Two-Stream Graph Convolutional Network-Incorporated Latent Feature Analysis

Fanghui Bi, Tiantian He, *Member*, *IEEE*, Yuetong Xie, and Xin Luo, *Senior Member*, *IEEE*

Abstract—Historical Quality-of-Service (QoS) data describing existing user-service invocations are vital to understanding user behaviors and cloud service conditions. Collaborative Filtering (CF) models based on Matrix Factorization (MF) have proven to be highly efficient in performing representation learning in QoS data. However, its performance is hindered by its linear inherence and implicit encoding of collaborative QoS signal. To address this critical issue, we present a novel approach in this paper, dubbed as **T**wo-stream **G**raph convolutional network-incorporated **L**atent **F**eature **A**nalysis (TGLFA). The proposed TGLFA is significantly different from previous approaches to representation learning in QoS data in the following three aspects. First, it constructs a multilayer fully-connected network to capture the attribute characteristics representing the nonlinear latent features of service. Second, TGLFA constructs a biparty graph to represent the user-service interactions, where the light graph convolutional network is adopted to acquire the high-order connectivity in QoS data. Last, Aiming to improve computational efficiency, the proposed approach considers the mechanism for data density-oriented modeling when building the input and output layers. Detailed experimental results on eight large-scale cases constructed on two real QoS datasets demonstrate that the proposed TGLFA significantly outperforms its state-of-the-art peers in both estimation accuracy for missing QoS data and computational efficiency. The notable results show that TGLFA is a novel and effective approach to QoS data representation learning.

Index Terms—Quality-of-Service, Representation Learning, Data Science, Cloud Service, Missing Data Estimation, Graph Neural Network, Non-Euclidean Data, High-Dimensional and Incomplete Data, Latent Factor Analysis, Latent Feature Model.

1 Introduction

CLOUD services are ubiquitously employed by industrial software applications under the service-oriented architecture [1], [2], [3]. They are taken as the fundamental components to achieve easy data exchange among these applications over the World Wide Web. In this era of cloud computing, an increasing number of service providers serve their customers by deploying cloud services. Therefore, the number of online cloud services increases explosively [4], [5], [6]. Due to the highly similar functionality of available candidate cloud services, it becomes a vital challenge for users to select appropriate ones to build a reliable system [7], [8].

Most nonfunctional characteristics of cloud services are reflected by Quality-of-Service (QoS) data at both the server and user sides [9], [10], [11], [12], [13], which are critical for service selection. Specifically, user-service selections are influenced by various QoS properties, e.g., availability, cost, robustness, capacity, confidentiality, reputation, throughput, accessibility, response time, authentication, and interoperability [2], [11], [12], [13]. Among them, service providers can provide server-side QoS data, e.g., popularity and price. On the other hand, response time, availability, cost, throughput, and other user-side QoS metrics vary sharply due to both the server and user-side factors, such as invocation burden and network condition [2], [13], [14], [15]. Conducting warming-up tests that directly invoke the cloud services involved appears to be a straightforward approach to estimating the user-side QoS [1], [2], [3]. However, such warming-up tests are always time-consuming and economically expensive since most invocations provided by candidate services are charged.

Motivated by the success of Collaborative Filtering (CF) in e-commerce, researchers attempt to adopt this technique to perform QoS estimation aiming at accurately estimating missing QoS data based on historical records [16], [17], [18], [19], [20], [21], [22], [23]. A CF-based QoS estimator works on a given user-service QoS matrix, where the information of users, cloud services, and their corresponding invocations are recorded in its rows, columns, and entries, respectively. In real-world scenarios, the target QoS matrix is highly sparse since a user typically experiences a very limited number of candidate cloud services [1], [2], [3], [20], [21], [22], [23]. Hence, how to perform efficient and accurate representation learning to such a highly sparse user-service matrix is a major challenge of CF-based QoS estimation.

Among various CF-based QoS estimators, Matrix Factorization (MF)-based models for Latent Feature Analysis (LFA) have proven to be effective in learning representations for QoS data [21], [22], [23], [24], [25], [26], [27], [28]. To perform the task of QoS data estimation, they first map both users and services into a low-rank latent feature space to extract their latent features based on the known entries in a target user-service QoS matrix. The inner products between corresponding latent feature vectors of users and cloud services are then calculated to estimate the missing entries. A pyramid of sophisticated LFA models has emerged in recent years. Zhu *et al.* [21] attempt to achieve reliable and accurate QoS estimation by regularizing the low-rank matrix factorization to be location-aware and user-privacy-preserved. Luo *et al.* [24] adopt the alternating direction method and ensemble mechanism to build an effective

- *F. Bi is with the College of Computer and Information Science, Southwest University, Chongqing 400715, China, and also with the Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, and also with the Chongqing School, University of Chinese Academy of Sciences, Chongqing 400714, China. E-mail: bifanghui@cigit.ac.cn.*
- *T. He is with the Center for Frontier AI Research, Institute of High Performance Computing, Agency for Science, Technology and Research (A*STAR), 699010, Singapore. E-mail: he_tiantian@cfar.a-star.edu.sg.*
- *Y. Xie is with the Faculty of Data Science, City University of Macau. E-mail: D21090101535@cityu.mo.*
- *X. Luo is with the College of Computer and Information Science, Southwest University, Chongqing 400715, China. E-mail: luoxin@swu.edu.cn.*
- *F. Bi and T. He are the co-first authors of this paper.*
- *Corresponding Author: X. Luo.*

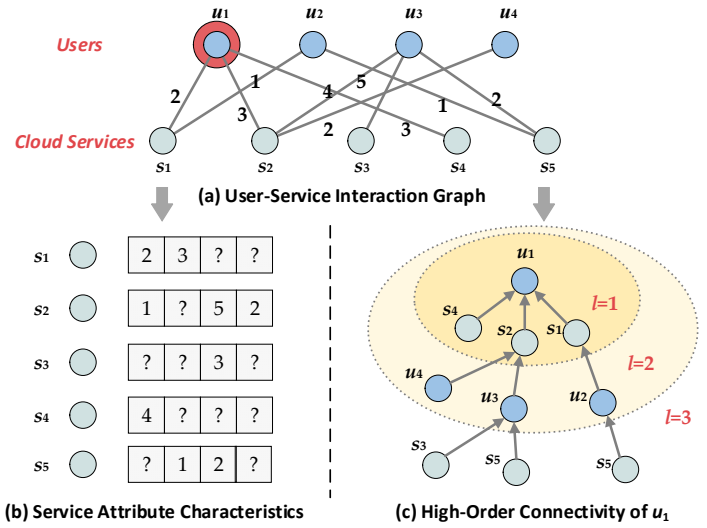


Fig. 1. An illustrative example of attribute characteristics (e.g., response time) and high-order connectivity information in user-service interactions.

LFA-based QoS estimator. Zhao *et al.* [29] present a theoretical analysis on the popularity bias in graph-based collaborative filtering and carefully control the normalization strength to balance the accuracy and novelty in the neighborhood aggregation process. Yang *et al.* [25] consider the correlation between users and services, i.e., incorporating the neighborhood information into the latent factorization process for accurate QoS estimation. Fan *et al.* [30] introduce a graph trend CF method to overcome the non-robust and non-adaptive propagation problems. To address the cold-start problem in QoS estimation, Ryu *et al.* [19] extract the fusion of invocation and neighborhood similarity, which is then used to model the preference propagation by matrix factorizations. Wu *et al.* [26] propose a density-peaks-based clustering method to detect the neighborhoods and noises for achieving a data-characteristic-aware model.

Despite their effectiveness, existing LFA-based QoS estimators are observed to suffer from several inherent defects:

(1) First, the attribute characteristics of users/services carried by the user-service interaction graph are crucial for representation learning. They are yet under-explored by existing models. For example, four users (u_1, u_2, u_3, u_4) depicted in Fig. 1(a-b) invoke five services with solid lines, and the numbers indicate the observed invocations on each service by corresponding users. These records can be organized as vectors of services' attribute characteristics, which are always ignored by existing models. Moreover, existing LFA models mostly rely on matrix factorization [16], [24], whose linear nature limits their ability to capture the implicit connections between users and services. Other studies propose to replace the inner product with multilayer perception for nonlinear interactions. However, they suffer from a high computational burden [10], [17], [22].

(2) Second, the non-Euclidean structure in QoS data, i.e., high-order connectivity information shown in Fig. 1(c), is not fully captured by existing models. Pioneer research has noticed this drawback and put forward various improving strategies [10], [31], [32], [33], [34], [35]. However, they fail to capture the higher-order neighborhood information (up to the third or fourth order), resulting in sparse collaborative QoS signals and accuracy loss.

(3) Third, existing nonlinear models, e.g., GCN-based ones [6], [10] adopt less-efficient methods for modeling the sparse graph, where the detected links are commonly much fewer than the unknown ones. Consequently, they suffer from considerable time cost when learning in large-scale incomplete QoS data.

Given the observations mentioned above, we have the following open questions: Is it possible to perform effective representation learning in QoS data by simultaneously capturing unique attribute characteristics and higher-order connectivity information? ***Can an improved nonlinear LFA model for highly accurate and efficient cloud service QoS estimation be successfully implemented?*** To answer these critical questions, we innovatively propose a model named **T**wo-stream **G**raph convolutional network-incorporated **L**atent **F**eature **A**nalysis (TGLFA) with the following three-fold ideas:

(1) A fully-connected network is built for capturing high-level and nonlinear latent features from attribute characteristics of service QoS data.

(2) The light graph convolutional network [28] is adopted to integrate user-service interactions, i.e., the bipartite graph structure, into the process of representation learning. High-order connectivity information can be learned by TGLFA from the holistic graph structure.

(3) The principle of data density-oriented modeling is considered by TGLFA to construct input and output layers so that high computational efficiency can be achieved. The CUDA-based method for accelerating the computation in incomplete data is also integrated into TGLFA to significantly reduce the training time.

Accordingly, the main contributions of the paper can be summarized as follows:

(1) We propose TGLFA, a novel approach to representation learning in cloud service QoS data. Different from existing approaches, TGLFA considers capturing both attribute characteristics and high-order neighborhood information (even to the 20th order) when learning representations in QoS data. The subsequent tasks of QoS estimation can be accomplished with the learned representations effectively and efficiently. TGLFA provides researchers and engineers in the service computing community with a well-designed platform for QoS estimation tasks.

(2) We have extensively conducted empirical studies on eight large-scale testing cases constructed from two commonly-used real QoS datasets to evaluate the proposed TGLFA. The notable experimental results obtained demonstrate that TGLFA significantly outperforms its state-of-the-art peers in both estimation accuracy for missing cloud service QoS data and computational efficiency.

Section 2 gives the Preliminaries. Section 3 presents the TGLFA model. Section 4 provides the Experimental Results and Analyses. Section 5 performs the Discussions. In the end, Section 6 draws the Conclusions of this paper.

2 Preliminaries

Table 1 summarizes the notations used in this paper. A QoS matrix describing the relationship between user and service sets is the fundamental input for an LFA-based QoS estimator, and it is well defined in [28], [36], [37], [38], [39], [40], [41]:

***Definition* 1. (*A QoS matrix*).** Given a user set U and a service set S, $Q^{|S|\times|U|}$ is a QoS matrix where each element $q_{s,u}$ describes a QoS record of invocation by $u \in U$ on $s \in S$.

