A Two-Stream Light Graph Convolution Network-based Latent Factor Model for Accurate Cloud Service QoS Estimation

Fanghui Bi
Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences
Chongqing School, University of Chinese Academy of Sciences
Chongqing, China
bifanghui@cigit.ac.cn

Tiantian He
Center for Frontier AI Research, Institute of High Performance Computing
*Agency for Science, Technology and Research (A*STAR)*
Singapore
he_tiantian@ihpc.a-star.edu.sg

Xin Luo
College of Computer and Information Science, Southwest University
Chongqing, China
luoxin@swu.edu.cn

***Abstract*—Historical Quality-of-Service (QoS) data regarding past user-service invocations are vital to understand the user behaviors and cloud service conditions. A Matrix Factorization (MF)-based Collaborative Filtering (CF) model has proven to be highly effective in performing representation learning to such QoS data. However, its performance is hindered by its linear interaction and implicit encoding of collaborative QoS signal. To address this critical issue, this paper presents a Two-stream Light Graph Convolution Network-based latent factor (TLGCN) model with the three-fold ideas: 1) constructing a multilayered and fully-connected network to represent services' nonlinear latent features; 2) integrating the user-service interactions, i.e., the bipartite graph structure into the representation learning process with a light graph convolution network for illustrating the high-order connectivity information in QoS data; and 3) incorporating the data density-oriented modeling mechanism into the input and output of TLGCN for high computational efficiency. Experimental results on two real QoS datasets demonstrate that the proposed TLGCN model significantly outperforms its state-of-the-art peers in both estimation accuracy for missing QoS data and computational efficiency.**

Keywords*—Quality-of-Service, Representation Learning, Data Science, Cloud Service, Missing Data Estimation, Graph Neural Network, Non-Euclidean Data.

I. Introduction

In industrial software applications based on service-oriented architectures [1, 2], cloud services are taken as the fundamental components to achieve easy exchange of data among them over the World Wide Web. With this era of cloud computing, more and more service providers serve their customers by deploying cloud services, i.e., the number of online cloud services are explosively increasing [3, 4]. And owing to the highly similar functionality of available candidate cloud services, it becomes a vital challenge for users to select appropriate ones and build a reliable system [4, 5].

Most nonfunctional characteristics of cloud services can be reflected by Quality-of-Service (QoS) data at both server and user sides [1, 2], which is critical to service selection. Server-side QoS data, e.g., popularity and price, can be directly provided by service providers. On the other hand, user-side data, e.g., response time and throughput, vary greatly among different users depending on many factors [4, 5], e.g., invoking environment. Conducting warming-up tests is a straightforward approach to retrieve user-side QoS data by actually invoking each cloud service [1-3]. However, owing to the large number of candidate cloud services and charged commercial service invocations in most cases, to perform such warming-up tests is very time-consuming and expensive thereby impractical.

Motivated by the success of Collaborative Filtering (CF) in e-commerce, researchers employ this technique to implement QoS estimation aiming at accurately estimating missing QoS data based on historical ones [1-3, 6-12]. 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 owing to the fact that a user typically experiences just a very limited number of candidate cloud services [1-3, 16]. Hence, how to perform efficient and accurate representation learning on such a highly sparse user-service matrix is the major problem of CF-based QoS estimation.

Among various CF-based QoS estimators, a Matrix Factorization (MF)-based Latent Factor Analysis (LFA) model has proven to be effective in performing representation learning to QoS data [2, 6-12]. It works by mapping 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, and then perform inner products of corresponding latent features of users and cloud services to obtain the missing entries. Zhu *et al.* [6] incorporate a similarity-maintaining privacy preservation strategy into their location-aware low-rank matrix factorization model to achieve reliable and accurate QoS estimation. Luo *et al.* [7] adopt the principle of alternating direction method and ensemble mechanism to build an effective matrix factorization-based QoS estimator. Yang *et al.* [8] consider the correlation between users and services, i.e., incorporate their neighborhood information into the factorization process for accurate QoS estimation. Ryu *et al.* [9] propose a matrix factorization-based

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 (*Corresponding Author: T. He*).

