IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 26 July 2023, accepted 12 August 2023, date of publication 17 August 2023, date of current version 23 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3305945

THEORY

Matrix Completion of Adaptive Jumping Graph Neural Networks for Recommendation Systems

XIAODONG ZHU, (Member, IEEE), JUNYU FU, AND CHEN CHEN
Business School, University of Shanghai for Science and Technology, Shanghai 200093, China

Corresponding author: Junyu Fu (fujunyu0319@163.com)

This work was supported by the National Natural Science Foundation of China under Grant 71871144.

ABSTRACT Using graph neural networks to model recommendation scenarios can effectively capture high-order relationship features between objects, thereby helping the model better handle recommendation problems. However, the over-smoothing phenomenon poses a performance constraint for recommendation algorithms based on graph convolutional node aggregation. In realistic recommendation scenarios that involve social relationships, the imbalance of node degrees can deepen the impact of over-smoothing on recommendation accuracy. To address these issues, we propose an adaptive matrix completion algorithm for collaborative filtering recommendation, which is based on the aggregation rules of relational graph convolutional networks, and introduces jumping knowledge connection for adaptive selection of user-item feature aggregation results of deep graph convolutional networks. And in order to overcome the limitations of existing interlayer aggregation mechanisms, we design a self-attention-based aggregation mechanism to integrate the output of each layer and enhance the generalization ability of the model. In addition, we introduce normalization in the process of data transmission between layers to ensure the distinguishability between nodes. Finally, we conduct experiments on three real recommendation datasets to compare the algorithm's performance and perform ablation analysis. Our model achieves RMSEs of 0.9058, 0.8346 and 0.7176 on the three datasets respectively. The results show that the recommendation performance of our model achieves a leading level when compared with current state-of-the-art algorithms and verifies the influence of node degree distribution on the recommendation process.

INDEX TERMS Feature aggregation, jumping knowledge connections, matrix completion, recommendation systems, relational graph convolutional network.

I. INTRODUCTION

In the era of big data, the massive amount of data contains rich value and great potential, but also presents the serious problem of information overload. As an important tool to address this problem, recommendation algorithms have become a hot topic in academia. Currently, recommendation algorithms have been successfully applied in many fields, including e-commerce, information retrieval, social networks, location-based services, news delivery and other fields [1].

Traditional recommendation algorithms are primarily classified into collaborative filtering recommendation, content-based recommendation and hybrid recommendation [2]. Among them, collaborative filtering recommendation [3] predicts users' interests based on the interaction between users and items, making it one of the most popular algorithms due to its good performance. Additionally, with the advancement of deep learning technology, some collaborative filtering recommendation algorithms based on neural network models have been proposed [4]. These algorithms attempt to solve the recommendation problem from two aspects.

On the one hand, the algorithm models the recommendation for a certain user as a sequential prediction problem, where the interaction between the user and different items is considered as a sequence in a chronological order [5]. The reason behind is that the items chosen by the user are often related in the dimension of time, and the choices made by the user within a short time frame can reflect certain preferences, and the learning of such preferences can help to provide the user with proper item recommendations. This

The associate editor coordinating the review of this manuscript and approving it for publication was Taous Meriem Laleg-Kirati.

type of approach draws on the learning ability of recurrent neural networks for sequential data [6], and learns users' preferences through their interaction history. Besides, there are some works that integrate the user's interaction using the self-attention mechanism [7] to learn the correlation of historical items and thus select other items in the system that are suitable for the user. Most of the recommendation methods for sequential prediction are item-based recommendations, which calculate the similarity between items based on the user's behavior such as ratings for the items. It provides users with suitable recommendations based on the similarity of items and their historical behaviors.