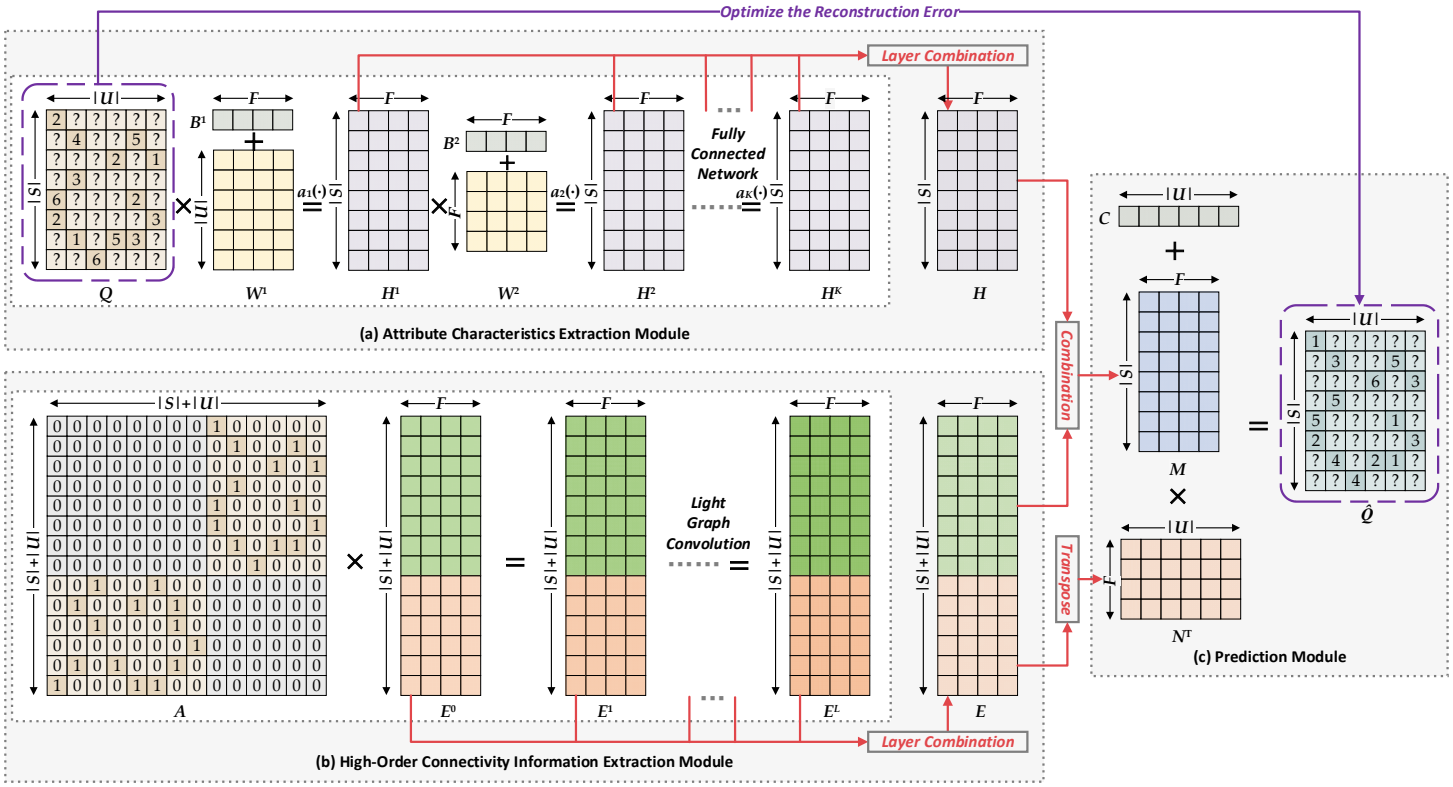


Fig. 2. The structure and flow chart of the proposed TGLFA: (a) The attribute characteristics module receives a highly-incomplete user-service QoS matrix and learns services' representations by a multilayered and fully-connected network; (b) The module for extracting high-order connectivity iteratively performs light graph convolution with a constructed adjacency matrix; (c) The prediction module combines the outputs of Modules (a) and (b), and then estimates the missing QoS data via the inner product of their achieved latent feature vectors.

TABLE 1
NOTATIONS USED IN THE MANUSCRIPT.

Symbol	Description
$\mathcal{U}$, $\mathcal{S}$	Concerned user and service sets.
Q	An $\|\mathcal{S}\|\times\|\mathcal{U}\|$ QoS matrix between $\mathcal{U}$ and $\mathcal{S}$.
Λ, O	Known and unknown entry sets of Q.
F	Latent feature space dimension.
$\hat{Q}$	Q's rank-F approximation.
$q_{s,u}$, $\hat{q}_{s,u}$	Single entries in Q and $\hat{Q}$.
K	Number of hidden layers in Module (a).
H^k	$\|\mathcal{S}\|\times F$ services' feature matrix in the kth layer.
$h^k_{s,f}$	Single entries in H^k.
W^k, B^k	Weight matrix and bias vector of kth hidden layer.
$w^k_{d,f}$, b^k_f	Single variables in W^k and B^k.
L	Number of propagation layers in Module (b).
E^l	$(\|\mathcal{U}\|+\|\mathcal{S}\|)\times F$ combined feature matrix of lth layer.
e^l_u, e^l_s	lth order neighborhood representations of u and s in E^l.
A	$(\|\mathcal{U}\|+\|\mathcal{S}\|)\times(\|\mathcal{U}\|+\|\mathcal{S}\|)$ adjacency matrix of Q.
D	A's degree matrix.
$\tilde{A}$	A's symmetrically normalized matrix.
$\tilde{A}^*$	Pre-trained average combined matrix.
M, N	Resultant latent feature matrices of users and services.
$m_{s,f}$, $n_{u,f}$	Single entries in M and N.
C	Bias vector of the output layer.
c_u	Single variables in C.
$\|\cdot\|$	Cardinality of the involved set.
$\lceil\cdot\rceil$	Ceiling function of an involved number.
$a_k(\cdot)$	kth layer activation function.
$\varepsilon(\cdot)$	Loss function.
$N(s)$, $N(u)$	First order neighbor sets of s and u.
$\Lambda(s)$, $\Lambda(u)$	Subsets of Λ related to each s and u in Q.
Ψ	Testing set from Λ.
η	Learning rate.
λ	L_2 regularization coefficient.
ω	Coefficient controlling the importance of $E_{1\sim\|\mathcal{S}\|}$ and H.
T_1, S_1	Time and storage costs of Algorithm 1.
t, T_{max}	Current and maximum training epochs.

Since a user usually invokes a very limited number of candidate cloud services, Q is highly incomplete. Specifically, let Λ and O denote the known and unknown entry sets, $|\Lambda|$ is commonly far less than $|O|$. In light of prior research [25], [26], [27], [28], [42], [43], [44], [45], [46], an LFA-based QoS estimator has proven to be able to efficiently perform representation learning to such QoS data, which has been introduced in [9], [26], [27], [38], [39], [40], [41]:

***Definition* 2. (*An LFA-based QoS estimator*).** Given Q, an LFA-based QoS estimator builds Q's rank-F approximation $\hat{Q}$ only based on Λ, generating an estimate $\hat{q}_{s,u}$ for $s\in\mathcal{S}$ and $u\in\mathcal{U}$ such that the generalized loss (e.g., $\sum_{q_{s,u}\in\Lambda}(q_{s,u}-\hat{q}_{s,u})^2$) is minimized.

3 THE TGLFA MODEL

In this section, we introduce the proposed TGLFA, which can be further divided into User-oriented TGLFA (U-TGLFA) and Service-oriented TGLFA (S-TGLFA). Note that they have the same structure, but the former takes each user's records regarding all invoked cloud services as the input data, while the latter takes each service's invocation records as the input data. This paper presents the Service-oriented TGLFA for conciseness, whose structure and processing flow are illustrated in Fig. 2. It consists of three main components:

(a) **The module for extracting attribute characteristics** receives the input QoS matrix and adopts a fully-connected network to acquire the nonlinear service representations;

(b) **The module for extracting high-order connectivity information** iteratively performs light graph convolution to learn smooth representations for users and services on their interaction graph;

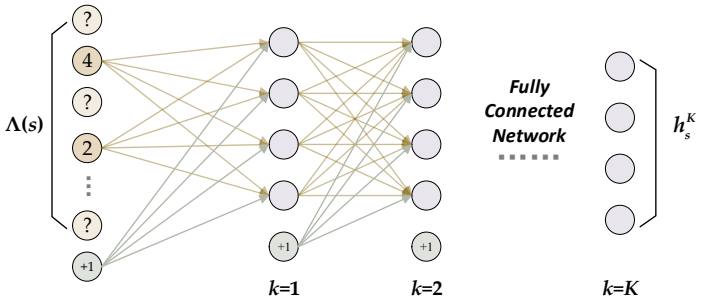


Fig. 3. An illustrative example of extracting the attribute characteristics for service *s*. In the first layer, all input nodes regarding unknown QoS data are not activated to obtain high computational efficiency.

(c) **The prediction module** performs estimation, e.g., calculating the inner product, to predict the unknown QoS data based on the learned representations of users and services.

3.1 Extracting Attribute Characteristics

The proposed LFA model can learn more accurate nonlinear representations for services by feeding the known invocation records in a user-service QoS matrix. To capture such attribute characteristics in the sparse QoS data, we design an extraction module based on skip connection and data density-oriented modeling. Fig. 2(a) illustrates its architecture, and Fig. 3 depicts the propagation process of a specified service s, whose full explanations are as follows.

Input Layer. The input layer receives the service-oriented QoS data, i.e., an $|S|\times|U|$ QoS matrix Q describing the attribute characteristics of services. As illustrated in Figs. 2(a) and 3, the known data are marked with real numbers denoting the user-service invocation records (e.g., throughput values), and the remaining unknown ones are marked with question marks. Since the real-world QoS data are mostly highly sparse, previous methods [6], [10] fill the unknown parts with artificial values, e.g., zeros, which bring huge computational costs. Following the principle of data density-oriented modeling, TGLFA only activates the input layer nodes based on the known user-service interactions to achieve high efficiency.

Hidden Layer. The module for extracting attribute characteristics makes use of multiple fully-connected hidden layers to learn high-level nonlinear latent features. As shown in Fig. 2(a), H^k denotes the kth layer latent feature matrix, W^k and B^k denote the corresponding weight matrix and bias vector in the kth layer. Note that the kth layer node count F^k is set uniformly for simplicity, i.e., $F^1=F^2=\cdots=F^K=F$. Taking the first layer as an example, the mapping formula can be written as:

$$h^1_{s,f}=a_1\left(\sum_{u\in\Lambda(s)} q_{s,u}w^1_{u,f}+b^1_f\right), \tag{1}$$

where $q_{s,u}$, $h^1_{s,f}$, $w^1_{u,f}$, and b^1_f are single entries in Q, H^1, W^1, and B^1, $\Lambda(s)$ denotes the known entry subset related to s, and $a_1(\cdot)$ denotes the activation function that can be hyperbolic tangent (tanh), sigmoid or Rectified Linear Unit (ReLU). Since $|\Lambda(s)|\ll|U|$, Eq. (1) significantly reduces the computational and storage cost.

For the (2~K)th hidden layers, the computing process follows the principle of a fully-connected network:

$$h^k_{s,f}=a_k\left(\sum_{d=1}^{F} h^{k-1}_{s,d}w^k_{d,f}+b^k_f\right), \tag{2}$$

where $h^k_{s,f}$, $h^{k-1}_{s,d}$, $w^k_{d,f}$, and b^k_f are the single entries in H^k, H^{k-1}, W^k, and B^k respectively, and $a_k(\cdot)$ is the activation function. Different activation functions are tested in our implementation, and we find that uniformly adopting the sigmoid function for all layers ensures the best performance in our implementation.

Layer Combination. When learning in large-scale QoS data, a multilayer neural network suffers from long-tail convergence and accuracy loss due to the issue of vanishing gradients. Inspired by the idea of skip connection, we incorporate the layer combination into our attribute characteristics module. As shown in Fig. 2(a), after propagating K hidden layers, the weighted sum of all the obtained latent features is regarded as the output and calculated as follows:

$$h_{s,f}=\alpha_1 h^1_{s,f}+\alpha_2 h^2_{s,f}+\cdots+\alpha_K h^K_{s,f}, \tag{3}$$

where $h_{s,f}$ is the single element in H, and $\alpha_k\geq 0$ denotes the importance of kth layer features, respectively. Note that H is the resultant service representations learned by the attribute characteristics extraction module, and α_k is set at one uniformly for simplicity.

3.2 Extracting High-Order Connectivity Information

As discussed in [10], [28], [29], [30], [31], [32], [33], explicitly exploiting the high-order connectivity information from the user-service interaction graph can acquire stronger collaborative QoS signal, thereby boosting the accuracy of resultant latent feature representations. Recently, Graph Convolutional Networks (GCNs) [6], [10] have become highly popular owing to their high accuracy and scalability in capturing high-order connectivity information, which works by iteratively performing message passing to aggregate multi-hop neighborhood information. Inspired by this discovery, we propose a module for capturing high-order connectivity information, based on the light graph convolutional network (Fig. 2(b)), to perform effective representation learning for QoS data.