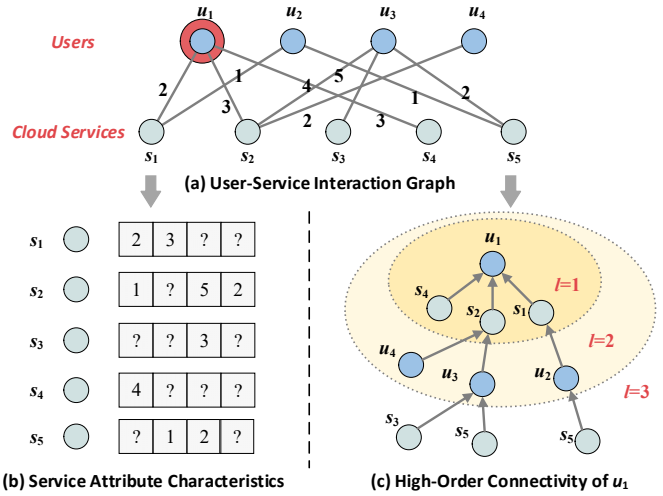


Fig. 1. An illustrative example of the attribute characteristics (e.g., response time and throughput) and high-order connectivity information in a user-service interaction graph.

preference propagation approach to use the fused invocation and neighborhood similarity information for addressing the cold start problem in QoS estimation.

Despite their effectiveness, existing MF-based LFA methods possess inherent deficiencies. As depicted in Fig. 1, the user-service interaction graph contains the attribute characteristics of users/services and high-order connectivity information. MF-based LFA models adopt inner product to combine the multiplication of the latent feature vectors of users and services, whose linear nature limits to fully capture the implicit information in them. And they cannot well handle the non-Euclidean structure in QoS data, resulting in extremely sparse collaborative QoS signals and inevitable accuracy loss.

To address the above issues, this paper presents a **T**wo-stream **L**ight **G**raph **C**onvolution **N**etwork-based latent factor (TLGCN) model for accurate and efficient representation learning to QoS data with the three-fold ideas:

(1) Constructing a cloud service-oriented multilayered and fully-connected network to learn high level of services' nonlinear latent features;
(2) Integrating the user-service interactions, i.e., the bipartite graph structure into the representation learning process with a light graph convolution network for illustrating high-order connectivity information;
(3) Incorporating the data density-oriented modeling principle into the input and output of TLGCN for high efficiency.

To summarize, this paper achieves the main contributions:

(1) Proposing a TLGCN model with high representation learning ability for cloud service QoS data in both estimation accuracy and computational efficiency;
(2) Conducting extensive empirical studies on two commonly-adopted real QoS datasets to evaluate the TLGCN model.

To the best of our knowledge, the TLGCN model significantly outperforms several its state-of-the-art peers in both estimation accuracy for missing QoS data and computational efficiency.

The remainder of this paper is organized as below. Section II presents the preliminaries. Section III describes the TLGCN model. Section IV gives the experimental results. In the end, Section V concludes this paper.

II. Preliminaries

A QoS matrix describing the relationship among user set and service set is the fundamental input for an LFA-based QoS estimator, which is defined as [2, 5-9]:

***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 $s\in S$.

Since a user usually invokes only a very limited number of candidate cloud services, Q is incomplete. Let Λ and O denote the known entry set and unknown one, an LFA-based QoS estimator is defined as [2, 5-9]:

***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 specified $s\in S$ and $u\in U$ such that $\sum_{q_{s,u}\in\Lambda}(q_{s,u}-\hat{q}_{s,u})^2$ is minimized.

III. A TLGCN Model

In this section, we introduce the proposed TLGCN model, which can be divided into User-oriented TLGCN (U-TLGCN) and Service-oriented TLGCN (S-TLGCN). Note that they enjoy the same structure and the only difference between them is that 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. Hence, in this paper, we mainly present the Service-oriented TLGCN model for better readability and conciseness, whose structure and flowchart are illustrated in Fig. 2 and details are as below.

A. Attribute Characteristics Extraction Module

We design the attribute characteristics extraction module to explicitly learn accurate nonlinear node representations from the known invocation records, and incorporate the data density-oriented modeling mechanism into it to accommodate the sparsity of QoS data. Fig. 2(a) illustrates the whole architecture.

Input Layer. The input layer receives service-oriented QoS data, i.e., a $|S|\times|U|$ QoS matrix Q regarded as the attribute characteristics of services. Since the real-world QoS data are mostly highly sparse, the prior methods filling the unknown parts with artificial values, e.g., zero values, are time-consuming and can cause accuracy loss. Hence, following the data density-oriented principle, TLGCN only activates the input layer nodes based on the known user-service interactions to achieve high efficiency.

