2) Evaluation Metrics: We focus on the accuracy of generated QoS predictions since it directly reflects whether or not the model has captured the essential characteristics of an HiDS tensor. Hence, we adopt mean absolute error (MAE) and root mean square error (RMSE) as the metrics

$$\text{MAE} = \sum_{y_{ijk} \in \Psi} \left| y_{ijk} - \hat{y}_{ijk} \right| / |\Psi|$$

$$\text{RMSE} = \sqrt{\sum_{y_{ijk} \in \Psi} \left(y_{ijk} - \hat{y}_{ijk} \right)^2 / ||\Psi||} \tag{24}$$

TABLE III
DETAILED SETTINGS OF TESTING CASES

Dataset	No.	Train:Test	Training data	Testing data
D1	D1.1	5%:95%	1,514,381	28,773,230
	D1.2	10%:90%	3,028,761	27,258,850
	D1.3	15%:85%	4,543,142	25,744,469
	D1.4	20%:80%	6,057,522	24,230,089
	D1.5	25%:75%	7,571,903	22,715,708
	D1.6	30%:70%	9,086,283	21,201,328
	D1.7	35%:65%	10,600,664	19,686,947
	D1.8	40%:60%	12,115,044	18,172,567
D2	D2.1	5%:95%	1,514,381	28,773,230
	D2.2	10%:90%	3,028,761	27,258,850
	D2.3	15%:85%	4,543,142	25,744,469
	D2.4	20%:80%	6,057,522	24,230,089
	D2.5	25%:75%	7,571,903	22,715,708
	D2.6	30%:70%	9,086,283	21,201,328
	D2.7	35%:65%	10,600,664	19,686,947
	D2.8	40%:60%	12,115,044	18,172,567

where Ψ denotes the validation set and naturally we have $\Psi \cap \Lambda = \emptyset$. For a tested model, small MAE and RMSE values stand for high prediction accuracy.

B. Comparison Results

First of all, we compare the performance of a BNLFT model with state-of-the-art models that are able to perform temporal pattern-aware QoS prediction. The following models are involved in this part of the experiment.

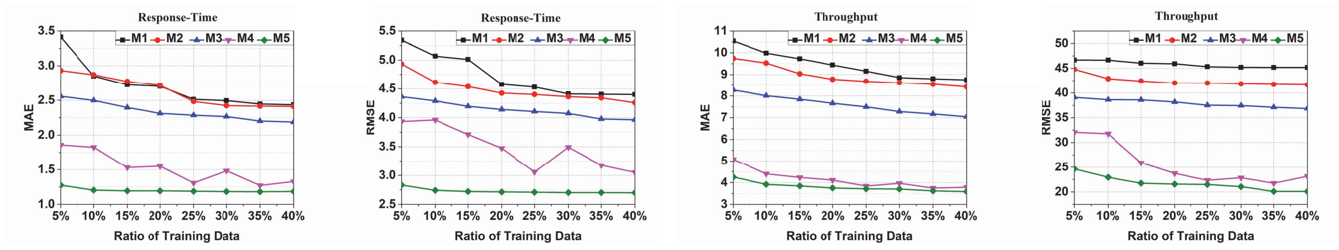
M1 (A Non-Negative Matrix Factorization Model [51], [52]): It considers a user–service–time tensor as a set of user–service matrix slices along the time dimension. On each slice matrix, NMF [52] is adopted for performing LFA. Note that for handling incomplete data, the weighting strategy proposed in [51] is adopted.

M2 (A Time-SVD++ Model Proposed in [25]): It also splits a user–service–time tensor into a set of slice matrices, but models the temporal effects into the objective function of an LF model for temporal-aware predictions of missing data.

M3 (A WSPred Model Proposed in [11]): It applies CPTF to a user–service–time tensor for LFT, and adopts the AGD as a learning scheme. Note that M3 adopts the average QoS data to further regularize its objective function. However, it does not consider the non-negativity of QoS data.

M4 (An NNCP Model Proposed in [9]): It also applies CPTF to a user–service–time tensor for LFT, but adopts SLF-NMU as a learning scheme.