On the other hand, with the powerful feature extraction ability and the nonlinear mapping ability of neural networks, some methods find suitable items for users using hidden features. These methods use neural networks to perform nonlinear fitting on the feature vectors of users and items to accomplish matrix completion task. Commonly used methods include convolutional neural networks as well as auto-encoding [8]. To address the sparsity of the recommendation data, some methods enhance the fitting ability of neural networks by tensor decomposition [51] or data augmentation [52] to make predictions for missing values. Meanwhile, researchers have noticed the impact of social networks or recommendation networks on user preferences, and hence graph neural networks (GNNs) are applied to recommendation problems to capture high-order information in user-item networks [9]. GNNs have demonstrated excellent topology awareness in areas such as Nonfactoid question answering [53], traffic flow prediction [54], and edge computing [55]. The community information contained in the network topology expresses the mutual influence between nodes [56]. The connectedness between communities, the activity of nodes, and the social similarity of nodes are used to measure the interaction between nodes [57], which reflects the user's preferred choice of items in a recommendation scenario. The user-item interaction can be modeled as a heterogeneous bipartite graph with different nodes representing users and items and edges representing ratings. GNNs can well represent graphs and effectively capture local as well as global graph features using message aggregation. In the field of recommendation, this modeling approach captures the interaction between different objects, which is helpful in capturing the high-order features of nodes in the network, and can effectively alleviate the cold start problem of recommendation systems [10].

However, existing recommendation methods based on GNN have not fully considered the imbalance in the node degrees in a network. The degrees of nodes in real networks vary significantly. In Figure 1, we visualize three real-world datasets: MovieLens 100K, 1M, and 10M. The scatter plot depicts the distribution of node degrees, where the nodes closer to the center have higher degrees, i.e., they have more connected edges. In ML-100K, there are more nodes in the center, while in ML-1M and ML-10M, a large number of nodes are distributed at the boundary, indicating that most nodes in the datasets are low-degree nodes. In the histogram, the degrees of all nodes are sorted, and nodes in the blue part in MovieLens 100K, 1M, and 10M account for 10%-30% of the total number of nodes, but occupy 80% of the connections. The above phenomenon indicates that there exists imbalance in node degrees in the real-world recommendation problems, which may negatively affect the performance of GNN-based recommendation algorithms.

Graph convolutional networks (GCNs) along with other GNNs rely mainly on neighborhood aggregation, which generally suffers from over-smoothing when dealing with graphs that vary dramatically in local neighborhoods. This over-smoothing leads to degradation of the model when increasing the number of GCN layers. The essence of GCN over-smoothing is that as the number of GCN layers increases, the node features in the network tend to homogenize during multiple convolutional processing, i.e., the nodes become increasingly indistinguishable [11]. In the recommendation problem, we regard the different degrees of nodes' preferences for the same class of items as the difference between them. Then reflected in the recommendation accuracy, when nodes tend to be homogeneous, it is difficult for us to predict whether some nodes have similar preferences for the current item, thus reducing the accuracy of the recommendation. In the recommendation scenario, this imbalance in the degrees of user and item nodes can greatly exacerbate the over-smoothing of the GCN and therefore introduce performance losses for the GCN-based recommendation model, as shown by our experiments.

Therefore, in this paper, we aim to mitigate over-smoothing in GCNs and address the uneven distribution of node degrees in real-world data. Firstly, when adding multiple convolutional layers, we introduce the jumping knowledge connection, and node features are obtained by adaptive selection of the output of each layer, and the outputs between the convolutional layers are normalized using PairNorm [12] to reduce the similarity of nodes generated in the aggregation process. Secondly, we design an inter-layer aggregation based on the self-attention module to adaptively select the appropriate number of convolutional layers based on the features of the datasets. Finally, we conduct experiments on three real-world datasets to verify the impact of node degree imbalance on the recommendation performance and evaluate the effectiveness of our proposed mode. Sensitivity of the parameters and the effectiveness of each module in our model are tested by parameter analysis and ablation experiments, respectively.