Hidden Layer. The attribute characteristics extraction module achieves multiple fully-connected hidden layers to learn high level of nonlinear latent features. As shown in Fig. 2(a), H^k denotes the k-th layer latent feature matrix, W^k and B^k denote the corresponding weight matrix and bias vector in k-th layer. Note that the k-th layer node count F^k is set uniformly for simplicity, i.e., $F^1=F^2=\cdots=F^K=F$. For the first layer, we present the mapping formula 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 the single entries in Q, H^1, W^1 and B^1, $\Lambda(s)$ represents the known entry subset related to s, and $a_1(\cdot)$ indicates the first-layer activation function, such as Sigmoid. Note that since $|\Lambda(s)|\ll|U|$, (2) reduces the computational and storage cost greatly. For the (2~K)-th hidden layers, the computing process is given as:

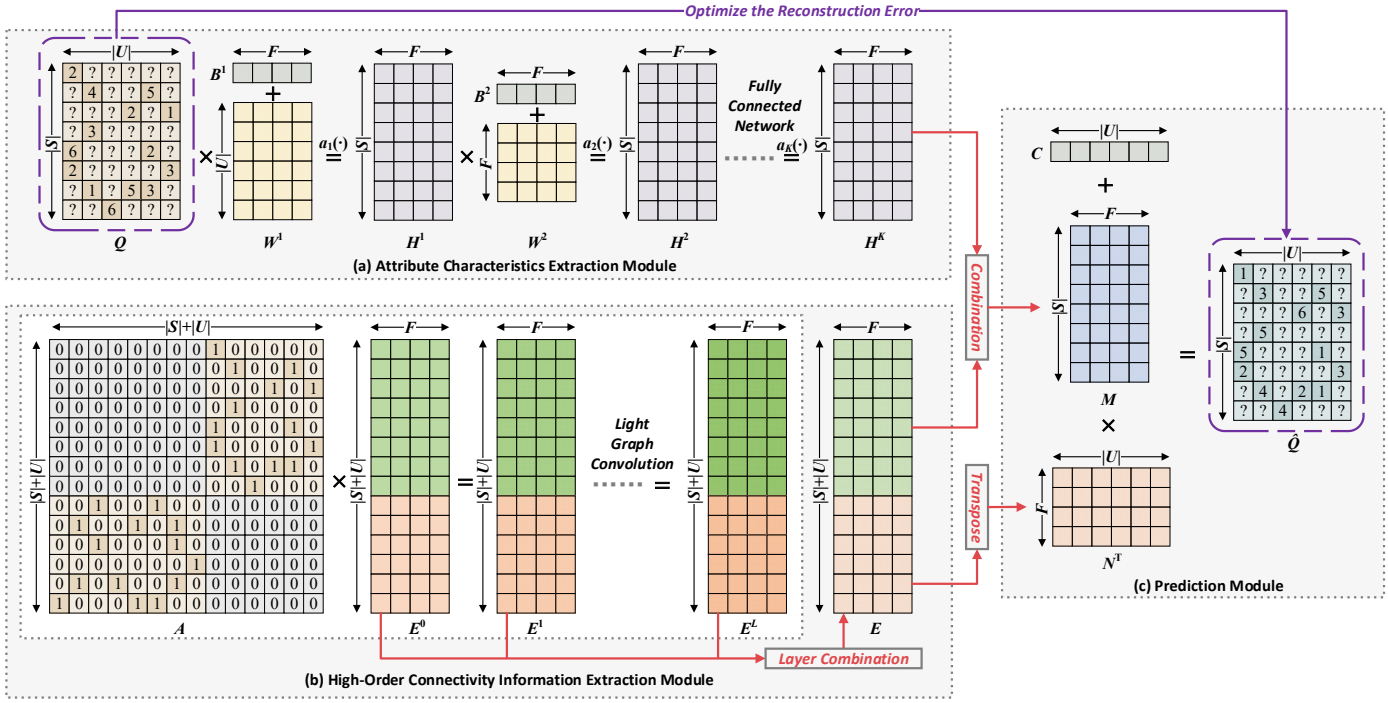


Fig. 2. The structure and flowchart of the proposed TLGCN model.

$$h_{s,f}^{k} = a_k\left(\sum_{d=1}^{F} h_{s,d}^{k-1} w_{d,f}^{k} + b_f^{k}\right), \tag{2}$$

where $h_{s,f}^{k}$, $h_{s,d}^{k-1}$, $w_{d,f}^{k}$, and b_f^{k} 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, and we find that uniformly adopting sigmoid for all layers plays the best effect.