M5 (BNLFT in This Paper): On either dataset, we design eight testing cases, whose details are given in Table III. For instance, the training-data-ratio of testing case D1.1 is 5%, which means that we randomly choose 5% of known data from D1 as Λ to predict the remaining 95% as Ψ. The above process is repeated ten times to achieve ten different sets of experiments for eliminating the possible biases caused by data splitting. Considering the hyper parameter settings, we set the LF space dimension $R = 20$ for all involved models for fair comparison. Meanwhile, to eliminate the affects by initial hypotheses, we initialize each model with the same and randomly generate the hypothesis



(a) (b) (c) (d)

Fig. 5. Impact of tensor density. (a) and (b) are MAE and RMSE on D1; (c) and (d) are MAE and RMSE on D2.

in each set of tests, and adopt different randomly generated hypotheses in different sets of experiments to achieve the final results. In terms of modeling specific hyper parameters, on either dataset, we first tune them for each model on one set of experiments to achieve their optimal values, and then apply the same settings on the other sets of experiments.

The comparison results are depicted in Fig. 5. From these results, we have the following important findings.

It is vital to correctly model the temporal dynamics hidden in a user–service–time tensor to achieve highly accurate predictions for its missing data. As shown in Fig. 1, M1, that is, an NMF model without considering temporal patterns, is clearly outperformed by its peers. For instance, from Fig. 5(b), we see that on D1 with the training-data-ratio at 5%, M1's RMSE is 5.34, which is about 7.68% higher than M2's 4.93. With the training-data-ratio at 40%, M1's RMSE is 4.40, which is still 3.18% higher than 4.26 by M2. Note that although M2 also treats an HiDS tensor as a set of matrix slices where the temporal connections are actually isolated, its temporal modeling enables it to capture part of the temporal patterns in an HiDS tensor, thereby making it outperform M1. Nonetheless, when compared to M5 (our proposed method), where the temporal patterns are carefully discovered through LFT, both M1 and M2 are significantly outperformed by M5 in terms of prediction accuracy. On D1 with the training-data-ratio at 5%, M5's RMSE is 2.83, which is 47.00% and 42.60% lower than that achieved by M1 and M2, respectively. When the training-data-ratio is 40%, M5's RMSE is 2.70, which is about 38.64% and 36.62% lower than the RMSE of M1 and M2, respectively. Similar conclusions can be made on the other testing cases, as depicted in Fig. 5(a), (c), and (d).

It is essential to apply non-negativity constraints to an LFT model to correctly describe non-negative QoS data of an HiDs tensor. M3 performs LFT on an HiDS tensor for QoS prediction. However, its prediction accuracy for missing QoS data is lower than that of M4 and M5, both of which are non-negative LFT models. For instance, on D1 with the training-data-ratio at 5%, M3's MAE is 2.56, which is about 27.34% higher than 1.86 by M4, and 50.00% higher than MAE at 1.28 by M5. When the training-data-ratio increases to 40%, M3's MAE is 2.19, which is about 39.27% and 46.12% higher than 1.33 by M4 and 1.18 by M5, respectively. Similar results can be found on the other testing cases, as depicted in Fig. 5(b)–(d). It turns out that the correct description of data non-negativity is vital for an LFT-based QoS predictor.

TABLE IV
PREDICTION ERROR OF TESTED MODELS
WITH TRAINING-DATA-RATIO = 0.1%

MAE	**M1**	**M2**	**M3**	**M4**	**M5**
D1	4.84	4.12	3.68	3.19	2.06
D2	17.76	16.83	15.54	9.83	7.88
RMSE	**M1**	**M2**	**M3**	**M4**	**M5**
D1	7.53	6.75	6.24	4.90	4.09
D2	76.62	74.41	71.06	62.12	44.02

Algorithm performance when a target HiDS tensor gets sparser. We have tested the performance of involved models when the training-data-ratio decreases to 0.1%. Their MAE and RMSE are summarized in Table IV.

As shown in Table IV, when a target HiDS tensor gets sparser, the accuracy gain by M5 over its peers gets even larger. For instance, M5's MAE on D1 with a training-data-ratio = 0.1% is 2.06, which is 57.43%, 50.00%, 44.02%, and 35.42% lower than M1's 4.84, M2's 4.12, M3's 3.68, and M4's 3.19, respectively. Similar situations are also encountered on the other testing cases. From this phenomenon, we summarize that BNLFT can well address an HiDS tensor with extremely low data sparsity.

For a non-negative LFT model, fine-grained modeling considerations can further enable accuracy gain. By comparing M4 and M5, we see that although both models consider the non-negativity of QoS data, M5 outperforms M4 in prediction accuracy for missing QoS data significantly. For instance, on D2 with the training-data-ratio at 5%, M5's RMSE is 24.64, which is about 23.14% lower than M4's RMSE at 32.06. When the training-data-ratio increases to 40%, M5's RMSE is 20.09 and about 13.18% lower than 23.14 by M4. Compared to M4, M5 further incorporates both the LBs and ADM-based training scheme into the model, thereby achieving such accuracy gain.