Some related solutions to the GNN over-smoothing problem have been thought through in terms of GNN message passing. But when modelling with GNNs in recommendation scenarios, there are few methods to personalise and improve these solutions for the real-world problems mentioned. This paper first demonstrates the prevalence of node imbalance in recommendation scenarios through visualisation of realistic datasets. Then we adapt various over-smoothing coping

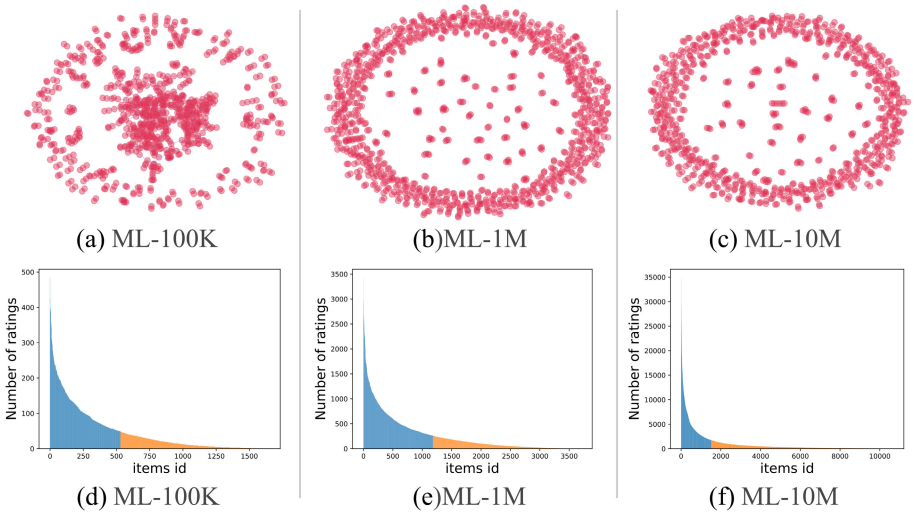

FIGURE 1. Node degree distribution for different datasets. The node that is closer to the center of the scatter plot has higher degrees, and the histogram shows degrees of different items, with nodes in the blue part occupying 80% of the connections.

methods to this feature of the recommendation dataset. The main contributions of this paper include:

1. We combine over-smoothing coping methods such as jumping connections and PairNorm in the GNN-based matrix complementation task, and design a new attention-based aggregation mechanism in the inter-layer aggregation process of jumping connections to improve the model performance while obtaining an acceptable computational overhead. To our knowledge, such personalised solutions to the problem of node-degree imbalance in recommendation tasks are relatively rare.

2. In the process of modeling recommendation prediction based on social relationships, the different node degree distributions reflected in the model require us to adopt individualised over-smoothing responses to this data characteristic to ensure the validity and generality of the model. In terms of understanding user interaction behaviour, the different node degree distributions remind us to consider the role of node influence on interaction outcomes and further understand user preferences. Our research goes some way to establish a relationship between user interaction behaviour and recommendation model design.

3. And we demonstrate through rich and well-developed experiments that personalisation improvements to these methods in recommendation models are effective, and that the effectiveness of our models is universal across multi-scale datasets.

The rest of this paper is organized as follows. Section II reviews related works. Section III presents the main theoretical approach of this paper. Section IV describes the modeling process of the matrix completion problem. A matrix completion algorithm based on jumping connections with attention inter-layer aggregation is proposed in Section V. Section VI presents the results of comparison and ablation experiments, and analyzes them. Section VII concludes our work.

II. RELATED WORKS

In this section, we will give a brief review of existing graph neural networks and analyse the advantages as well as the disadvantages of different graph neural networks. On this basis, we present a review of recent research on recommendation algorithms and matrix completion models that incorporate graph learning methods. Specifically, these works either employ sequentially connected convolutional layers or utilize mechanisms to address issues such as over-smoothing and cold-start problems.

GNN is a graph representation learning framework that follows a neighborhood aggregation scheme. First the graph convolution network GCN performs convolutional operations via local fast first-order approximations on the graph [26]. Variants such as GAT [28], which is based on an attention mechanism, and GraphSAGE [22], an inductive framework based on neighborhood sampling and aggregation of features, have emerged from this foundation. In addition to this, a number of targeted solutions have been proposed for different graph neural networks in order to address some of the problems that exist in GNNs. To address the problem that GCNs cannot learn to distinguish certain simple graph structures, GIN [58] designs a message aggregation architecture that is as powerful as the WL test by conducting a theoretical analysis of the representational power of GNNs. Wang et al. [59] show that nonlinearity is unnecessary for spectral GNNs to reach high expressiveness, and for the first time build a bridge between the expressivity analyses of spectral GNNs and spatial GNNs. The ACMP model [60] simulates attractive and repulsive forces in interacting particle

systems via a neural ODE solver, substantially increasing the number of GNN layers without over-smoothing effects, and theoretically proving a strict positive lower bound on the Dirichlet energy. In addition, He et al. [61] address the problems of over-squashing and poor long-range dependencies in message aggregation GNNs by fusing the ViT/MLP-Mixer architecture in computer vision and making the complexity of the model linear to the number of nodes and edges.