B. High-Order Connectivity Information Extraction Module

Recently, Graph Convolution Networks (GCNs) become popular owing to their high accuracy and scalability in capturing high-order connectivity information, which work by iteratively performing message passing to aggregate multi-hop neighborhood information [13]. To explicitly exploit the high-order connectivity information from user-service interactions can acquire stronger collaborative QoS signal, its complete message passing layer with self-connection is defined as:

$$E^{l+1} = \sigma\left(\left(D^{-0.5}AD^{-0.5} + I\right)E^{l}W^{l}\right). \tag{3}$$

Note that A is a $(|U|+|S|)\times(|U|+|S|)$ adjacency matrix:

$$A = \begin{pmatrix} 0 & Q_{adj} \\ Q_{adj}^{T} & 0 \end{pmatrix}. \tag{4}$$

where Q_{adj} is Q's adjacency matrix. D is A's diagonal degree matrix. E^l and W^l denote the latent feature matrix and feature transformation weight matrix of the l-th layer. $\sigma(\cdot)$ is a nonlinear activation function.

Despite GCNs' wide success in various graph learning fields, several recent studies [13-16] argue that by appropriately simplifying GCNs, the performance on CF tasks can be further boosted. Inspired by this, we design the high-order connectivity information extraction module as depicted in Fig. 2(b) and adopt the following message passing strategy in each layer:

$$E^{l+1} = D^{-0.5}AD^{-0.5}E^{l}. \tag{5}$$

Note that compared with the complete form, (5) removes: 1) the feature transformation, i.e., W^l; 2) the nonlinear activation, i.e., $\sigma(\cdot)$; and 3) the self-connection, i.e., I. For each specified user or service, the $(l+1)$-th layer latent feature can be obtained by only aggregating the normalized sum of its neighborhood features, which greatly lowers the training difficulty of E^0 thereby achieving more efficient and accurate representations. Specifically, in (5), the propagation rules for user u and service s are defined as:

$$\begin{aligned} e_u^{l+1} &= \sum_{s\in N(u)} \frac{1}{\sqrt{|N(u)|}\sqrt{|N(s)|}} e_s^{l}, \\ e_s^{l+1} &= \sum_{u\in N(s)} \frac{1}{\sqrt{|N(s)|}\sqrt{|N(u)|}} e_u^{l}. \end{aligned} \tag{6}$$

where $1/\sqrt{|N(u)|}\sqrt{|N(s)|}$ is the symmetrically normalized form, in which $N(u)$ and $N(s)$ denote the directly-connected neighbor node sets of u and s. It can avoid the feature scale increasing with multiple graph convolution operations and assign different importance to each neighbor for higher accuracy and diversity. After propagating L layers, we obtain the final representations by adopting the layer combination, i.e., calculating the weighted sum of the latent features propagated at each layer as:

$$\begin{aligned} E &= \alpha_0 E^0 + \alpha_1 E^1 + \alpha_2 E^2 + \cdots + \alpha_L E^L \\ &= \alpha_0 E^0 + \alpha_1 \tilde{A} E^0 + \alpha_2 \tilde{A}^2 E^0 + \cdots + \alpha_L \tilde{A}^L E^0, \end{aligned} \tag{7}$$

where $\tilde{A} = D^{-0.5}AD^{-0.5}$ is A's symmetrically normalized matrix; $\alpha_l \geq 0$ denotes the importance of l-th layer features to constitute the final representations, we set it uniformly as $1/(L+1)$ here. Note that (7) plays a similar effect to self-connection, and it can alleviate the over-smoothing problem. In the whole propagation process, E^0 is the only parameter matrix to optimize, which is

easy to train thereby gaining high efficiency and accuracy.