LBs incorporation is first adopted in artificial neural networks [31]–[33]. As the LFA-based models emerge during the Netflix prize [31]–[33], it becomes a frequently adopted strategy in various LFA-based models since it can significantly improve the robustness of a resulting model, with additional improvement in prediction accuracy. It is widely accepted that LBs can enlarge the solution space of a resulting model [31]–[33]. Prior research [31]–[33] also points out that they act similarly as some principal factors that stabilize the learning direction of an LFA-based model, thereby

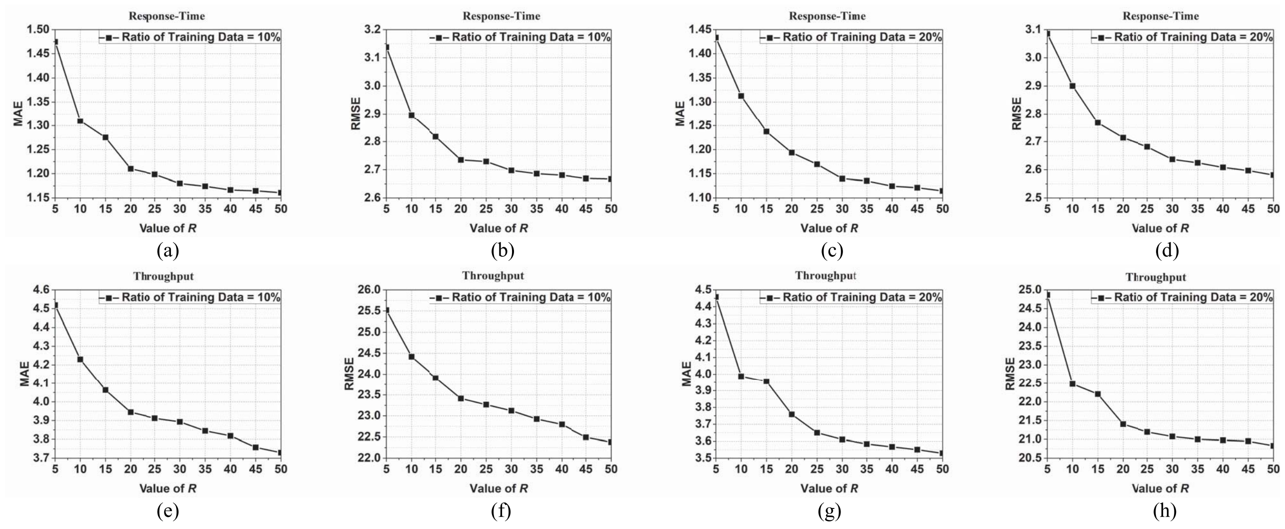


Fig. 6. Impact of dimensionality. (a) and (b) are MAE and RMSE on D1 with Ratio of Training Data = 10%; (c) and (d) are MAE and RMSE on D1 with Ratio of Training Data = 20%; (e) and (f) are MAE and RMSE on D2 with Ratio of Training Data = 10%; (g) and (h) are MAE and RMSE on D2 with Ratio of Training Data = 20%.

TABLE V
TIME COST PER ITERATION OF TESTED MODELS

Dataset (5%:95%)	Time cost per iteration (ms)				
	M1	M2	M3	M4	M5
D1	1341	1822	1326	4884	1623
D2	1212	1468	1095	4456	1298

TABLE VI
TOTAL TIME COST OF TESTED MODELS

Dataset (5%:95%)	Total time cost/Iteration Count				
	M1	M2	M3	M4	M5
D1	1030/768	1629/894	1142/1000	313/64	97/60
D2	663/547	1223/833	982/1000	169/38	34/26

avoiding the training fluctuation caused by irregular data. In our problem, QoS data are highly irregular: they fluctuate constantly in a large scale of real numbers [5], [18]–[21]. Therefore, LB incorporation is highly useful for a BNLFT model to achieve higher prediction accuracy than its peers do, as shown in the experimental results.

As shown in prior research, LBs can be extended from various aspects [31]–[33]. From (18) and Algorithm 1, we see that extended LBs are highly compatible with a BNLFT model: it is feasible to adjust the training scheme with LBs, thereby achieving a BNLFT model with extended LBs. Moreover, it is also highly desired to further clarify the reason why a BNLFT model achieves accurate QoS predictions with them. We plan to address the issue in the future.