Complete Graph Convolution. When addressing non-Euclidean QoS data, an empirical graph convolutional layer with self-connection is given as:

$$E^{l+1}=\sigma\left(\left(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}+I\right)E^lW^l\right), \tag{4}$$

where A is a $(|U|+|S|)\times(|U|+|S|)$ adjacency matrix build with the bipartite user-service interaction graph directly, and I is the identity matrix used to represent the self-loops on nodes:

$$A=\begin{pmatrix}0 & Q_{adj}\\ Q^T_{adj} & 0\end{pmatrix}, \tag{5}$$

where Q_{adj} is Q's adjacency matrix. D is A's diagonal degree matrix, in which each entry D_{ii} denotes the number of invocation records regarding each user or service, i.e., the nonzero entries in A's ith row vector. E^l denotes the latent feature matrix of the lth layer and W^l is the feature transformation weight matrix. $\sigma(\cdot)$ is a nonlinear activation function such as ReLU. Specifically, corresponding to the matrix form given in Eq. (4), the message passing rules of each user u and service s are formulated as:

$$\begin{cases} e^{l+1}_u=\sigma\left(W^le^l_u+\sum\limits_{s\in N(u)}\dfrac{1}{\sqrt{|N(u)|}\sqrt{|N(s)|}}W^le^l_s\right),\\ e^{l+1}_s=\sigma\left(W^le^l_s+\sum\limits_{u\in N(s)}\dfrac{1}{\sqrt{|N(s)|}\sqrt{|N(u)|}}W^le^l_u\right),\end{cases} \tag{6}$$

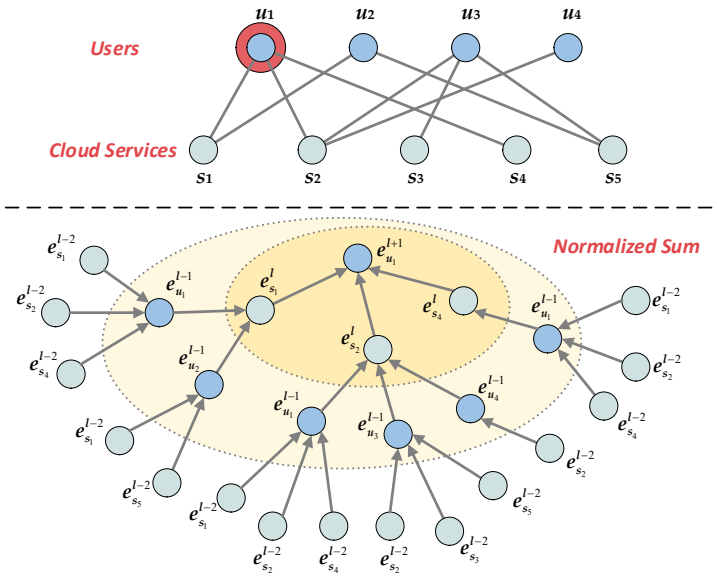


Fig. 4. An illustrative example of extracting the high-order connectivity information for user u_1. In light graph convolution, the feature transformation, self-connection, and nonlinear activation are all removed; only the normalized sum of neighborhood representations is aggregated towards the next layer, whose simplified structure reduces the training difficulty thereby achieving higher estimation accuracy and computational efficiency compared with the complete one.

where $\frac{1}{\sqrt{|N(u)|}\sqrt{|N(s)|}}$ are the symmetrically normalized form, $N(u)$ and $N(s)$ denote the directly-connected neighbor sets of u and s. The message passing rule in Eq. (6) can avoid the feature scale increasing and numerical instabilities owing to multiple graph convolutional operations and attenuate the propagated message with the path length.

Light Graph Convolution. Despite the broad success of GCNs in various graph learning tasks, recent studies [29], [31], [32], [33] indicate that appropriate simplification of the network structure enables efficiency gain. Based on this finding, we modify the graph convolutional layer shown in Eq. (4) to a light form:

$$E^{l+1} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} E^{l}. \tag{7}$$

Compared with the empirical graph convolution in Eq. (4), the proposed light graph convolution (Eq. (7)) removes feature transformation matrix (W^l), nonlinear activation function ($\sigma(\cdot)$), and self-loops (I). This simplified graph convolutional operation is regarded as a fixed low pass filter from the perspective of graph spectral domain and greatly reduces the training cost of node representations, i.e., E^0 [29], [31], [32], [33]. As shown in Fig. 4, for each specified user or service, the (l+1)th layer feature is obtained by only aggregating the normalized representations of first order neighbors in the lth layer. Accordingly, the message passing rules in Eq. (6) are simplified as:

$$\begin{cases} e_u^{l+1} = \sum_{s \in N(u)} \frac{1}{\sqrt{|N(u)|}\sqrt{|N(s)|}} e_s^l, \\ e_s^{l+1} = \sum_{u \in N(s)} \frac{1}{\sqrt{|N(s)|}\sqrt{|N(u)|}} e_u^l. \end{cases} \tag{8}$$

Layer Combination. Note that the above light graph convolutional propagation process starts from E^0, which contains the representations of all users and services. Considering l in {1, 2, …, L}, E^l only contains the aggregated representations of lth order neighbors as the self-connections are removed in light graph convolution. To obtain the representations embedded with different orders of neighborhood information, we perform the layer combination operation before estimation, i.e., calculating the weighted sum of the latent features propagated at each layer:

$$E = \beta_0 E^0 + \beta_1 E^1 + \beta_2 E^2 + \cdots + \beta_L E^L, \tag{9}$$

where E is the resultant representation matrix, and $\beta_l \geq 0$ denotes the importance of the features at the lth layer. Note that Eq. (9) contributes a similar effect as self-connection, and it can alleviate the over-smoothing. With it, for each user u and service s, the output representations are:

$$\begin{cases} e_u = \beta_0 e_u^0 + \beta_1 e_u^1 + \beta_2 e_u^2 + \cdots + \beta_L e_u^L, \\ e_s = \beta_0 e_s^0 + \beta_1 e_s^1 + \beta_2 e_s^2 + \cdots + \beta_L e_s^L, \end{cases} \tag{10}$$

Pre-Trained Strategy. Although the proposed light graph convolution is much faster than the empirical, it still suffers from multiple data exchanges and gradient vanishment. After analyzing the representation matrix E (Eq. (9)), we propose an effective pre-training strategy to address the problem encountered by light graph convolution. By incorporating the propagation rules of (7) into (9), E can be calculated as:

$$E = \beta_0 E^0 + \beta_1 \tilde{A} E^0 + \beta_2 \tilde{A}^2 E^0 + \cdots + \beta_L \tilde{A}^L E^0, \tag{11}$$

where $\tilde{A}=D^{-1/2}AD^{-1/2}$ is A's symmetrically normalized matrix. As for the importance coefficients, we set them uniformly as $1/(L+1)$. In this case, Eq. (11) is reformulated as:

$$\begin{aligned} E &= \frac{1}{L+1} E^0 + \frac{1}{L+1} \tilde{A} E^0 + \frac{1}{L+1} \tilde{A}^2 E^0 + \cdots + \frac{1}{L+1} \tilde{A}^L E^0 \\ &= \frac{1}{L+1} \left(E^0 + \tilde{A} E^0 + \tilde{A}^2 E^0 \cdots + \tilde{A}^L E^0 \right) \\ &= \frac{1}{L+1} \left(I + \tilde{A} + \tilde{A}^2 + \cdots + \tilde{A}^L \right) E^0 \\ &= \tilde{A}^* E^0, \end{aligned} \tag{12}$$

where $\tilde{A}^* = (I+\tilde{A}+\tilde{A}^2+\cdots+\tilde{A}^L)/(L+1)$ and I is the identity matrix. Given the derivation above, we see that only E^0 needs optimization during the whole propagation process, which is convenient to train and thereby acquires more accurate node representations. When calculating E, the combined matrix containing different levels of high-order neighborhood information in can be directly constructed by $\tilde{A}$ (Eq. (12)). Therefore, after setting L, i.e., determining the desired order of neighborhoods, we can calculate the combined adjacency matrix before model training. With this strategy, we avoid multiple iterations during the training process and thus achieving affordable computational cost. It strongly supports capturing high-order QoS signals in a highly efficient way.

3.3 Prediction Module

For a QoS graph, the attribute characteristics and high-order neighborhood information greatly enrich each node's representation. Hence, by performing attribute characteristics extraction and light graph convolution, we address the data sparsity problem to obtain complementary latent features. After that, in the prediction module, we obtain the estimations and minimize the differences between them and the truth values.

Model Prediction. Firstly, the final representation matrix of service is formulated as:

$$M = \omega E_{1\sim|S|} + (1-\omega) H, \tag{13}$$

where M is the service latent feature matrix, and ω is a hy-

perparameter representing the importance of $E_{1\sim|S|}$ and H^K in constituting M. According to Eq. (13), for a specified service, its representation is calculated as:

$$m_s = \omega e_s + (1-\omega) h_s, \tag{14}$$

where m_s is representation to s. Then we can extract N from E as the user latent feature matrix (see Fig. 2). Based on M and N, TGLFA estimates the interaction between each user and service by calculating the inner product with bias:

$$\hat{q}_{s,u} = \sum_{f=1}^{F} m_{s,f} n_{u,f} + c_u, \tag{15}$$

where $\hat{q}_{s,u}$ is the estimation for each $q_{s,u}$, and c_u is the bias for u to enlarge the solution space. Note that the combination and estimation strategies can be further extended to boost the performance of TGLFA with different methods, e.g., a graph attention networks.

Model Training. As presented in Section 2, for optimizing model parameters, we utilize Euclidean distance, which is a commonly adopted strategy, to construct the objective function:

$$\begin{aligned}&\varepsilon\left(W^1, W^2, \cdots, W^K, B^1, B^2, \cdots, B^K, E^0, C\right)\\ &= \sum_{q_{s,u}\in\Lambda} \left(q_{s,u} - \hat{q}_{s,u}\right)^2 + \lambda \begin{pmatrix} \left\|W^1\right\|_F^2 + \left\|W^2\right\|_F^2 + \cdots + \left\|W^K\right\|_F^2 + \left\|E^0\right\|_F^2 \\ + \left\|B^1\right\|_2^2 + \left\|B^2\right\|_2^2 + \cdots + \left\|B^K\right\|_2^2 + \left\|C\right\|_2^2 \end{pmatrix},\end{aligned} \tag{16}$$

where Λ denotes the known entry set and λ controls the strength of L_2 regularization to prevent overfitting. Like the input layer in the module for attribute characteristics extraction, Eq. (16) follows the principle of data density-oriented modeling to only activate the known entries in Q, which avoids a large number of redundant computations regarding the missing data. The computation and storage costs of TGLFA are therefore significantly reduced. To train the proposed TGLFA, we adopt the mini-batch Adaptive Moment Estimation (Adam) optimizer [47] to optimize the parameters in an end-to-end manner.

3.4 Algorithm Design and Analysis

Based on descriptions above, learning TGLFA is summarized in Algorithm 1. Its time cost involves the following two factors:

(1) Initialization: $T_{11}=\Theta((|U|+|S|)\times F)$; (17a)

(2) Two-stream light GCN-based learning:

$$T_{12} = \Theta\left(|\Lambda| \times F \times L \times \left\lceil \frac{|S|}{batch\ size} \right\rceil \times t\right). \tag{17b}$$

Therefore, the overall computational cost of Algorithm 1 is:

$$\begin{aligned} T_1 &= T_{11} + T_{12} \\ &\approx \Theta\left(|\Lambda| \times F \times L \times \left\lceil \frac{|S|}{batch\ size} \right\rceil \times t\right). \end{aligned} \tag{18}$$

Note that in the input and output of Modules (Fig. 2 (a) and (c)), we adopt the principle of data density-oriented modeling rather than performing data filling, which makes their time cost at $\Theta(|\Lambda|\times F\times t)$. Hence, the main time cost is given by the light graph convolutional operations in Module (b). As discussed in Section 3.2, with the well-designed simplified graph convolution and pre-trained strategy, TGLFA achieves high computational efficiency compared to its peers.