C. Prediction Module

In the prediction module, the services' final latent features are combined as:

$$M = \omega E_{1\sim|S|} + (1-\omega) H^K, \tag{8}$$

where M is the final service feature matrix, and ω denotes the importance of $E_{1\sim|S|}$ and H^K in constituting the final latent features. Then we extract N from E as the resultant user latent feature matrix, as shown in Fig. 2(c). Based on M and N, TLGCN estimates the unknown entries in Q as:

$$\hat{q}_{s,u} = \sum_{f=1}^{F} m_{s,f} n_{u,f} + c_u, \tag{9}$$

where $\hat{q}_{s,u}$ is the estimation for each $q_{s,u}$, and $c_u \in C$ is the bias for u to enlarge the solution space. And then as discussed in Section II, we utilize the commonly-adopted Euclidean distance-based objective function to optimize the variables as:

$$\varepsilon = \frac{1}{2} \sum_{q_{s,u} \in \Lambda} \left(q_{s,u} - \hat{q}_{s,u} \right)^2 + \lambda \left\| \Theta \right\|^2 \tag{10}$$

where Λ denotes the known entry set and λ controls the L_2 regularization strength. Note that (10) also follows the data density-oriented modeling principle to optimize the variables only on the known entries in Q.

D. CUDA-Parallelized Sparse Matrix Computation

TLGCN adopts the data-density oriented principle in its input and output parts, i.e., only based on the known entry set without any data filling, for avoiding time and storage consumption. However, existing mainstream deep learning frameworks are immature in sparse computing and they cannot absolutely support TLGCN's numerous sparse matrix operations in GPU. Hence, following the prior studies [17, 18], an efficient CUDA-parallelized sparse matrix computation module is implemented for TLGCN's GPU acceleration.

Two key schedulers, i.e., SM and warp, are adopted in CUDA programming to achieve two-level sparse matrix computation parallelism. Compressed Sparse Row (CSR) format is utilized to re-represent a sparse user-service QoS matrix, which is constituted by three arrays: a) *ptr* array storing 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 invocation records in QoS data. Note that given a target 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 some sparse matrix operators (SMOs), e.g., the outer product one and row product one. Note that we follow the previous studies [17, 18] to set the block size to 32×32 in this paper for better performance.

IV. Experiments and Results

A. General Settings

Datasets. Two real QoS data[1] collected by the WS-Dream system are applied in our experiments, which are the largest publicly-available QoS datasets and widely adopted in prior studies [15-28]. The details of designed testing cases based on them are shown in Table I.

TABLE I. Properties of Testing Cases

Dataset	$\|\Lambda\|$	$\|U\|$	$\|S\|$	No.	Train:Validation:Test
Response Time (D1)	1,873,838	339	5,825	D1.1	1%:4%:95%
				D1.2	2%:8%:90%
				D1.3	3%:12%:85%
				D1.4	4%:16%:80%
Throughput (D2)	1,831,253	339	5,825	D2.1	1%:4%:95%
				D2.2	2%:8%:90%
				D2.3	3%:12%:85%
				D2.4	4%:16%:80%

Evaluation Metrics. The Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are adopted to evaluate each model's estimation accuracy:

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

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

where $|\Psi|$ denotes the number of entries in the testing set Ψ.

Each model's GPU time cost to converge is recorded to evaluate the computational efficiency. We implement all the experiments in Python 3.7, except that the compressed sparse matrix parallel program is written with CUDA C and compiled with CUDA 11.1. All empirical tests are deployed on a server with a 2.4-GHz Intel Xeon 4214R CPU, four NVIDIA RTX 3090 GPUs, and 128-GB RAM. And our codes are available[2].

Comparison Models. Eight state-of-the-art QoS estimators are involved in our comparison, whose details are as below.

M1. Mult-VAE [11]. A variational AutoEncoder introducing a multinominal distribution into data.
M2. NeuMF [10]. A widely adopted CF baseline which combines multilayered perception and inner product.
M3. MetaMF [12]. A federated meta matrix factorization model to generate private embeddings of users.
M4. LR-GCCF [14]. A linear GCN-based QoS estimator which removes nonlinearities to enhance the performance.
M5. LightGCN [13]. A light GCN-based model, which removes feature transformation and nonlinear activation.
M6. DGCN-HN [15]. A deep GCN-based model, which uses hybrid normalization for flexible message aggregation.
M7. HMLET [16]. A hybrid method of linear and non-linear collaborative filtering.
M8. TLGCN. The model given in Section III.