Computational efficiency. Tables V and VI summarize the time per iteration and total time of each tested model on D1 and D2 with the training-data-ratio at 5%. Note that similar results are observed on the other testing cases. From these results, we see that M5's time efficiency is competitive. As shown in Table V, its time per iteration is comparable with that of M1-3, and much higher than that of M4. However, as shown in Table VI, it converges with much fewer iterations than M1-3. Hence, its total time is the least among its peers.

As discussed in prior research [30], SLF-NMU indeed makes a model converge slow. However, we find that it requires much fewer iterations in dealing with an HiDS tensor than an HiDS matrix. For instance, as shown in Table VI, M4 and M5 both take dozens of iterations to converge with SLF-NMU. However, it is necessary to make the model converge even faster for high efficiency. In this paper, we adopt an ADM-based training sequence, which can enhance model convergence. As shown in Table VI, M5's iteration count for convergence is less than M4's. Further research regarding this issue is highly desired, and we plan to address it in the future.

To summarize, compared with the state-of-the-art models able to perform temporal-aware QoS prediction, BNLFT achieves significant gain in prediction accuracy for missing data. Its high prediction accuracy mainly relies on: 1) modeling of the temporal patterns through LFT-based LFA on an HiDS tensor; 2) enforcement of non-negativity constraints on the desired LFs and LBs for correctly describing non-negative QoS data; and 3) fine-grained modeling considering constant fluctuations of real QoS data.

C. Effects of R

As indicated by prior research, the LF dimension affects the performance of an LFA-based model for an HiDS matrix. Commonly, the model's prediction error decreases as its LF dimension increases, and tends to stabilize as its LF dimension gets close to the actual rank of a target HiDS matrix. Can we expect similar phenomenon with a BNLFT model on an HiDS tensor? To answer this question, in this set of experiments, we validate its performance as R increases on all testing cases and depict the results on D1 and D2 with the training-data-ratio at

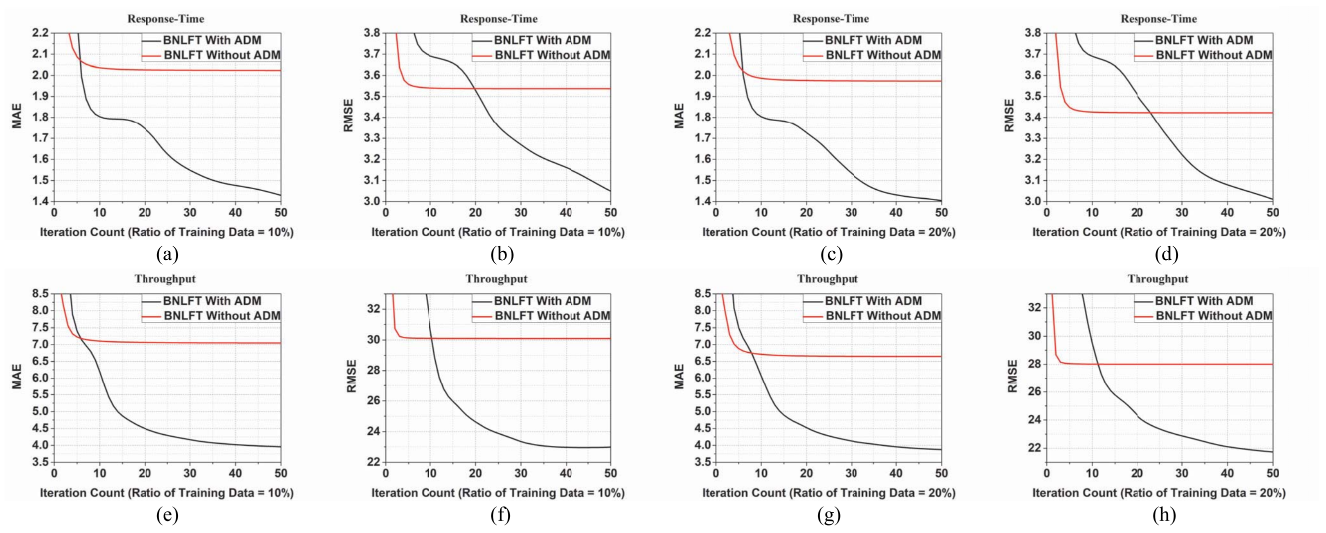


Fig. 7. Impact of ADM-based training rules. (a) and (b) are MAE and RMSE on D1 with Ratio of Training Data = 10%; (c) and (d) are MAE and RMSE on D1 with Ratio of Training Data = 20%; (e) and (f) are MAE and RMSE on D2 with Ratio of Training Data = 10%; (g) and (h) are MAE and RMSE on D2 with Ratio of Training Data = 20%.