As for the storage cost of Algorithm 1, it consists of the following five factors:

(1) Data caching: $S_{11}=\Theta(|\Lambda|)$; (19a)

(2) Weight and bias parameter arrays: $S_{12}=\Theta(|U|\times F)$; (19b)

(3) Arrays caching the node representation parameters:

ALGORITHM 1. TGLFA

Input: Λ, U, S

	Operation	**Cost**
	/* **Initialization** */	
	Initialize $W^1, \cdots, W^K, B^1, \cdots, B^K$	
1:	**Initialize** E^0, C, F, K, L	T_{11}
	Initialize η, λ, ω, *batch size*, T_{max}	
	/* **Two-Stream Light GCN-Based Learning** */	
2:	Construct sparse A and D according to formula (5);	
3:	Calculate $\tilde{A}$ based on $\tilde{A}=D^{-1/2}AD^{-1/2}$ in a sparse fashion;	
4:	**while not** converge and $t\le T_{max}$ **do**	
5:	**for** *batch*=1 to $\lceil\|S\|/batch\ size\rceil$	
6:	**for** l=1 to L	
7:	Calculate E^l according to formulas (7) and (8);	
8:	**end for**	
9:	Calculate E according to formulas (9) and (10);	
10:	Sample a *batch size*$\times|U|$ Q_{batch} from Q;	
11:	**for** s=1 to *batch size*	
12:	**for** f=1 to F	
13:	Calculate $h_{s,f}^1$ according to formula (1);	
14:	**end for**	
15:	**for** k=2 to K	
16:	**for** f=1 to F	
17:	Calculate $h_{s,f}^k$ according to formula (2);	T_{12}
18:	**end for**	
19:	**end for**	
20:	Calculate h_s according to formula (3);	
21:	Calculate m_s according to formula (14);	
22:	**for** $u\in\Lambda(s)$	
23:	Calculate $\hat{q}_{s,u}$ according to formula (15);	
24:	**end for**	
25:	Calculate the loss of s according to formula (16);	
26:	**end for**	
27:	Accumulate the losses in Q_{batch};	
28:	Calculate the gradients by backpropagation;	
29:	Update all the variables based on the gradients;	
30:	**end for**	
31:	**end while**	
	/* **Operation Ending** */	

Output: W^1, W^2, …, W^K, B^1, B^2, …, B^K, E^0, and C

$$S_{13} = \Theta\left(\left(|U| + |S|\right) \times F\right); \tag{19c}$$

(4) Arrays caching the adjacency matrices: $S_{14}=\Theta(|\Lambda|)$; (19d)

(5) Arrays caching the intermediate variables:

$$S_{15} = \Theta\left(\left(|U| + |S|\right) \times F\right). \tag{19e}$$

Hence, the storage cost of training TGLFA is:

$$\begin{aligned} S_1 &= S_{11} + S_{12} + S_{13} + S_{14} + S_{15} \\ &\approx \Theta\left(F \times \max\left(\frac{|\Lambda|}{F}, |U| + |S|\right)\right), \end{aligned} \tag{20}$$

which is linear with the number of known entries in the target QoS matrix and the number of latent features in TGLFA.

According to the above analysis, the proposed TGLFA is highly efficient in computation and storage when performing representation learning in user-service QoS data. The experiments validating the model efficiency will be given in Section 4.

3.5 CUDA-based Sparse Matrix Computation

To address the high sparsity of QoS data, TGLFA adopts the data density-oriented principle in its input and output parts, i.e., only considering the known entry set rather than arbitrarily filling missing data, to reduce time and storage consumption. However, existing mainstream deep learning frameworks are immature in sparse computing, and they cannot support TGL-

FA's numerous sparse matrix operations on GPU. Hence, following previous studies [48], [49], [50], an efficient CUDA-parallelized sparse matrix computation module is specially implemented for GPU-based acceleration of TGLFA.

More specifically, as illustrated in Fig. 5, two key schedulers, i.e., SM and warp, are adopted in CUDA programming to achieve two-level parallelization of sparse matrix computation. Compressed Sparse Row (CSR) format is utilized to re-represent an incomplete user-service QoS matrix, which is constituted by three arrays: a) *ptr* array storing the row pointers, which denotes the position of each row's first entry; b) *idx* array storing the column indexes; and c) *val* array storing the values of the known data, i.e., the known QoS data. Note that given an incomplete QoS matrix Q, the size of *ptr* array is $|S|$, i.e., the number of rows or services of Q, while the size of *idx* and *val* arrays is $|\Lambda|$, i.e., the number of Q's known entries. Based on such CSR format, we achieve efficient GPU computing by designing sparse matrix operators (SMOs) [48], [49], [50], e.g., the outer product one and the row product one.

On the other hand, the block size significantly influences the performance of CUDA-parallelized computation. Since a warp consists of 32 threads, it is better to set the size of block as a multiple of 32. As GPU's various architectures have different numbers of SM, we follow prior studies [48], [49], [50] to set the block size to 32×32 in this paper.

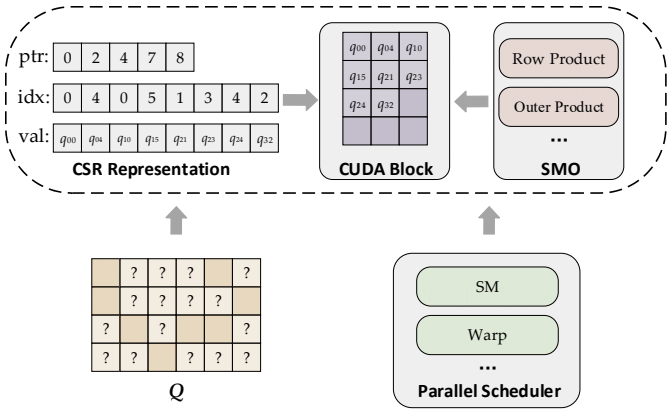


Fig. 5. Illustration of the CUDA module for sparse matrix computation.

4 Experiments and Results

In this section, we present the experimental results that reveal the effectiveness and efficiency of the proposed TGLFA. Eight large-scale testing cases have been constructed from two real-world QoS datasets to evaluate the proposed model. By comparing TGLFA with state-of-the-art methods, we intend to answer the following critical Research Questions (RQs):

RQ1. How do individual components, e.g., modules for attribute characteristics extraction, high-order connectivity information extraction, and layer combination operation, affect the representation ability of TGLFA?

RQ2. How do the multilayered design of attribute characteristics extraction and light graph convolution affect the estimation accuracy for missing QoS data and the convergence rate of TGLFA?

RQ3. How do different hyperparameter settings, e.g., the final feature combination weight (ω) and the hidden layer unit count (F), affect the performance of TGLFA?

RQ4. How does TGLFA perform when compared with state-of-the-art QoS estimators?

4.1 General Settings

Evaluation Metrics. The representation learning accuracy, i.e., missing QoS estimation accuracy is mainly considered in this study to evaluate the performance of each model. Hence, two commonly-adopted metrics, including Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) [16], [17], [42], [43], [44], [45], [46], [51], [52] are used in our experiments. Generally, they are formulated as:

$$RMSE = \sqrt{\left(\sum_{q_{s,u}\in\Psi}\left(q_{s,u}-\hat{q}_{s,u}\right)^2\right)\Big/|\Psi|},$$

$$MAE = \left(\sum_{q_{s,u}\in\Psi}\left|q_{s,u}-\hat{q}_{s,u}\right|_{abs}\right)\Big/|\Psi|,$$

where $|\Psi|$ denotes the number of entries in the testing dataset. According to the definitions above, lower RMSE and MAE indicate a model can obtain higher estimation accuracy for missing cloud service QoS data.

For evaluating the computational efficiency of different approaches, the training time of each model after GPU-acceleration on each testing case is carefully recorded. All the experiments are implemented in Python 3.7, except that the compressed sparse matrix parallel program for TGLFA is developed with CUDA C and compiled with CUDA 11.1. All empirical tests are conducted on a server with a 2.4-GHz Intel Xeon 4214R CPU, four NVIDIA RTX 3090 GPUs, and 128-GB RAM.

Datasets. Two real-world datasets for QoS estimation, *Response Time*, and *Throughput* are used in our experiments. *Response Time* records the response times of 5,825 cloud services experienced by 339 users' 1,873,838 invocations. While *Throughput* records the throughputs of 5,825 cloud services experienced by 339 users' 1,831,253 invocations. They are all user-experienced QoS affected by both server-side factors like the invocation burden and user-side factors such as the network condition. Moreover, the adopted datasets are known as the largest and most frequently-used public datasets [7], [8], [9], [20], [21], [23], [24], [25], [26], [53], [54]. Therefore, the QoS estimation results achieved on them are supposed to be highly informative and representative.

For a fair evaluation, prior studies [2], [24], [26] propose a method for dataset expansion by introducing Gaussian noise. In our experiments, we also follow this method to synthetically generate more data samples for the two testing datasets. Specifically, the generation process is three-fold: a) For each known entry $q_{s,u}$ in each dataset, a synthetic entry $g_{s,u}$ is generated by $g_{s,u}=q_{s,u}+q_{s,u}\times0.05\times rand$, where *rand* denotes a random real number generated by a standard normal distribution; b) The generation process is repeated 20 times to acquire 20 different synthetic datasets; c) By merging them, two testing datasets that are 20 times larger than the original are obtained for performance evaluation. Based on the aforementioned method, the total number of users in both *Response Time* and *Throughput* is expanded from 339 to 6,780. While the total number of invocation records in both testing sets is expanded to 37,476,760, and 36,625,060, respectively. Evidently, evaluating the performance of different models on these two large-scale datasets can better demonstrate the scalability of tested models. The characteristics of these two generated datasets are summarized in Table 2. It is worth noting that we randomly sample different testing cases containing various ratios of data splits for training, vali-

TABLE 2
PROPERTIES OF TESTING CASES.

Dataset	$\|\Lambda\|$	$\|U\|$	$\|S\|$	No.	Train:Validation:Test
Response Time (D1)	37,476,760	6,780	5,825	D1.1	1%:4%:95%
				D1.2	2%:8%:90%
				D1.3	3%:12%:85%
				D1.4	4%:16%:80%
Throughput (D2)	36,625,060	6,780	5,825	D2.1	1%:4%:95%
				D2.2	2%:8%:90%
				D2.3	3%:12%:85%
				D2.4	4%:16%:80%

dation, and test (*Train:Validation:Test*), enabling us to test all models' effectiveness comprehensively.

Models for Comparisons. Fourteen state-of-the-art QoS estimators are compared with the proposed TGLFA model in our experiments. Their nonlinearities are summarized in Table S1 of the Supplementary File (SF)[1] and the details of these methods are described as below.

M1. NeuMF [17] is a widely adopted approach to neural collaborative filtering which combines multilayered perception and inner product into matrix factorization to achieve nonlinear QoS estimation.

M2. Mult-VAE [55] is a generative LFA model based on a variational auto-encoder, which assumes data to be generated by a multinomial distribution and performs variational inference for parameter estimation.

M3. MetaMF [22] is a federated meta matrix factorization model for missing QoS data estimation, which builds a federated learning framework to generate private embeddings of users.

M4. GC-MC [56] is a graph convolutional matrix completion model, which learns the first order neighborhood information by the graph auto-encoder with node dropout.

M5. NGCF [57] is a neural graph collaborative filtering model for QoS estimation, which incorporates a complete graph neural network (GCN) into CF for integrating high-order connectivity information.

M6. LR-GCCF [33] is a collaborative filtering model based on linear residual GCN, which removes nonlinearities to enhance the performance.

M7. LightGCN [28] is a QoS estimator based on light GCN, which removes feature transformation and nonlinear activation for obtaining efficient and accurate representations.

M8. DGCN-HN [58] is a deep GCN model, which uses hybrid normalization for flexible aggregation and adopts two residual connection methods to alleviate the over-smoothing problem in QoS estimation.

M9. SGL-ED [59] is a self-supervised graph learning model for QoS data, which generates multiple node views by edge dropout and maximizes the agreement between different views to improve the accuracy and robustness.

M10. GDE [60] is a graph denoising encoder model, which captures important graph features by a band pass filter and alleviates the over-smoothing problem by indefinite-layer propagation.

M11. HMLET [27] is a hybrid method of linear and nonlinear CF. It utilizes GCN to perform latent feature analysis and adopts a gating module to select each node's best propagation method.

M12. LightGCN$_{r\text{-}AdjNorm}$ [29] is an improved light-GCN-based model, which proposes *r-AdjNorm* strategy by controlling the normalization strength to balance the recommendation accuracy and novelty.

M13. GTN [30] performs the task of collaborative QoS estimation through a graph trend filtering network, which uses the principled graph trend to address the non-adaptive propagation and non-robustness problems.

M14. GDSRec [52] is a graph-based model for decentralized collaborative filtering, which captures the statistical bias offsets by decentralized neighborhood aggregation and differentiates the social connections.

M15. TGLFA is the model proposed in this paper.

Note that the source code of the proposed TGLFA has been released on GitHub[2]. As for the source codes of other compared baselines, they can be freely downloaded from the sharing repositories.

Model Settings. The following settings are employed for each model.

(1) We initialize all the trained variables randomly with Xavier method [61], and optimize all the models with Adam [47];
(2) For models based on conventional neural networks, including M1-3, we adopt the model architectures suggested in their original papers. For all GCN-based models (M4-14), the latent feature dimension is set to 128;
(3) To achieve fair comparison, we carefully tune each model's hyperparameters to obtain the best performance. Specifically, in individually-built training and validation cases, we tune all the hyperparameters for the lowest validation error on the validation case, and then fix the same settings on the testing case to record the generalization performance;
(4) On each testing case, we repeat the splitting and training process for ten times and record the average results and corresponding standard deviations;
(5) Each model's learning process terminates if: a) the training epoch reaches a given threshold, i.e., 1000; or b) its estimation accuracy keeps decreasing for 30 epochs.