The recommendation model in question utilizes graph convolutional networks (GCN) to capture high-order information about nodes. In addition, the model aggregates embeddings of entities outside the knowledge base for observed entities, effectively addressing the cold-start problem.

Previous research has also leveraged GCN for matrix completion in recommendation systems. For example, GCMC [13] developed a micro information transferable graph auto-encoder framework that represents interaction data as a user-item bipartite graph. However, this framework sequentially connects graph convolution layers without considering the effect of node degrees distribution on over-smoothing.

NGCF [14], on the other hand, uses GCN to mine association rules between users and items in the recommendation process. NGCF's three-layer structure incorporates high-order connectivity to extract useful information in each layer, following the GCN approach.

Hu et al. [15] proposed the GNewsRec algorithm, which addresses both the data sparsity problem and changing user preferences. Specifically, the model analyzes user interests using an attention-based LSTM model and utilizes graph convolutional networks to learn user and news embeddings.

IGMC [16] is an inductive recommendation matrix completion model that trains graph neural networks based on first-order neighborhood subgraphs. The model maps these subgraphs to the link predictions of unseen nodes, resulting in accurate and efficient recommendations.

Finally, graph learning is also used to deal with multi-modal entity attributes in recommendation scenarios. Guan et al. [17] unify user, item and attribute entities and relationships by constructing heterogeneous graphs in the context of personalized fashion compatibility modeling. Besides, pre-defined metapaths are adopted to capture the higher-order relationships between entities, and a contrastive regularization is introduced to strengthen embedded learning. Achieved sota performance. BiHGH [18] integrates multiple entities into a heterogeneous graph and applies a bi-directional graph convolution scheme to model multiple entities in the outfits recommendation scenario, alleviates the computational cost of big graph convolution. And designs a dual similarity preserving regularization to enhance the hash code learning of users and outfits. Chen et al. [62] use the effective conductance on the user-item bipartite graph as the hardness score, propose model-agnostic hard negative subsampling methods, provide a sustainable and consistent data subsampling solution to real-world recommendation systems. Park et al. [63] devise a novel criteria preference-aware light graph convolution (CPA-LGC) method, apply GNN to the multi-criteria recommender system.

To mitigate the over-smoothing problem in graph learning, Xu et al. [11] consider graph structure and propose a JK network that can adaptively select network depth to improve performance. In addition, Zhang et al. [19] propose a stacked and reconstructed graph convolutional networks to learn node representations in recommendation scenarios. This approach addresses the limitations of previous methods, which relied on a fixed number of sequentially connected graph convolution modules. Instead, Zhang et al employ recursive multi-block graph codecs to stack multiple layers of graph convolution, allowing the model to learn low-dimensional user and item potentials as inputs. This limits the spatial complexity of the model and improves performance, particularly in cold-start environments.

In summary, some recommendation algorithms utilize stacked GNNs to learn user and item embeddings from various perspectives, including link prediction, association rules, and cold-start mitigation. However, they often pay less attention to issues such as GNN over-smoothing and the impact of node degree distribution on the performance of recommendation models. In addition, some of the graph neural networks mentioned in this section address some of the key problems in GNNs including over-smoothing, but there are few methods to apply these solutions to recommendation models. And the high complexity of these methods does not help us to design an efficient and universal recommendation model. Therefore, we firstly explore the reason why the model is susceptible to over-smoothing in the recommendation process from the data level, i.e. the imbalance of the social relationship node degree distribution. Secondly, we personalise methods such as jumping connections and design a matrix completion model that adaptively selects the number of GNN layers, effectively improving the recommendation accuracy of the model. Equally important, our model achieves performance improvements while its computational overhead is effectively controlled.