Training Settings. The following settings are employed.

a) We initialize all the trained variables randomly with Xavier method, and optimize all involved models with Adam [13];
b) For M1-3, we adopt the model architectures suggested in their original papers, and for M4-8, i.e., all GCN-based models, the latent feature dimension is fixed to 200;
c) To achieve fair comparison, we carefully tune each model's hyper parameters to achieve the best performance;
d) On each testing case, we repeat the splitting and training process for ten times and record the final average results;
e) Each model's training process terminates if: a) the training epoch reaches a preset threshold, i.e., 1000; or b) its estimation accuracy keeps decreasing for 30 epochs.

[1]https://wsdream.github.io/dataset/wsdream_dataset1.html

[2]https://github.com/Oak-B/ICDM-2022-TLGCN

TABLE II. THE RMSE AND MAE OF TLGCN WITH DIFFERENT NUMBERS OF ATTRIBUTE CHARACTERISTICS EXTRACTION ON D1.1-2.4.

No.	*K*=1		*K*=2		*K*=3		*K*=4	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
D1.1	1.7595	0.6898	1.7240	0.6963	**1.7129**	**0.6865**	1.7321	0.6902
D1.2	1.5480	0.6471	1.5280	0.6091	**1.5230**	**0.5989**	1.5292	0.6035
D1.3	1.4847	0.6290	1.4512	0.5765	**1.4455**	**0.5600**	1.4505	0.5611
D1.4	1.4424	0.6078	1.4035	0.5455	**1.3967**	0.5336	1.4042	**0.5307**
D2.1	0.9014	0.3378	0.8728	0.3327	0.8660	**0.3280**	**0.8643**	0.3297
D2.2	0.7907	0.3033	0.7779	0.3011	**0.7715**	**0.2944**	0.7718	0.2981
D2.3	0.7074	0.2688	**0.6539**	**0.2445**	0.6894	0.2720	0.6880	0.2695
D2.4	0.6677	0.2473	**0.6047**	**0.2248**	0.6535	0.2594	0.6541	0.2587

TABLE III. THE RMSE AND MAE OF TLGCN WITH DIFFERENT NUMBERS OF LIGHT GRAPH CONVOLUTION ON D1.1-2.4.

No.	*L*=1		*L*=2		*L*=4		*L*=8	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
D1.1	1.7477	0.6939	1.7330	0.6875	1.7057	0.6836	**1.6986**	**0.6755**
D1.2	1.5344	0.6079	1.5250	0.6045	1.5197	0.5984	**1.5063**	**0.5972**
D1.3	1.4533	0.5667	1.4490	0.5674	1.4454	**0.5564**	**1.4427**	0.5657
D1.4	1.4007	0.5407	1.3965	0.5373	1.3962	**0.5312**	**1.3962**	0.5328
D2.1	0.8724	0.3312	0.8667	0.3284	0.8613	0.3267	**0.8508**	**0.3260**
D2.2	0.7807	0.2995	0.7781	0.2984	0.7668	0.2931	**0.7524**	**0.2895**
D2.3	0.7043	0.2793	0.6962	0.2753	0.6826	0.2701	**0.6665**	**0.2664**
D2.4	0.6620	0.2618	0.6565	0.2603	0.6509	0.2585	**0.6430**	**0.2572**

B. Effect of Multilayered Structure

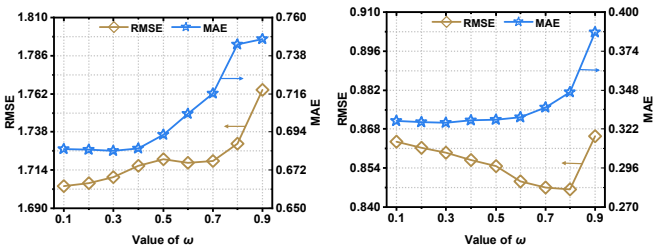
We vary the depth of two modules, i.e., we search K in {1, 2, 3, 4} and L in {1, 2, 4, 8}. The results are shown in Tables II and III. From them, we have the following findings:

a) **Increasing the depth of the attribute characteristics extraction module significantly enhances TLGCN's estimation accuracy.** For instance, as recorded in Table II, by fixing other hyper parameters on D1.3, as K varies from 1 to 3, TLGCN's RMSE decreases from 1.4847 to 1.4455, which achieves the estimation error gap (i.e., $(Error_{high}-Error_{low})/Error_{high}$) at 2.65%; MAE is similar. In most cases, TLGCN achieves the best performance as K=2 or 3, and stacking more layers leads to its overfitting.

b) **Capturing high-order connectivity information from the user-service interaction graph substantially enhances the collaborative QoS signal.** As summarized in Table III, on all eight testing cases, the estimation errors decrease greatly with the increase of L. Note that on most cases, TLGCN achieves the highest accuracy as L=8, which indicates that it is vital to explicitly exploit the collaborative signal from user-service interactions.