4.2 Ablation and Effectiveness Analysis (RQ1)

Comprehensive ablation studies have been conducted to verify the effectiveness of the essential components in TGLFA, including modules for extracting the **Attribute Characteristics (Module (a))** and **High-Order Connectivity Information (Module (b))**, along with the layer combination in the aforementioned modules.

4.2.1 Impact of The Proposed Modules

For testing Modules (a) and (b), we implement several variants of TGLFA, including TGLFA omitting Modules (a) and (b) (TGLFA-w/o-A&H), TGLFA omitting Module (a) (TGLFA-w/o-A), and TGLFA omitting Module (b) (TGLFA-w/o-H). The corresponding results have been summarized in Table S2 of SF. For TGLFA and all its ablated variants, we fix the layer count of Module (a) at three (K=3) and the layer count of Module (b) at ten (L=10). From these results, we obtain the following significant findings:

(1) **TGLFA's estimation accuracy for missing QoS data benefits from its correct extraction of attribute characteristics.** Compared with matrix factorization-based models, Module

[1] https://github.com/Oak-B/TGLFA/blob/main/TGLFA-Supplementary%20File.pdf
[2] https://github.com/Oak-B/TGLFA/blob/main/TGLFA-Codes.zip

(a) of TGLFA explicitly extracts the attribute characteristics in QoS data with a multilayered and fully-connected network. Module (a) enables TGLFA to learn better representations, thereby boosting its estimation accuracy for missing QoS data. As shown in Table S2, RMSE/MAE obtained by TGLFA is much lower than that of TGLFA-w/o-A on all eight testing cases. For instance, on D1.1, TGLFA achieves the best RMSE at 1.2613, which is 17.63% (i.e., $(Error_{high}-Error_{low})/Error_{high}$) lower than 1.5312 achieved by TGLFA removing Module (a). As for MAE on D1.1, TGLFA achieves it at 0.4992, which is 15.38% lower than 0.6530 of TGLFA-w/o-A. Similar results are observed in other testing cases, which highlight the importance of attribute characteristics in QoS data.

(2) **Correct extraction of high-order connectivity information enables TGLFA to achieve evident accuracy gain.** Compared with TGLFA-w/o-H, TGLFA maintains Module (b) to extract higher-order connectivity information from given QoS data with a pre-trained light graph convolutional network. The carefully-captured structural information enables TGLFA to involve multi-hop neighbors' representations when performing estimation for missing QoS data, thus enhancing the collaborative QoS signal and improving its accuracy. From the experimental results on all eight testing cases shown in Table S2 in the SF, TGLFA's RMSE/MAE is much lower than that of TGLFA-w/o-H. Taking case D1.1 as an example, TGLFA achieves the lowest RMSE at 1.2613, which is 11.88% lower than 1.4313 achieved by TGLFA-w/o-H, and the MAE at 0.4992, which is 13.44% lower than 0.5767 by TGLFA-w/o-H. Similar outcomes are achieved in the other testing cases as shown in Table S2.

(3) **The accuracy gain brought by the extraction of attribute characteristics and high-order connectivity information increases as the density of known data in the target HDI matrix decreases.** For instance, on D2.4-2.1 (whose known data density decreases from 18.98% to 4.74%) as shown in Table S2 in the SF, the RMSE values achieved by TGLFA are 0.4176, 0.4352, 0.4650, and 0.5266, which are 6.03, 6.71%, 6.93%, and 17.27% lower than that of TGLFA-w/o-A&H, 2.73%, 3.46%, 3.75%, and 8.86% lower than that of TGLFA-w/o-A, and 3.18, 4.06%, 4.69% and 11.26% lower than that of TGLFA-w/o-H, respectively. Similar results are obtained when taking MAE as the evaluating metric. Therefore, attribute characteristics, and high-order connectivity considered by the proposed TGLFA are crucial for real-world industrial applications related to QoS data. They work compatibly to enable the TGLFA model's high estimation accuracy for missing QoS data.

4.2.2 Impact of Layer Combination

To verify the effect of layer combination, we use the variants of TGLFA, omitting the layer combination operation to perform the estimation task. The experimental results are listed in Table S3 of SF, where TGLFA-Single-A/H means TGLFA does not adopt the layer combination strategy and only uses the features at the Kth layer of Module (a)/(b) to perform the estimation task. Note that for TGLFA and TGLFA-Single-A/H, we uniformly set K=3 and L=10. Given the results obtained, we have the following findings:

(1) **The layer combination strategy significantly enhances the representation ability of TGLFA in QoS data.** As shown in Table S3, summing the node features from all hidden layers in Module (a) and averaging feature matrices from all layers of the light graph convolutional network improve the estimation accuracy of missing QoS data. For instance, on D1.2, the RMSE and MAE achieved by TGLFA are 1.1986 and 0.4581, respectively. Owing to the ability to alleviate the gradient vanishing by Module (a) [58], [61], [62], the RMSE and MAE of TGLFA are 0.94% and 1.42% lower than that of TGLFA-Single-A. Due to the ability to alleviate the over-smoothing by Module (b) [28], [33], [58], the RMSE and MAE obtained by TGLFA are 13.20% and 29.62% lower than that of TGLFA-Single-H. Similar performances are observed in other testing cases.

(2) **Appropriate values of $\alpha_{1\sim K}$ and $\beta_{1\sim L}$ are convenient to discover.** It is worth noting that hyperparameters such as $\alpha_{1\sim K}$ and $\beta_{1\sim L}$ can be tuned or self-adaptation [63]. In this paper, by setting $\alpha_{1\sim K}$ at 1 and $\beta_{1\sim L}$ as $1/(L+1)$, we still achieve robust performance in the task of QoS estimation because the model parameters in E^0 implicitly learn the importance of each layer. Hence, we do not tune or optimize $\alpha_{1\sim K}$ and $\beta_{1\sim L}$ to avoid the extra computational burden. Moreover, developing an effective scheme to identify the importance of each layer is another interesting issue worthy of further investigation. We leave these issues in our future work.

4.3 Effect of Multilayered Structure (RQ2)

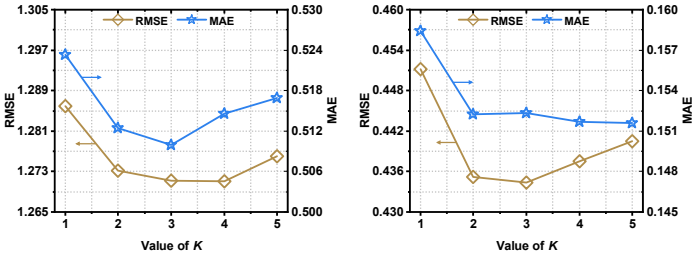
To verify whether TGLFA can benefit from the multilayered structure of attribute characteristics extraction and light graph convolution, we vary the depths of two modules, i.e., K and L, to conduct the following experiments.

4.3.1 Impact of K

We set K={1, 2, 3, 4, 5} when L is fixed to 4, and the experimental results as K varies are presented in Figs. S1-S2 in the SF. Besides, we have chosen several representative figures and presented them in Figs. 6 and 7. From them, we find that:

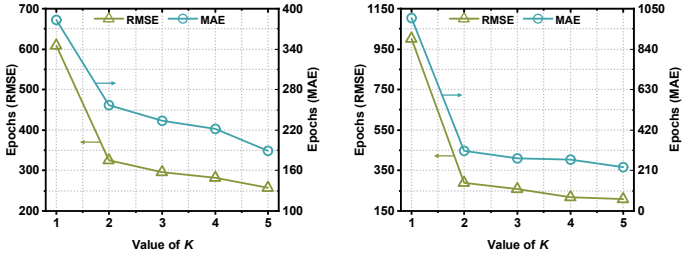
(1) **Increasing the depth of the module for attribute characteristics extraction vastly enhances TGLFA's estimation accuracy for missing QoS data.** Compared with a single-layer model, a multilayered neural network can better fit the nonlinearity hidden in the input QoS data, thus improving its representation ability. For instance, as recorded in Fig. 6(a), by fixing other hyperparameters on D1.1, as K increases from 1 to 3, the RMSE of TGLFA decreases from 1.2859 to 1.2711, which achieves an estimation error gap of 1.15%. As for MAE, it decreases from 0.5233 to 0.5099 with an estimation error gap of 2.56%. Similarly, on D2.3 (Fig. 6(b)), when K is fixed at 1, TGLFA achieves the RMSE and MAE at 0.4512 and 0.1584. However, when K is fixed, its RMSE reduces to 0.4405 with a decreasing gap of 2.37%, and the MAE reduces to 0.1516 with MAE decreasing gap of 4.29%. Similar situations can be found in other testing cases. In most cases (Fig. S1), TGLFA can achieve the lowest RMSE when K=2 or 3 and the lowest MAE when K=5. Stacking more layers may cause overfitting issues, which have to alleviate by adopting new learning objectives and regularization schemes [64], [65], [66]. It is observed from the results that the attribute characteristics are essential to improve the performance of TGLFA.

(2) **The convergence rate of TGLFA is greatly affected by the depth of modules for extracting attribute characteristics.** As recorded in Fig. S2 in the SF, its converging epoch count decreases as K increases. For instance, on D1.3, as K increas



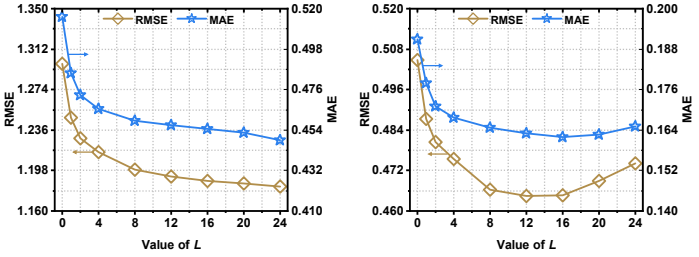
(a) Errors on D1.1 (b) Errors on D2.3

Fig. 6. Errors of TGLFA as K varies.



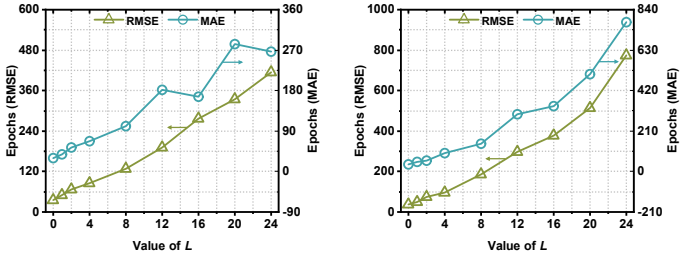
(a) Epochs on D1.3 (b) Epochs on D2.1

Fig. 7. Epochs of TGLFA as K varies.



(a) Errors on D1.2 (b) Errors on D2.2

Fig. 8. Errors of TGLFA as L varies.



(a) Epochs on D1.3 (b) Epochs on D2.3

Fig. 9. Epochs of TGLFA as L varies.

es from 1 to 5, TGLFA's converging epoch count in RMSE decrease from 609 to 257, and the converging epoch count in MAE decrease from 383 to 189. As Module (a) sums all the hidden layers as the output, the parameters stacked in multiple layers of the network can be optimized efficiently [62].

4.3.2 Impact of L

In this section, we set L={0, 1, 2, 4, 8, 12, 16, 20, 24} when K is fixed at 3. The results as L varies are depicted in Figs. S3-S4 in the SF. We have chosen several representative ones from the whole results and plotted them in Figs. 8-9. Through analyzing the presented results, we have the following findings:

(1) **Capturing the high-order connectivity information from the user-service interaction graph substantially enhances the collaborative QoS signal.** As illustrated in Figs. S3 and S4, in all eight testing cases, the estimation errors decrease greatly as L increases. For instance, as shown in Fig. 7(a), on D1.2, when L=0, the RMSE and MAE of TGLFA are 1.2981 and 0.5154. When L increases to 24, their values decrease to 1.1831 and 0.4485, meaning that the estimation error of RMSE and MAE significantly drops by 8.86% and 12.98%. Due to the correctly extracted high-order connectivity information and layer combination, TGLFA effectively addresses the over-smoothing issue and thus achieves evident performance gain. Note that the optimal L is data-dependent. Taking the estimation errors on D2.2 (Fig. 7(b)) as an example, TGLFA achieves the best RMSE when L=12 and the best MAE when L=16. Hence, it is unnecessary to set L to 24 as on D2.2. In general, in all the testing cases, TGLFA achieves the highest accuracy when L is chosen in between 12 and 24. From the analysis above, explicitly exploiting the collaborative signals from user-service QoS interactions is essential for QoS estimation tasks.