III. METHODS

A. GRAPH CONVOLUTION FEATURE AGGREGATION

Graph neural networks(GNNs) are deep learning models capable of capturing the topology of data by representing nodes and edges in a graph. There are two ways to represent graphs in GNNs: node attribute representation and structural representation of the graph itself [20]. GNNs utilize node neighborhood aggregation to extract high-dimensional information from node neighborhoods and generate dense vector embeddings [21]. These node representations can be used to perform various downstream tasks, such as node classification and link prediction in recommendation systems [22].

Suppose $G(V, E)$ denotes the input graph with node $v_i \in V$, edge $(v_i, v_j) \in E$, and size N. $X \in R^{N\times F}$ is called all features associated with the graph nodes, and the adjacency matrix is defined as $A \in R^{N\times N}$, associating each edge (v_i, v_j) with its element A_{ij}. The node degree is denoted by

$d = \{d_1 \ldots d_N\}$, where d_i refers to the sum of the edge weights connected to node i. Define D as the degree matrix with diagonal elements of d.

Let $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ be the symmetrically normalized Laplacian of the graph, the spectral decomposition of L yields $L = \sum_{n=1}^{N} \lambda_n u_n u_n{}^T$, and we use the graph filter $h(\lambda)$ to derive the frequency response for each eigenvalue λ_n. The filtered graph signal is:

$$\begin{aligned}\overline{X} &= \sum_{n=1}^{N} h(\lambda_n) u_n u_n{}^T X \\ &= U diag[h(\lambda_1), \ldots, h(\lambda_n)] U^T X \end{aligned} \tag{1}$$

The conventional graph filter is approximated by a polynomial of order K [23]. The filter and the filtered graph signal are:

$$h(\lambda) = \sum_{k=0}^{K} \alpha_k \lambda^k. \tag{2}$$

$$\begin{aligned}\overline{X} &= (\alpha_0 I + \alpha_1 L + \alpha_2 L^2 + \ldots + \alpha_K L^K) X \\ &= \sum_{k=0}^{K} \alpha_k L^k X \end{aligned} \tag{3}$$

The K-order polynomial approximation avoids the direct spectral decomposition of the Laplace matrix to derive the frequency. Also, the Chebyshev polynomial expansion $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ has been used in signal processing [24], [25], where a local approximation of the polynomial filter can be performed fast, and the filtered graph signal is obtained as:

$$\overline{X} = \sigma(\sum_{k=0}^{K-1} T_k(\widetilde{L}) X W_k). \tag{4}$$

where σ is a nonlinear activation function, $\widetilde{L} = 2L/\lambda_{\max} - I$, and W_k are trainable parameters, and the complexity of (4) is $O(|E|)$, which is linearly related to the number of edges [26]. Based on the local Chebyshev-approximation, Kipf & Welling obtain the graph convolutional network GCN [24] for the semi-supervised classification task by setting $K = 1$ and $W = W_0 = -W_1$:

$$\overline{X} = \sigma(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}} X W). \tag{5}$$

where the degree matrix and adjacency matrix are regularized such that $\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}} = I + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, the modified degree matrix and adjacency matrix contain self-loops, and W are trainable parameters, the complexity of the GCN filtering operation is $O(|E|FC)$ [26], where C is the number of input channels and F is the number of feature graphs. According to the spectral domain of GCN in (5), the aggregation rule of embedding $h_v{}^{(l)}$ in the l-th convolution is as follows:

$$h_v{}^{(l)} = ReLU(W_l \sum_{u \in N(v)} \frac{1}{\sqrt{\deg(v)\deg(u)}} h_u{}^{(l-1)}). \tag{6}$$

where $\deg(v)$ is the degree of node v in G and $N(v)$ is the set of nodes in the neighborhood of v.