10% and 20% in Fig. 6. Note that similar results can also be found on the other testing cases.

From Fig. 6, we see that a BNLFT model's prediction accuracy for missing data is positively related to R, that is, its LF dimension. Initially, its prediction error decreases drastically as R increases. However, as R grows larger, its error decreasing tendency slows down. For instance, on D1 with the training-data-ratio at 10% depicted in Fig. 6(a), its MAE is 1.47 with $R = 5$ and 1.21 with $R = 20$, denoting a 17.69% improvement in prediction accuracy for missing data as R increases from 5 to 20. Nonetheless, as R grows to 50, its MAE is 1.16, and compared to 1.21 with $R = 20$, the improvement is 4.13% only. Similar conclusions can be drawn on the other testing cases as shown in Fig. 6(b)–(h).

Note that as analyzed in Section III-D, BNLFT's computational complexity is linear with R. On the other hand, its prediction error decreases nonlinearly with R and tends to stabilize as R grows close to the actual rank of an HiDS tensor. Therefore, for industrial applications adopting BNLFT for predicting missing QoS data with the highest accuracy, it is important to adopt large R around the actual rank of an HiDS tensor. Meanwhile, since identifying the optimal rank of a tensor is a nondeterministic polynomial-time hard problem, it is desirable to tune R as shown in Fig. 6.

D. *Effects of ADM-Based Training Scheme*

BNLFT adopts the specified training scheme which incorporates the principle of ADM, as shown in (19). What is the performance gain brought by ADM? To address this issue, in this set of experiments, we compare the performance of BNLFT models with and without ADM. Note that for a BNLFT model without ADM, its parameter update rule is also formulated by (18), but all parameters are updated simultaneously in each training iteration.

The comparison results on D1 and D2 with the training-data-ratio at 10% and 20% are depicted in Fig. 7. From Fig. 7, we see that the ADM-based training scheme benefits a BNLFT model's performance greatly. For instance, on D1 with the training-data-ratio at 10% as depicted in Fig. 7(a), the MAE of a BNLFT model without ADM is 2.02, which is about 36.56% higher than 1.28 by a BNLFT model with ADM. Similar observations can be made on Fig. 7(b)–(h). As shown in Fig. 7, a BNLFT model without ADM can be easily stacked by some saddle points during the optimization process, thereby failing to achieve a local optimum as a BNLFT model with ADM would do. Therefore, the incorporation of ADM principle into its training scheme is vital for a BNLFT model to achieve high prediction accuracy for missing data.

E. *Summary*

Based on the empirical studies, we summarize the following items.

1) When performing LFA on an HiDS user–service–time tensor, it is vital to adopt LFT-based LFA to appropriately model temporal dynamics hidden in its data.
2) It is essential to enforce the non-negativity constraints to an LFT-based QoS-predictor to correctly describe non-negative QoS data.
3) Fine-grained model design, including LB modeling and the ADM-based parameter learning rule, is necessary to achieve high prediction accuracy for missing QoS data in a non-negative LFT-based QoS predictor.
4) With the full consideration of the above factors, the proposed BNLFT model is able to predict missing QoS data with temporal variations in an accurate manner never seen in prior methods.

V. CONCLUSION

This paper aims at implementing temporal pattern-aware QoS prediction on time-varying QoS data. To do so, target data are modeled into a user–service–time tensor to perform latent factorization of tensors. A BNLFT model is proposed with the consideration of: 1) LBs; 2) data non-negativity; 3) a highly efficient learning rule; and 4) an ADM-based learning scheme. Compared to state-of-the-art QoS predictors designed for time-varying QoS data, it better grasps the temporal patterns hidden in dynamic data, and clearly outperforms them in terms of prediction accuracy for missing QoS data. Hence, it is a promising model for industrial applications requiring highly accurate predictions for missing QoS data in dynamic environments.

BNLFT has shown its ability in the context of dynamic QoS prediction. However, will it be capable of addressing the problem of extremely large-scale and dynamic network analysis with the help of a distributive computing platform [2], [24], [53], [54]? Further efforts like distributive algorithm design and implementation are desired based on the results presented in this paper.

REFERENCES

[1] Y. Xia, "Cloud control systems," *IEEE/CAA J. Automatica Sinica*, vol. 2, no. 2, pp. 134–142, Apr. 2015.

[2] M. H. Ghahramani, M. C. Zhou, and C. T. Hon, "Toward cloud computing QoS architecture: Analysis of cloud systems and cloud services," *IEEE/CAA J. Automatica Sinica*, vol. 4, no. 1, pp. 6–18, Jan. 2017.