(2) **The convergence rate of TGLFA is vastly affected by the depth of the module for extracting high-order connectivity information.** For instance, as plotted in Fig. 9(a), on D1.3, as L increases from 0 to 24, the converging epoch count regarding RMSE and MAE increases from 35 and 30 to 414 and 266, respectively. Similar outcomes are also encountered in the other testing cases (see Fig. 9 and Fig. S4 in the SF). This is because all optimization parameters are in E^0, while stacking multiple layers of light graph convolution unnecessarily may slow down the overall optimization process. In general, to balance the estimation accuracy and efficiency of TGLFA, the depth of Modules (a) and (b) should be set with care, and we suggest setting K=3 and L=12 in most cases.

4.4 Analysis of Hyperparameter Sensitivity (RQ3)

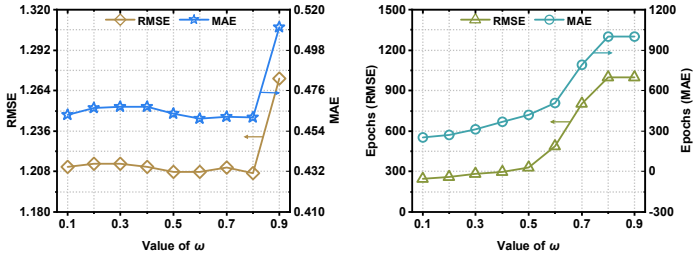
The performance of a deep learning model can be affected by many hyperparameters, such as batch size, initial learning rate, and regularization coefficient. For TGLFA, the batch size is fixed at 512, the learning rate is searched in {0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05}, and the coefficient of L_2 regularization is tuned in $\{10^{-5}, 10^{-4}, \cdots, 10^{1}, 10^{2}\}$. There are two unique hyperparameters, i.e., ω controlling feature importance and F determining the feature dimension of hidden layers. Their settings might influence the performance of TGLFA.

4.4.1 Impact of ω

We conduct a series of sensitivity tests to verify the impact of ω, and the corresponding results are depicted in Figs. S5-S6 in the SF. Herein, two examples are plotted in Fig. 10 for conciseness. By analyzing these results, we have the following findings:

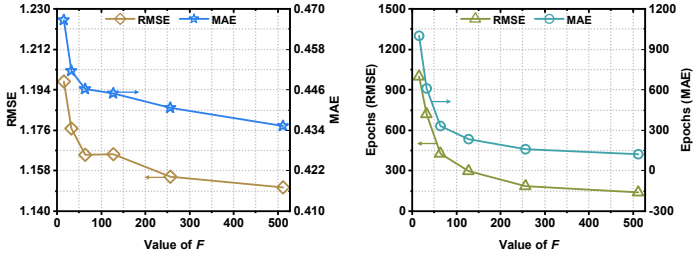
(1) **ω significantly influences the estimation accuracy of TGLFA**. As shown in Fig. S5, increasing the value of ω may greatly reduce the estimation accuracy of TGLFA. For instance, on D1.2 (Fig. 10(a)), the RMSE increases from 1.2070 to 1.2723 as ω varies from 0.8 to 0.9. For MAE, it increases from 0.4613 to 0.5106. On the other hand, TGLFA achieves steady performance when ω increases from 0.1 to 0.8. For example, when performing the task of QoS estimation on D1.2 (Fig. 10(a)), the differences regarding RMSE, and MAE are only 0.0063 and 0.0066.

(2) **The convergence rate of TGLFA is also affected by ω.** As depicted in Fig. S6, the convergence rate vastly decreases when increasing ω. Taking the case D2.1 as an example (Fig. 10(b)), the converging epoch count of TGLFA regarding the



(a) Errors on D1.2 (b) Epochs on D2.1

Fig. 10. Errors and epochs of TGLFA as ω varies.



(a) Errors on D1.3 (b) Epochs on D1.3

Fig. 11. Errors and epochs of TGLFA as F varies.

RMSE increase from 244 to 1000 when ω varies from 0.1 to 0.9. As for MAE, the epochs to converge increase from 252 to 1000. Similar trends of model convergence can also be observed in other testing cases. Based on the above analysis, it is seen that setting ω to a low value achieves both higher estimation accuracy and efficiency (e.g., ω=0.2). Note that a lower ω indicates the module for attribute characteristics extraction contributes more to output representations. This reveals attribute characteristics in QoS data are crucial for QoS estimations.

4.4.2 Impact of F

Herein, we conduct sensitivity tests with F, and the obtained results are demonstrated in Figs. S7-S8 in the SF. Note that Fig. 10 depicts two representative examples for readability. Based on these results, we summarize that:

(1) **RMSE and MAE of TGLFA significantly decrease with the increase of *F*.** A larger feature dimension of hidden layers expands the solution space of TGLFA, thereby learning more accurate representations. As illustrated in Fig. S7, the RMSE and MAE decrease in most testing cases as F increases. Taking the testing case D1.3 (Fig. 11(a)) as an example, the RMSE obtained by TGLFA decreases from 1.1977 to 1.1505, and the MAE decreases from 0.4666 to 0.4352 when F is changed from 16 to 512.

(2) **By incensing the dimensionality of the hidden units (*F*), TGLFA achieves fast convergence.** For instance, as shown in Fig. 11(b), on D1.3, when F=16, TGLFA's training epochs in RMSE and MAE are both at 1000. However, when F=512, the training epochs in RMSE decrease to 141 and the training epochs in MAE decrease to 122, respectively. Although the time cost per epoch becomes higher as F increases, larger F should be considered for improving TGLFA's efficiency.

Overall, we suggest setting ω = 0.2 and F = 128. With an appropriate learning rate (e.g., 5e-3) and regularization coefficient (e.g., 0.1), TGLFA can achieve high performance in both estimation accuracy and computational efficiency. We summarize the suggested settings of hyperparameters in Table S4 of the SF.

4.5 Comparison with State-of-the-Art Approaches (RQ4)

To validate the effectiveness of the proposed TGLFA, we compare it with fourteen state-of-the-art approaches to QoS estimation, including three approaches based on conventional neural networks and eleven GCN-based models. Tables S5-S8 of SF respectively show the RMSE, MAE, and converging time of RMSE and MAE obtained by all approaches on all testing cases.

Note that for more insights into the results presented in Tables S3-S6, we summarize the win/loss counts of TGLFA (M15) versus its peers on the second-to-last rows and record all tested models' Friedman statistical results on the last row [42], [67], [68]. With the significance level of 0.05, the hypothesis that all involved models are significantly different is accepted. Moreover, the results of the Wilcoxon signed-ranks test are presented in Tables S9 and S10, which can statistically verify whether M15 is substantially superior to other models in estimation accuracy and computational efficiency [42], [67], [68]. Wilcoxon signed-ranks test is a widely-adopted nonparametric pairwise comparison procedure, which has three important indicators, including $R+$, $R-$, and p-value [67], [68]. Among them, the higher $R+$ value stands for higher estimation accuracy (Table S9) and computational efficiency of TGLFA (Table S10), $R-$ is versa. And p-value indicates the significance level [42], [67], [68]. Based on these results, we find that:

(1) **TGLFA (M15) significantly outperforms its peers in estimation accuracy for missing QoS data.** Owing to the modeling for the attribute characteristics and high-order connectivity information, M15 achieves notable performance gain in all testing cases. For example, when testing case D1.1 (Table S5) is considered, the RMSE obtained by M15 is 1.2535, which is better than M6, M4, and M14 by 2.39%, 2.99%, and 11.87%, respectively. When MAE obtained on D1.1 (Table S6) is considered, M15 achieves 0.4948, which is better than M1-M14 by 29.69%, 11.74%, 38.66%, 2.35%, 18.63%, 12.99%, 22.96%, 29.23%, 25.41%, 34.37%, 12.80%, 22.89%, 27.89%, and 23.57%. More persuasively, M15 also obtained the lowest F-rank value and supported Wilcoxon signed-rank results. As Table S5 shows, for RMSE, the F-rank value of M15 is 1.00, which is lower than any other model in the experiment. As for MAE (Table S6), the F-rank value of TGLFA is also 1.00, which is much lower than all the compared models in the experiment. Apart from this, it can be seen in Table S9 that the $R+$ values are high, and the p-values are all within the accepted range of the hypothesis at the significance level of 0.1. Hence, it is concluded that M15 obtains significantly higher estimation accuracy than other models do.

(2) **TGLFA is more computationally efficient than other models when learning in QoS data.** As presented in Tables S7, S8, and S10, the training time for M15 to achieve the best RMSE/MAE is far less than that of M1-14, as it adopts the principle of data density-oriented modeling and light graph convolution. For instance, as shown in Table S7, M15 only spends 253 seconds on the convergence of RMSE on D1.2, which is only 0.76%, 0.87%, and 1.96% of the time used by M11, M14, and M13. On the other hand, considering the time cost in MAE, M15 only spends 228 seconds, which is about 1.88%, 64.04%, 21.55%, 17.67%, 11.83%, 12.45%, 1.96%, 1.71%, 1.45%, 9.73%, 0.70%, 1.91%, 1.40%, and 0.75% of the time used by M1-14. In other testing cases, the converging time cost of TGLFA is also less than that of other baselines. Furthermore, from the results of the Friedman test present-

ed in Tables S7 and S8, it is seen that M1-15 achieve the F-rank values on time cost to converge in RMSE are respectively 10.38, 1.63, 3.25, 6.25, 6.50, 5.00, 8.88, 10.75, 10.38, 4.63, 14.38, 9.88, 12.13, 14.63, and 1.38. And on time cost in MAE, their F-rank values are 9.75, 1.88, 4.75, 6.75, 5.50, 4.63, 8.63, 10.50, 10.75, 4.50, 14.50, 9.63, 12.75, 14.38, and 1.13. Compared with other models, M15 achieves the lowest F-rank value on training time cost in either RMSE or MAE. In Table S10, we additionally summarize the results of the Wilcoxon signed-ranks test regarding the model efficiency. The high $R+$ values of M15 versus other models and low p-values indicate that M15 has highly competitive computational efficiency compared with state-of-the-art approaches.

4.6 Summary

From the detailed experiments and analysis above, we summarize that TGLFA has the following advantages. First, TGLFA possesses high estimation accuracy for missing data in the user-service QoS matrix. Second, TGLFA is highly efficient in performing the task of QoS estimation. Third, TGLFA can well handle sparse QoS data, which greatly enhances its scalability in real-world applications. As for the limitations of TGLFA, Like other models based on graph neural networks [27], [57], [58], [59], the hyperparameters of TGLFA have to be carefully tuned, and model optimization highly relies on GPU.

5 Discussions

5.1 Connections between Interpretability and Performance of TGLFA

The proposed TGLFA is a data-driven QoS estimator, which is designed to effectively capture the features hidden in QoS data for accurate representation learning. Thus, the interpretability and effectiveness of TGLFA are determined by whether the incorporated learning modules can capture the desirable features in QoS data. It is known that TGLFA adopts a multilayer fully-connected network to learn the attribute characteristics from QoS data, utilizes a pre-trained light graph convolution network to capture the high-order connectivity information, uses the layer combination strategy to alleviate vanishing gradients and over-smoothing, and incorporates a CUDA-based computing module for fast training. The presented experimental results show that the considerations of the aforementioned learning modules can effectively capture the attribute characteristics and high-order connectivity in QoS data, thereby significantly enhancing the performance of TGLFA. It is said that the proposed TGLFA is interpretable to the connection between the model design and QoS estimation.

Also note that how to perform efficient and accurate representation learning to a highly sparse user-service matrix is a major problem of collaborative filtering for QoS estimation [2], [7], [16], [17], [26], [69], [70], [71], [72]. From the perspective of collaborative filtering, the proposed TGLFA well integrates diverse forms of information in the historical QoS data and accurately learns the representations for predicting missing QoS data. Hence, its high estimation accuracy for missing QoS data is convincing for facilitating accurate cloud service QoS estimation with confidence.

5.2 Significant Differences between TGLFA and Existing QoS Estimators

Most existing QoS estimators are based on matrix factorization (MF) [2], [7], [13], [15], [16], [26], [28], [32], which cannot learn nonlinear representations. To overcome this limitation, several estimators based on Deep Neural Networks (DNN) are subsequently developed [10], [12], [17], [18], [22], [55], [56], [57]. However, existing models mostly focus on the prediction part to implement complicated nonlinear interactions, which leads to high computational complexity. Moreover, neither an MF-based nor a DNN-based QoS estimator can well handle high-order connectivity and attribute characteristics information.