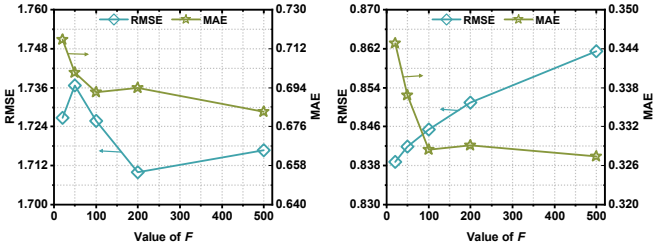
C. Analysis of Hyper Parameter Sensititvity

To investigate the impacts of ω and F, Figs. 3 and 4 depict the errors as they vary on D1.1 and D2.1. From them, we achieve the findings as below:



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

Fig. 3. Errors of TLGCN as ω varies while others being fixed.



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

Fig. 4. Errors of TLGCN as F varies while others being fixed.

a) ***ω* significantly affects TLGCN's estimation accuracy.** For instance, as depicted in Fig. 3(a), on D1.1, as ω varies from 0.1 to 0.9 while others being fixed, the RMSE increases from 1.7041 to 1.7646 with the RMSE increment of 1.94%. Considering the MAE, it increases from 0.6842 to 0.7476 with the MAE increment of 8.48%. Similar situations are encountered on other testing cases.

b) ***F* also greatly influences TLGCN's estimation accuracy, and it performs differently in RMSE and MAE.** For instance, as illustrated in Fig. 4(b), on D2.1, as F increases from 20 to 500, the RMSE increases from 0.8388 to 0.8615 with the RMSE increment of 2.63%. However, F's effects vary greatly in MAE, the MAE decreases from 0.3448 to 0.3274 with the estimation error gap of 5.05%. We attribute this phenomenon to that the testing cases are still not big enough, and we will test TLGCN's performance on larger-scale datasets to deeply understand it in our future work.

D. Comparison with State-of-the-Art Models

Tables IV and V present the RMSE, MAE, and GPU time costs in RMSE and MAE of M1-8 on all testing cases. Their last rows summarize the Friedman statistical results of all tested models [19]. From them, we have the following findings:

TABLE IV. THE RMSE, MAE, AND FRIEDMAN TEST RESULTS OF TLGCN ON ALL TESTING CASES.

No.	RMSE								MAE							
	M1	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**
D1.1	1.9396	1.9718	1.9229	1.8621	1.8147	2.0220	1.8685	**1.6990**	0.9399	0.7936	0.8768	0.8899	0.7771	0.7590	0.7571	**0.6872**
D1.2	1.9141	1.7398	1.8414	1.8443	1.6361	1.8406	1.6499	**1.5210**	0.9108	0.7025	0.8814	0.8782	0.7162	0.7452	0.7043	**0.6041**
D1.3	1.8973	1.6148	1.7820	1.7536	1.5848	1.7278	1.5771	**1.4503**	0.8925	0.6413	0.8635	0.8521	0.6845	0.7223	0.6651	**0.5632**
D1.4	1.8839	1.5318	1.7446	1.5449	1.5514	1.6525	1.5169	**1.4016**	0.8562	0.6100	0.8264	0.7498	0.6542	0.7019	0.6288	**0.5380**
D2.1	1.1404	1.0480	1.0827	1.0971	0.9775	1.1756	1.0578	**0.8501**	0.5136	0.3934	0.5472	0.5549	0.3875	0.4380	0.3983	**0.3278**
D2.2	1.1266	0.7710	0.9995	1.0310	0.7769	1.0756	0.8061	**0.7422**	0.4838	0.3035	0.5267	0.5354	0.3147	0.3942	0.3238	**0.2864**
D2.3	1.1171	0.6727	0.9334	0.8805	0.6674	0.9548	0.6825	**0.6125**	0.4767	**0.2302**	0.4812	0.4889	0.2647	0.3660	0.2840	0.2303
D2.4	1.0992	0.6696	0.8760	0.7467	0.5982	0.8566	0.6411	**0.5718**	0.4775	0.2160	0.4283	0.3889	0.2352	0.3351	0.2674	**0.2121**
F-Rank*	7.63	3.75	6.00	5.25	2.63	6.50	3.25	**1.00**	7.25	2.38	6.88	6.88	3.38	4.75	3.38	**1.13**