[3] L. Zeng *et al.*, "QoS-aware middleware for Web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.

[4] Y. Liu, A. H. Ngu, and L.-Z. Zeng, "QoS computation and policing in dynamic Web service selection," in *Proc. 13th Int. Conf. World Wide Web*, New York, NY, USA, 2004, pp. 66–73.

[5] X. Luo, M.-C. Zhou, Z.-D. Wang, Y.-N. Xia, and Q.-S. Zhu, "An effective scheme for QoS estimation via alternating direction method-based matrix factorization," *IEEE Trans. Services Comput.*, to be published. doi: 10.1109/TSC.2016.2597829.

[6] S.-G. Deng, H.-Y. Wu, D. Hu, and J.-L. Zhao, "Service selection for composition with QoS correlations," *IEEE Trans. Services Comput.*, vol. 9, no. 2, pp. 291–303, Mar./Apr. 2016.

[7] X. Luo *et al.*, "Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 524–537, Mar. 2016.

[8] X. Luo *et al.*, "Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data," *IEEE Trans. Cybern.*, vol. 48, no. 4, pp. 1216–1228, Apr. 2018.

[9] W.-C. Zhang, H.-L. Sun, X.-D. Liu, and X.-H. Guo, "Temporal QoS-aware Web service recommendation via non-negative tensor factorization," in *Proc. 23rd Int. Conf. World Wide Web*, Seoul, South Korea, 2014, pp. 585–596.

[10] Y. Ma, S. G. Wang, F. C. Yang, and R. N. Chang, "Predicting QoS values via multi-dimensional QoS data for Web service recommendations," in *Proc. IEEE Int. Conf. Web Services*, New York, NY, USA, 2015, pp. 249–256.

[11] Y. Zhang, Z. Zheng, and M. R. Lyu, "WSPred: A time-aware personalized QoS prediction framework for Web services," in *Proc. 22nd IEEE Int. Symp. Softw. Rel. Eng.*, Hiroshima, Japan, 2011, pp. 210–219.

[12] M.-D. Tang *et al.*, "Collaborative Web service quality prediction via exploiting matrix factorization and network map," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 1, pp. 126–137, Mar. 2016.

[13] J.-M. Zhu, P.-J. He, Z.-B. Zheng, and M. R. Lyu, "Online QoS prediction for runtime service adaptation via adaptive matrix factorization," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 2911–2924, Oct. 2017.

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

[15] J. Rao and X. Su, "A survey of automated Web service composition methods," in *Proc. Int. Workshop Semantic Web Services Web Process Composition*, San Diego, CA, USA, 2004, pp. 43–54.

[16] S. Ran, "A model for Web services discovery with QoS," *ACM SIGecom Exchanges*, vol. 4, no. 1, pp. 1–10, 2003.

[17] Y.-L. Zhang, Z.-B. Zheng, and M. R. Lyu, "Exploring latent features for memory-based QoS estimate in cloud computing," in *Proc. 30th IEEE Int. Symp. Reliable Distrib. Syst.*, Madrid, Spain, 2011, pp. 1–10.

[18] W. Lo, J.-W. Yin, S.-G. Deng, Y. Li, and Z. Wu, "Collaborative Web service QoS prediction with location-based regularization," in *Proc. 19th IEEE Int. Conf. Web Services*, Honolulu, HI, USA, 2009, pp. 464–471.

[19] Z.-B. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative Web service QoS prediction via neighborhood integrated matrix factorization," *IEEE Trans. Services Comput.*, vol. 6, no. 3, pp. 289–299, Jul./Sep. 2012.

[20] L.-S. Shao *et al.*, "Personalized QoS prediction for Web services via collaborative filtering," in *Proc. 17th IEEE Int. Conf. Web Services*, Salt Lake City, UT, USA, 2007, pp. 439–446.

[21] Z.-B. Zheng, Y.-L. Zhang, and M. R. Lyu, "Distributed QoS evaluation for real-world Web services," in *Proc. 20th IEEE Int. Conf. Web Services*, Miami, FL, USA, 2010, pp. 83–90.

[22] J. Cao, Z. Wu, Y.-K. Wang, and Y. Zhuang, "Hybrid collaborative filtering algorithm for bidirectional Web service recommendation," *Knowl. Inf. Syst.*, vol. 36, no. 3, pp. 607–627, 2013.

[23] X. Chen, Z.-B. Zheng, and M. R. Lyu, "QoS-aware Web service recommendation via collaborative filtering," *Web Services Foundations*. New York, NY, USA: Springer, 2011, pp. 563–588.