In contrast, the proposed TGLFA constructs Module (a) to learn the attribute characteristics and Module (b) to capture the high-order connectivity information. It incorporates the nonlinearities into Module (a) and removes the activations in graph convolution to enjoy a fast and accurate nonlinear learning process. Moreover, TGLFA adopts the principle of data density-oriented modeling to achieve high computational efficiency. According to the empirical studies, the proposed model achieves significant accuracy and efficiency gains when compared with state-of-the-art QoS estimators [17], [22], [27], [29], [30], [32], [33], [52], [55], [56], [57], [58], [59], [60].

5.3 Distinctive contributions

The proposed TGLFA is not a straightforward fusion of existing methods. Existing approaches to learning representation in QoS data are mainly developed based on linear learning techniques, e.g., matrix factorization [2], [7], [26]. They cannot appropriately model the higher-order connectivity and attribute characteristics information. In contrast, the proposed TGLFA carefully considers the above information within the GCN framework. Based on these observations, the distinctive contributions of this paper lie in the following aspects.

(1) TGLFA is the first approach that considers both attribute characteristics and high-order connectivity information in QoS data to perform the task of QoS estimation. TGLFA carefully builds the modules for extracting these two types of information (Modules (a) and (b)), which can learn accurate representations for the prediction module (Module (c)).
(2) The input and output layers of TGLFA are built according to the principle of data density-oriented modeling. Besides, the CUDA-based module for sparse matrix computation is developed. These amendments related to learning objectives and optimization enable TGLFA to train in a highly efficient manner.
(3) In Module (b), a pre-trained light graph convolutional network is proposed to effectively capture the high-order connectivity information in QoS data. Unlike existing techniques, it removes nonlinear activation and feature transformation, which significantly simplifies the network structure of classical graph convolutional networks. Therefore, accurate representations can be efficiently learned from the user-service invocation bipartite graph.
(4) An effective strategy for layer combination is proposed to use in Modules (a) and (b) to alleviate the issues of vanishing gradients and over-smoothing.

According to the experimental results, the above ideas enable TGLFA to achieve high estimation accuracy and training efficiency when predicting missing QoS data. Compared with state-of-the-art models that are based on matrix factorization, deep neural networks, and graph convolutional networks, it achieves significant gains in accuracy and efficiency. Overall, this study can provide researchers and engineers in the service computing community with novel methodologies to perform QoS research and novel insights for learning in QoS data and

other bipartite graphs.

6 Conclusions

In this paper, a novel model, labeled as TGLFA, is proposed to learn highly accurate representations in QoS data. Different from previous approaches, TGLFA considers extracting attribute characteristics of cloud services by constructing a multi-layered and fully-connected network and acquiring higher-order connectivity information by designing a pre-trained light graph convolutional network. It further incorporates the principle of data density-oriented modeling into the learning process for efficiency. To the best of our knowledge, existing approaches have never considered such learning paradigms. Extensive experiments on two real-world datasets have been conducted to validate the effectiveness of the proposed TGLFA. Ablation studies are further performed to verify the significance of explicitly encoding attribute characteristics and high-order connectivity in the task of QoS data estimation. The notable results demonstrate that TGLFA significantly outperforms state-of-the-art approaches to QoS estimation in both accuracy and efficiency.

In the future, we will try to improve TGLFA in the following ways. First, it is of high interest to achieve more effective interactions between the latent features of users and services when performing QoS data estimations [10], [63], [70], [73], [74], [75], and other industrial applications, e.g., big data analytics in healthcare [76], [77]. Second, self-adaption mechanisms can be developed to enable TGLFA to distinguish the importance of neighbors in different orders [78], [79]. Last, more forms of trust-worthy data, e.g., Service Level Agreement (SLA), and cloud service invocations with diverse metrics, can be used to augment TGLFA. This enables TGLFA to be effective in various learning tasks in graph data [18], [69], [70], [82], [83], and directly identify key factors in QoS, e.g., trusted services providers [80], [81].

Acknowledgments

The authors would like to express their sincere gratitude to editors and reviewers for their careful review and constructive comments. The authors would like to express their sincere gratitude to Mr. Fei Luo for his assistance in the programming of TGLFA's sparse matrix computation.

This research is supported in part by the National Natural Science Foundation of China under Grant 62272078, and in part by the CAAIHuawei MindSpore Open Fund under Grant CAAIXSJLJJ-2021-035A.

References

[1] Y. Yang, Z. Li, J. Zhang, and Y. Chen, "Transfer learning for web services classification," in *Proc. of 2021 IEEE Int. Conf. on Web Services*, Chicago, USA, 2021, pp. 185-191.

[2] D. Wu, Q. He, X. Luo, M. Shang, Y. He, and G. Wang, "A posterior-neighborhood-regularized latent factor model for highly accurate web service QoS prediction," *IEEE Trans. on Services Computing*, DOI: 10.1109/TSC.2019.2961895, 2019.

[3] R. Lu, X. Jin, S. Zhang, M. Qiu, and X. Wu, "A study on big knowledge and its engineering issues," *IEEE Trans. on Knowledge and Data Engineering*, vol. 31, no. 9, pp. 1630-1644, 2019.

[4] S. Li, H. Luo, and G. Zhao, "Bi-HPTM: an effective semantic matchmaking model for web service discovery," in *Proc. of 2020 IEEE Int. Conf. on Web Services*, Beijing, China, 2020, pp. 433-440.

[5] S. Badsha, X. Yi, L. Khalil, D. Liu, S. Nepal, E. Bertino, and K. Y. Lam, "Privacy preserving location-aware personalized web service recommendations," *IEEE Trans. on Services Computing*, vol. 14, no. 3, pp. 791-804, 2021.

[6] Y. Xiao, G. Kang, J. Liu, B. Cao, and L. Ding, "WSGCN4SLP: weighted signed graph convolutional network for service link prediction," in *Proc. of the 2021 IEEE Int. Conf. on Web Services*, Chicago, USA, 2021, pp. 135-144.

[7] X. Luo, M. Zhou, S. Li, Y. Xia, Z. You, Q. Zhu, and H. Leung, "Incorporation of efficient second-order solvers into latent factor models for accurate prediction of missing QoS data," *IEEE Trans. on Cybernetics*, vol. 48, no. 4, pp. 1216-1228, 2018.

[8] G. N. Rai, and G. R. Gangadharan, "Model checking based web service verification: a systematic literature review," *IEEE Trans. on Services Computing*, vol. 14, no. 3, pp. 747-764, 2021.

[9] P. Zhang, H. Jin, H. Dong, W. Song, and L. Wang, "LA-LMRBF: online and long-term web service QoS forecasting," *IEEE Trans. on Services Computing*, vol. 14, no. 6, pp. 1809-1823, 2021.

[10] L. Ding, G. Kang, J. Liu, Y. Xiao, and B. Cao, "QoS prediction for web services via combining multi-component graph convolutional collaborative filtering and deep factorization machine," in *Proc. of the 2021 IEEE Int. Conf. on Web Services*, Chicago, USA, 2021, pp. 551-559.

[11] M. Karakus, E. Guler, and S. Uludag, "QoSChain: provisioning inter-as QoS in software-defined networks with blockchain," *IEEE Trans. on Network and Service Management*, vol. 18, no. 2, pp. 1706-1717, 2021.

[12] J. Liu, and Y. Chen, "HAP: a hybrid QoS prediction approach in cloud manufacturing combining local collaborative filtering and global case-based reasoning," *IEEE Trans. on Services Computing*, vol. 14, no. 6, pp. 1796-1808, 2021.

[13] H. Wu, and X. Luo, "Instance-frequency-weighted regularized, nonnegative and adaptive latent factorization of tensors for dynamic QoS analysis," in *Proc. of the 2021 IEEE Int. Conf. on Web Services*, Chicago, USA, 2021, pp. 560-568.

[14] H. Sun, C. Peng, D. Yue, Y. L. Wang, and T. Zhang, "Resilient load frequency control of cyber-physical power systems under QoS-dependent event-triggered communication," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 4, pp. 2113-2122, 2020.

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

[16] D. Wu, M. Shang, X. Luo, and Z. Wang, "An L_1-and-L_2-norm-oriented latent factor model for recommender systems," *IEEE Trans. on Neural Networks and Learning Systems*, DOI: 10.1109/TNNLS.2021.3071392, 2021.

[17] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. of the 26th Int. Conf. on World Wide Web*, Perth, Australia, 2017, pp. 173-182.

[18] Z. Zheng, X. Li, M. Tang, F. Xie, and M. R. Lyu, "Web service QoS prediction via collaborative filtering: a survey," *IEEE Trans. on Services Computing*, DOI: 10.1109/TSC.2020.2995571, 2020.

[19] D. Ryu, K. Lee, and J. Baik, "Location-based web service QoS prediction via preference propagation to address cold start problem," *IEEE Trans. on Services Computing*, vol. 14, no. 3, pp. 736-746, 2021.

[20] X. Luo, Z. Liu, L. Jin, Y. Zhou, and M. Zhou, "Symmetric nonnegative matrix factorization-based community detection models and their convergence analysis," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 33, no. 3, pp. 1203-1215, 2022.

[21] X. Zhu, X. Jing, D. Wu, Z. He, J. Cao, D. Yue, and L. Wang, "Similarity-maintaining privacy preservation and location-aware low-rank matrix factorization for QoS prediction based web service recommendation," *IEEE Trans. on Services Computing*, vol. 14, no. 3, pp. 889-902, 2021.

[22] Y. Lin, P. Ren, Z. Chen, Z. Ren, D. Yu, Jun Ma, M. Rijke, and X. Cheng, "Meta matrix factorization for federated rating predictions," in *Proc. of the 43rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieva*', Xi'an, China, 2020, pp. 981-990.

[23] Y. Zhang, P. Zhang, Y. Luo, and J. Luo, "Efficient and privacy-preserving federated QoS prediction for cloud services," in *Proc. of 2020 IEEE Int. Conf. on Web Services*, Beijing, China, 2020, pp. 549-553.

[24] X. Luo, M. Zhou, Z. Wang, Y. Xia, and Q. Zhu, "An effective scheme for QoS estimation via alternating direction method-based matrix factorization," *IEEE Trans. on Services Computing*, vol. 12, no. 4, pp. 503-518, 2019002E

[25] Y. Yang, Z. Zheng, X. Niu, M. Tang, Y. Lu, and X. Liao, "A location-based factorization machine model for web service QoS prediction," *IEEE Trans. on Services Computing*, vol. 14, no. 5, pp. 1264-1277, 2021.

[26] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, "A data-characteristic-aware latent factor model for web services QoS prediction," *IEEE Trans. on Knowledge and Data Engineering*, vol. 34, no. 6, pp. 2525-2538, 2022.

[27] T. Kong, T. Kim, J. Jeon, J. Choi, Y.-C. Lee, N. Park, and S.-W. Kim, "Linear, or non-linear, that is the question," in *Proc. of the 15th ACM Int. Conf. on Web Search and Data Mining*, Tempe, USA, 2022, pp. 517-525.

[28] X. He, K. Deng, X. Wang, Y. li, Y. Zhang, and M. Wang, "LightGCN: simplifying and powering graph convolution network for recommendation," in *Proc. of the 43rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieva'*, Xi'an, China, 2020, pp. 639-648.

[29] M. Zhao, L. Wu, Y. Liang, L. Chen, J. Zhang, Q. Deng, K. Wang, X. Shen, T. Lv, and R. Wu, "Investigating accuracy-novelty performance for graph-based collaborative filtering," in *Proc. of the 45th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Madrid, Spain, 2022, pp. 50-59.

[30] W. Fan, X. Liu, W. Jin, X. Zhao, J. Tang, and Q. Li, "Graph trend filtering networks for recommendation," in *Proc. of the 45th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Madrid, Spain, 2022, pp. 112-121.

[31] K. Mao, J. Zhu, X. Xiao, B. Lu, Z. Wang, and X. He, "UltraGCN: ultra simplification of graph convolutional networks for recommendation," in *Proc. of the 30th ACM Int. Conf. on Information and Knowledge Management*, Queensland, Australia, 2021, pp. 1253-1262.

[32] F. Wu, A. H. S. Júnior, T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proc. of the 36th Int. Conf. on Machine Learning*, Long Beach, USA, 2019, pp. 6861-6871.

[33] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph based collaborative filtering: a linear residual graph convolutional network approach," in *Proc. of the 34th AAAI Conf. on Artificial Intelligence*, New York, USA, 2020.

[34] X. Shi, Q. He, X. Luo, Y. Bai, and M. Shang, "Large-scale and scalable latent factor analysis via distributed alternative stochastic gradient descent for recommender systems," *IEEE Trans. on Big Data*, vol. 8, no. 2, pp. 420-431, 2022.