**A lower F-rank value denotes a higher estimation accuracy for missing QoS data.*

TABLE V. THE TIME COSTS TO CONVERGE IN RMSE AND MAE (SEC.), AND FRIEDMAN TEST RESULTS OF TLGCN ON ALL TESTING CASES.

No.	RMSE								MAE							
	M1	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M1**	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**
D1.1	20	2586	324	267	207	263	990	**5**	25	2552	80	236	247	157	1205	**17**
D1.2	20	1623	206	187	274	370	1377	**6**	33	1544	82	170	418	444	1791	**13**
D1.3	19	2872	172	687	261	541	1755	**7**	40	2851	137	300	410	604	2037	**30**
D1.4	**20**	2391	128	1028	232	645	1915	21	41	2470	153	850	438	693	2112	**25**
D2.1	48	3178	306	314	252	229	1208	**7**	55	3102	98	259	309	180	991	**15**
D2.2	42	2627	243	821	355	346	1196	**5**	52	2632	192	276	445	329	974	**8**
D2.3	45	2691	175	796	452	549	2281	**38**	50	2655	189	655	442	534	1856	**33**
D2.4	45	1841	127	771	495	740	2421	**30**	45	1971	155	642	449	712	2044	**28**
F-Rank*	1.88	7.88	3.75	5.50	4.13	4.63	7.13	**1.13**	2.00	7.75	3.00	4.88	5.00	5.13	7.25	**1.00**

**A lower F-rank value denotes a higher efficiency when addressing a QoS matrix.*

a) **M8, i.e., TLGCN significantly outperforms its peers in estimation accuracy for missing QoS data.** For instance, as recorded in Table IV, on D2.1, M8 achieves the RMSE at 0.8501, which is about 25.46% lower than M1's 1.1404, 18.88% lower than M2's 1.0480, 21.48% lower than M3's 1.0827, 22.51% lower than M4's 1.0971, 13.03% lower than M5's 0.9775, 27.69% lower than M6's 1.1756, and 19.64% lower than M7's 1.0578. Similar situations in RMSE and MAE are encountered on other testing cases. More persuasively, M8 also has the lowest F-rank value, e.g., in RMSE, M8's F-rank value is 1.00, which is lower than 7.63, 3.75, 6.00, 5.25, 2.63, 6.50, and 3.25 achieved by M1-7.

b) **TLGCN has substantially higher computational efficiency than that of its peers when addressing a QoS matrix.** As recorded in Table V, since it adopts the data density-oriented principle and light graph convolution, M8's time cost in RMSE/MAE is far less than its peers. For instance, on D1.1, in RMSE, M8 takes 5 seconds to converge, which is 25.00% of 20 seconds by M1, 0.19% of 2586 seconds by M2, 1.54% of 324 seconds by M3, 1.87% of 267 seconds by M4, 2.42% of 207 seconds by M5, 1.90% of 263 seconds by M6, and 0.51% of 990 seconds by M7; MAE is similar.

V. CONCLUSION

Aiming at performing efficient and accurate representation learning to QoS data, this paper proposes a TLGCN model. It constructs a multilayered fully-connected network to extract services' attribute characteristics; uses light graph convolution to learn high-order connectivity information from the user-service interactions; and incorporates the data density-oriented modeling principle to achieve high computational efficiency. Experimental results on two real QoS data demonstrate that TLGCN significantly outperforms the state-of-the-arts. And we plan to boost the strengths of its graph convolution on larger-scale QoS data in our future work.

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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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, 2019.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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 Retrieval*, Xi'an, China, 2020, pp. 981-990.

[13] 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 Retrieval*, Xi'an, China, 2020, pp. 639-648.

[14] 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 Conference on Artificial Intelligence*, New York, USA, 2020.

[15] 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.

[16] 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.

[17] 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.

[18] 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 Architecture*, Vienna, Austria, 2018, pp. 724-736.

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