[24] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[25] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Paris, France, 2009, pp. 447–456.

[26] N. Sahoo, P. V. Singh, and T. Mukhopadhyay, "A hidden Markov model for collaborative filtering," *Soc. Sci. Electron. Publ.*, vol. 36, no. 4, pp. 1329–1356, 2012.

[27] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.

[28] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos, "Tensors for data mining and data fusion: Models, applications, and scalable algorithms," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 2, p. 16, 2017.

[29] Y. Koren and R. Bell, "Advances in collaborative filtering," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Boston, MA, USA: Springer, 2011, pp. 145–186.

[30] X. Luo, M.-C. Zhou, Y.-N. Xia, and Q.-S. Zhu, "An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems," *IEEE Trans. Ind. Inf.*, vol. 10, no. 2, pp. 1273–1284, May 2014.

[31] A. Paterek, "Improving regularized singular value decomposition for collaborative-filtering," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, San Jose, CA, USA, 2007, pp. 39–42.

[32] G. Takács, I. Pilászy, B. Németh, and D. Tikky, "Scalable collaborative filtering approaches for large recommender systems," *J. Mach. Learn. Res.*, vol. 10, pp. 623–656, Mar. 2009.

[33] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Comput.*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[34] K. Javed, R. Gouriveau, and N. Zerhouni, "A new multivariate approach for prognostics based on extreme learning machine and fuzzy clustering," *IEEE Trans. Cybern.*, vol. 45, no. 12, pp. 2626–2639, Dec. 2015.

[35] H. Li, Q.-F. Zhang, and J.-D. Deng, "Biased multiobjective optimization and decomposition algorithm," *IEEE Trans. Cybern.*, vol. 47, no. 1, pp. 52–66, Jan. 2017.

[36] D. P. Bertsekas, "Nonlinear programming," *J. Oper. Res. Soc.*, vol. 48, no. 3, p. 334, 1997.

[37] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Las Vegas, NV, USA, 2008, pp. 426–434.

[38] S. Meng *et al.*, "A temporal-aware hybrid collaborative recommendation method for cloud service," in*Proc. IEEE Int. Conf. Web Services*, San Francisco, CA, USA, 2016, pp. 252–259.

[39] W.-K. Wong, Z. Lai, Y. Xu, J. Wen, and C.-P. Ho, "Joint tensor feature analysis for visual object recognition," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2425–2436, Nov. 2015.

[40] T. Xie, S.-T. Li, L.-Y. Fang, and L.-C. Liu, "Tensor completion via nonlocal low-rank regularization," *IEEE Trans. Cybern.*, to be published. doi: 10.1109/TCYB.2018.2825598.

[41] X.-Y. Wang *et al.*, "A spatial-temporal QoS prediction approach for time-aware Web service recommendation," *ACM Trans. Web*, vol. 10, no. 1, pp. 1–25, 2016.

[42] W.-C. Zhang, H.-L. Sun, X.-D. Liu, and X.-H. Guo, "An incremental tensor factorization approach for Web service recommendation," in *Proc. IEEE Int. Conf. Data Min. Workshop*, Shenzhen, China, 2014, pp. 346–351.
[43] S.-G. Wang, Y. Ma, B. Cheng, F.-C. Yang, and R. N. Chang, "Multi-dimensional QoS prediction for service recommendations," *IEEE Trans. Services Comput.*, vol. 12, no. 1, pp. 47–57, Jan./Feb. 2019.
[44] A. Shashua and T. Hazan, "Non-negative tensor factorization with applications to statistics and computer vision," in *Proc. 22nd Int. Conf. Mach. Learn.*, Bonn, Germany, 2015, pp. 792–799.
[45] J. Wang *et al.*, "Diverse non-negative matrix factorization for multiview data representation," *IEEE Trans. Cybern.*, vol. 48, no. 9, pp. 2620–2632, Sep. 2018.
[46] X. Luo *et al.*, "A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 579–592, Mar. 2016.
[47] N. Y. Zeng *et al.*, "Image-based quantitative analysis of gold immunochromatographic strip via cellular neural network approach," *IEEE Trans. Med. Imag.*, vol. 33, no. 5, pp. 1129–1136, May 2014.
[48] N. Y. Zeng, Z. D. Wang, and H. Zhang, "Inferring nonlinear lateral flow immunoassay state-space models via an unscented Kalman filter," *Sci. China Inf. Sci.*, vol. 59, no. 11, 2016, Art. no. 112204.
[49] N. Y. Zeng, H. Zhang, Y. R. Li, J. L. Liang, and A. M. Dobaie, "Denoising and deblurring gold immunochromatographic strip images via gradient projection algorithms," *Neurocomputing*, vol. 247, pp. 165–172, Jul. 2017.
[50] N. Y. Zeng *et al.*, "A new switching-delayed-PSO-based optimized SVM algorithm for diagnosis of Alzheimer's disease," *Neurocomputing*, vol. 320, pp. 195–202, Dec. 2018.
[51] S. Zhang, W.-H. Wang, J. Ford, and F. Makedon, "Learning from incomplete ratings using non-negative matrix factorization," in *Proc. SIAM Int. Conf. Data Min.*, 2006, pp. 549–553.
[52] D. D. Lee and H. S. Seung, "Learning the parts of objects with non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.
[53] W. Pedrycz, "Granular computing for data analytics: A manifesto of human-centric computing," *IEEE/CAA J. Automatica Sinica*, vol. 5, no. 6, pp. 1025–1034, Nov. 2018.
[54] W. Li, Y. Xia, M. Zhou, X. Sun, and Q. Zhu, "Fluctuation-aware and predictive workflow scheduling in cost-effective infrastructure-as-a-service clouds," *IEEE Access*, vol. 6, pp. 61488–61502, 2018.