[35] Z. Wang, H. Zhao, and C. Shi, "Profiling the design space for graph neural networks based collaborative filtering," in *Proc. of the 15th ACM Int. Conf. on Web Search and Data Mining*, Tempe, USA, 2022, pp. 1109-1119.

[36] W. Yu, Z. Zhang, and Zheng Qin, "Low-pass graph convolutional network for recommendation," in *Proc. of the 36th AAAI Conf. on Artificial Intelligence*, Virtual Event, 2022, pp. 8954-8961.

[37] Z. Liu, Q. Sheng, X. Xu, D. Chu, and W. Zhang, "Context-aware and adaptive QoS prediction for mobile edge computing services," *IEEE Trans. on Services Computing*, vol. 15, no. 1, pp. 400-413, 2022.

[38] H. Wu, X. Luo, and M. Zhou, "Advancing non-negative latent factorization of tensors with diversified regularization schemes," *IEEE Trans. on Services Computing*, vol. 15, no. 3, pp. 1334-1344, 2022.

[39] J. Li, H. Wu, J. Chen, Q. He, and C.-H. "su, "Topology-aware neural model for highly accurate QoS prediction," *IEEE Trans. on Parallel and Distributed Systems*, vol. 33, no. 7, pp. 1538-1552, 2022.

[40] P. Zhang, H. Jin, H. Dong, and W. Song, "M-BSRM: multivariate Bayesian runtime QoS monitoring using point mutual information," *IEEE Trans. on Services Computing*, vol. 15, no. 1, pp. 484-497, 2022.

[41] J. Falk, H. Geppert, F. Dürr, S. Bhowmik, and K. Rothermel, "Dynamic QoS-aware traffic planning for time-triggered flows in the real-time data plane," *IEEE Trans. on Network and Service Management*, vol. 19, no. 2, pp. 1807-1825, 2022.

[42] X. Luo, Z. Liu, M. Shang, J. Lou, and M. Zhou, "Highly-accurate community detection via pointwise mutual information-incorporated symmetric non-negative matrix factorization," *IEEE Trans. on Network Science and Engineering*, vol. 8, no. 1, pp. 463-476, 2021.

[43] Y. Tian, C. Zhang, Z. Guo, C. Huang, R. A. Metoyer, and N. V. Chawla, "RecipeRec: a heterogeneous graph learning model for recipe recommendation," in *Proc. of the 31th Int. Joint Conf. on Artificial Intelligence*, Vienna, Austria, 2022, pp. 3466-3472.

[44] Z. Cao, Y. Zhang, J. Guan, S. Zhou, and G. Chen, "Link weight prediction using weight perturbation and latent factor," *IEEE Trans. on Cybernetics*, vol. 52, no. 3, pp. 1785-1797, 2022.

[45] A. Baggag, S. Abbar, A. Sharma, T. Zanouda, A. AlHomaid, A. Mohan, and J. Srivastava, "Learning spatiotemporal latent factors of traffic via regularized tensor factorization: imputing missing values and forecasting," *IEEE Trans. on Knowledge and Data Engineering*, vol. 33, no. 6, pp. 2573-2587, 2021.

[46] D. Wu, Y. He, X. Luo, and M. Zhou, "A latent factor analysis-based approach to online sparse streaming feature selection," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, DOI: 10.1109/TSMC.2021.3096065, 2021.

[47] D. P. Kingma, and J. Ba, "Adam: a method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[48] J. Lee, S. Kang, Y. Yu, Y.-Y. Jo, S.-W. Kim, and Y. Park, "Optimization of GPU-based sparse matrix multiplication for large sparse networks," in *Proc. of the 36th IEEE Int. Conf. on Data Engineering*, Dallas, USA, 2020, pp. 925-936.

[49] S. Pal, J. Beaumont, D.-H. Park, A. Amarnath, S. Feng, C. Chakrabarti, H.-S. Kim, D. Blaauw, T. N. Mudge, and R. G. Dreslinski, "Outerspace: an outer product based sparse matrix multiplication accelerator," in *Proc. of the 24th IEEE Int. Symposium on High Performance Computer Archi-tecture*, Vienna, Austria, 2018, pp. 724-736.

[50] M. Fey, and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," *arXiv preprint arXiv:1903.02428*, 2019.

[51] Z. Li, S. Li, and X. Luo, "An overview of calibration technology of industrial robots," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 1, pp. 23-36, 2021.

[52] J. Chen, X. Xin, X. Liang, X. He, and J. Liu, "GDSRec: Graph-Based Decentralized Collaborative Filtering for Social Recommendation," *IEEE Trans. on Knowledge and Data Engineering*, DOI: 10.1109/TKDE.2022.3153284, 2022.

[53] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "WSRec: a collaborative filtering based web service recommender system," in *Proc. of the 2009 IEEE Int. Conf. on Web Services*, Los Angeles, USA, 2009, pp. 437-444.

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

[55] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proc. of the 27th Int. Conf. on World Wide Web*, Lyon, France, 2018, pp. 689-698.

[56] R. Berg, T. N. Kipf, and W. Max, "Graph convolutional matrix completion," in *Proc. of the 24th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, London, UK, 2018.

[57] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proc. of the 42nd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Paris, France, 2019, pp. 165-174.

[58] W. Guo, Y. Yang, Y. Hu, C. Wang, H. Guo, Y. Zhang, R. Tang, W. Zhang, and X. He, "Deep graph convolutional networks with hybrid normalization for accurate and diverse recommendation", in *Proc. of 3rd Workshop on Deep Learning Practice for High-Dimensional Sparse Data with KDD*, Virtual Event, China, 2021.

[59] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *Proc. of the 44th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Virtual Event, Canada, 2021, pp. 726-735.

[60] S. Peng, K. Sugiyama, and T. Mine, "Less is more: reweighting important spectral graph features for recommendation," in *Proc. of the 45th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Madrid, Spain, 2022, pp. 1273-1282.

[61] X. Glorot, and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. of the 13th Int. Conf. on Artificial Intelligence and Statistics*, Sardinia, Italy, 2010, pp. 249-256.

[62] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," in *Proc. of the 29th IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, USA, 2016, pp. 770-778.

[63] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, T-S. Chua, "Attentive collaborative filtering: multimedia recommendation with item- and component-level attention," in *Proc. of the 40th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Tokyo, Japan, 2017, pp. 335-344.

[64] Z. Li, B. Qiao, B. Wen, Y. Wang, and X. Chen, "Acoustic mode measuring approach developed on generalized minimax-concave regularization and Tikhonov regularization," *IEEE Trans. on Instrumentation and Measurement*, vol. 71, pp. 1-11, 2022.

[65] Y. Lu, Z. Zhang, G. Lu, Y. Zhou, J. Li, and D. Zhang, "Addi-Reg: a better generalization-optimization tradeoff regularization method for convolutional neural networks," *IEEE Trans. on Cybernetics*, vol. 52, no. 10, pp. 10827-10842, 2022.

[66] K. Fatras, B. B. Damodaran, S. Lobry, R. Flamary, D. Tuia, and N. Courty, "Wasserstein adversarial regularization for learning with label noise," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 7296-7306, 2022.

[67] Di Wu, Long Jin, and X. Luo, "PMLF: prediction-sampling-based multilayer-structured latent factor analysis," in *Proc. of the 20th IEEE Int. Conf. on Data Mining*, Sorrento, Italy, 2020, pp. 671-680.

[68] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1-30, 2006.

[69] K. Sangsanit, W. Kurutach, and S. Phoomvuthisarn, "REST Web service composition: a survey of automation and techniques," in *Proc. of the 2018 Int. Conf. on Information Networking*, Chiang Mai, Thailand, 2018, pp. 116-121.

[70] O. Kondratyeva, N. Kushik, A. Cavalli, and N. Yevtushenko, "Evaluating quality of Web services: a short survey," in *Proc. of the 2013 IEEE Int. Conf. on Web Services*, Santa Clara, USA, 2013, pp. 587-594.

[71] W. Huang, P. Zhang, Y. Chen, M. Zhou, Y. Al-Turki, and A. Abu-

sorrah, "QoS prediction model of cloud services based on deep learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 3, pp. 564-566, 2022.

[72] 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.

[73] Z. Li, L. Tang, and J. Liu, "A memetic algorithm based on probability learning for solving the multidimensional knapsack problem," *IEEE Trans. on Cybernetics*, vol. 52, no. 4, pp. 2284-2299, 2022.

[74] Y. Weng, X. Chen, L. Chen, and W. Liu, "GAIN: graph attention & interaction network for inductive semi-supervised learning over large-scale graphs," *IEEE Trans. on Knowledge and Data Engineering*, vol. 34, no. 9, pp. 4257-4269, 2022.

[75] F. Chen, P. Li, T. Miyazaki, and C. Wu, "FedGraph: federated graph learning with intelligent sampling," *IEEE Trans. on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1775-1786, 2022.

[76] S. Imran, T. Mahmood, A. Morshed, and T. Sellis, "Big data analytics in health–are - a systematic literature review and roadmap for practical implementation," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 1, pp. 1-22, 2021.

[77] W. Yue, Z. Wang, J. Zhang, and X. Liu, "An overview of recommendation techniques and their applications in healthcare," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 4, pp. 701-717, 2021.

[78] C. L. Camacho-Villalón, M. Dorigo, and T. Stützle, "PSO-X: a component-based framework for the automatic design of particle swarm optimization algorithms," *IEEE Trans. on Evolutionary Computation*, vol. 26, no. 3, pp. 402-416, 2022.

[79] Y. He, W. Cai, P. Zhou, G. Sun, S. Luo, H. Yu, and M. Guizani, "Beamer: stage-aware coflow scheduling to accelerate hyper-parameter tuning in deep learning clusters," *IEEE Trans. on Network and Service Management*, vol. 19, no. 2, pp. 1083-1097, 2022.

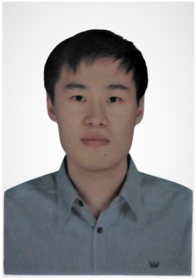
[80] F. Li, G. White, and S. Clarke, "A trust model for SLA negotiation candidates selection in a dynamic IoT environment," *IEEE Trans. on Services Computing*, vol. 15, no. 5, pp. 2565-2578, 2022.

[81] H. Chergui, L. Blanco, and C. Verikoukis, "Statistical federated learning for beyond 5G SLA-constrained RAN slicing," *IEEE Trans. on Wireless Communications*, vol. 21, no. 3, pp. 2066-2076, 2022.

[82] Y. Chen, X. Xiao, C. Peng, G. Lu, and Y. Zhou, "Low-rank tensor graph learning for multi-view subspace clustering," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 32, no. 1, pp. 92-104, 2022.

[83] Y. Qiu, G. Zhou, Y. Wang, Y. Zhang, and S. Xie, "A generalized graph regularized non-negative tucker decomposition framework for tensor data representation," *IEEE Trans. on Cybernetics*, vol. 52, no. 1, pp. 594-607, 2022.

Fanghui Bi received the B.S. degree in Software Engineering from the Northeastern University, Shenyang, China, in 2020. He is currently pursuing his Ph.D. degree in computer science at the University of Chinese Academy of Sciences, Beijing, China. His research interests include Big Data Analytics and Graph Neural Networks.



Tiantian He (*Member*, IEEE) received the Ph.D. degree in Computer Science from the Hong Kong Polytechnic University in 2017. Currently He is a Research Scientist in Center for Frontier AI Research (CFAR), Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR). His research interests include artificial intelligence, machine learning, data-centric transfer optimization, and data mining.

Yuetong Xie is currently studying for her bachelor's degree in Intelligent Technology and Services at the Faculty of Data Science, City University of Macau. Her research interests are focusing on the big data analysis and computational intelligence.

Xin Luo (*Senior Member*, IEEE) 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 the Beihang University, Beijing, China, in 2011. He is currently a Professor of Data Science and Computational Intelligence with the College of Computer and Information Science, Southwest University, Chongqing, China. He has authored or coauthored over 200 papers (including over 100 IEEE Transactions papers) in the areas of his interests. His research interests focus on big data analysis and graph learning. Dr. Luo was the recipient of the Outstanding Associate Editor Award from IEEE/CAA JOURNAL OF AUTOMATICA SINICA in 2020. He is currently serving as a Deputy-Editor-in-Chief for IEEE/CAA JOURNAL OF AUTOMATICA SINICA, and an Associate Editor for IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS. His Google page: https://scholar.google.com/citations?user=hyGlDs4AAAAJ&hl=zh-TW.