Xin Luo (M'14–SM'17) received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005 and the Ph.D. degree in computer science from Beihang University, Beijing, China, in 2011.

In 2016, he joined the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China, as a Professor of computer science and engineering. He is currently a Distinguished Professor of computer science with the Dongguan University of Technology, Dongguan, China. His current research interests include big data analysis and intelligent control. He has published over 100 papers (including over 30 IEEE TRANSACTIONS papers) in the above areas.

Dr. Luo was a recipient of the Hong Kong Scholar Program jointly by the Society of Hong Kong Scholars and China Post-Doctoral Science Foundation in 2014, the Pioneer Hundred Talents Program of Chinese Academy of Sciences in 2016, the Advanced Support of the Pioneer Hundred Talents Program of Chinese Academy of Sciences in 2018, and the Outstanding Associate Editor Reward of IEEE ACCESS in 2018. He is currently serving as an Associate Editor for the IEEE/CAA JOURNAL OF AUTOMATICA SINICA, IEEE ACCESS, and *Neurocomputing*. He has also served as the Program Committee Member for many international conferences, including ICDM 2018, IJCAI 2018, AAAI 2018-19, IEEE CASE 2015–2018, IEEE SMC 2016–2018, and IEEE ICNSC 2016–2018.

Hao Wu received the B.S. degree in information security from the Hefei University of Technology, Hefei, China, in 2014 and the M.S. degree in computer science from Chongqing University, Chongqing, China, in 2017. He is currently pursuing the Ph.D. degree in computer science with the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing.

His current research interests include data mining, big data analysis, and artificial intelligence.

Huaqiang Yuan received the B.S. degree in applied mathematics from Xiangtan University, Xaingtan, China, in 1988, the M.S. degree in computer science from the National University of Defense Technology, Changsha, in 1992, and the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China in 1996.

He is currently a Professor of computer science and network security with the Dongguan University of Technology, Dongguan, China. His current research interests include big data analysis and network security.

MengChu Zhou (S'88–M'90–SM'93–F'03) received the B.S. degree in control engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree in automatic control from the Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He joined the New Jersey Institute of Technology, Newark, NJ, USA, in 1990, where he is a Distinguished Professor of electrical and computer engineering. He has over 800 publications, including 12 books, 460 journal papers (360 in IEEE TRANSACTIONS), 12 patents, and 28 book chapters. His current research interests include Petri nets, intelligent automation, Internet of Things, big data, Web services, and intelligent transportation.

Dr. Zhou was a recipient of the Humboldt Research Award for U.S. Senior Scientists from Alexander von Humboldt Foundation, the Franklin V. Taylor Memorial Award, and the Norbert Wiener Award from the IEEE Systems, Man and Cybernetics Society. He serves as the VP for Conferences and Meetings for the IEEE Systems, Man and Cybernetics Society. He is the Founding Editor of IEEE Press Book Series on Systems Science and Engineering and the Editor-in-Chief of the IEEE/CAA JOURNAL OF AUTOMATICA SINICA. He is a Life Member of the Chinese Association for Science and Technology—USA and served as its President in 1999. He is a fellow of the International Federation of Automatic Control, American Association for the Advancement of Science, and Chinese Association of Automation.