B. OVER-SMOOTHING OF GRAPH NEURAL NETWORKS

Among deep learning approaches for node representation, neighborhood aggregation is a popular messaging scheme. These models iteratively update the hidden feature vector of each node based on its neighborhood in the aggregation graph to generate the final node representation [22], [26] [27], [28]. However, a problem with the common neighborhood aggregation model is that when multiple graph neural network layers are stacked, the model performance may not improve, and may even decrease significantly. This is due to the over-smoothing phenomenon, where the hidden feature vectors of the nodes at the output become indistinguishable after deepening the number of network layers.

Existing research on the causes of over-smoothing mainly considers two aspects. The first aspect is the imbalance of graph topology [11], i.e., the uneven distribution of graph node degrees, which is common in the recommendation scenario, since the system is more likely to be affected by a small number of high degree nodes. The other aspect is the overmixing of information and noise during aggregation. The study of [29] indicates that during node aggregation, the interaction among nodes of the same class can bring useful information, while that among nodes of different classes may lead to homogenization of node representations, which in turn affects the model performance. Recommendation modeling is usually performed on cross-class interactions, i.e., predicting the rating between users and items.

In order to alleviate the problem of over-smoothing in graph neural networks, it is necessary to consider the characteristics of neighborhood aggregation and the resulting limitations. Among existing approaches, JK-Nets [11] select the aggregation results for each layer, use jumping connections to learn representations for different subgraph structures in different sequences, and finally merge the selected aggregation results for each layer to generate the final node embedding. This method can selectively use the information from neighborhoods at different locations to adapt to drastic local changes in the graph. The Adaptive Graph Convolutional Neural Network(AGCN) [30] utilizes a new spectral graph convolutional network to perform adaptive convolution on the original input data from different graph structures. AGCN creates graph-specific Laplacian matrices for different samples instead of sharing a spectral kernel. This operation allows the model to aggregate neighborhood features according to different topologies. Zhou et al. [31] introduced two over-smoothing measures by considering the community structure of node clusters: group distance ratio and instance information gain, which quantify the over-smoothing of global topology and local node features, respectively. Based on these measures, they proposed a differentiable group normalization that can effectively alleviate over-smoothing by preventing the nodes of different groups from being represented too similarly.

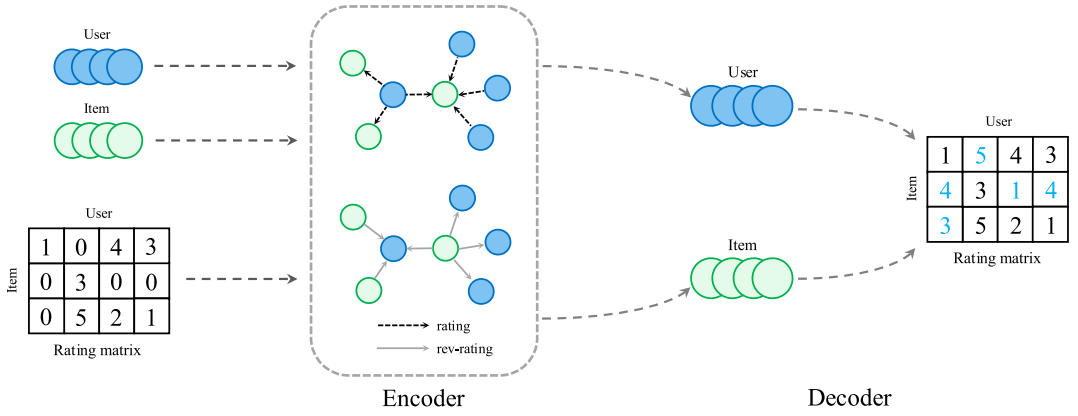

FIGURE 2. Forward propagation process of matrix complement for recommendation system.

The above methods propose the solution to over-smoothing arising from the drastic change of graph topology or the imbalance of node degree. In the recommendation scenario, such imbalance is prevalent in user or item nodes, and hence it is necessary to consider the impact of over-smoothing generated by such imbalance on the model performance.

IV. PROBLEM FORMULATION

The matrix completion of the recommendation system treats the rating between user and item as a rating matrix $M \in R^{|U|\times|V|}$, where U and V are the sets of user and item, respectively. The element M_{ij} in the matrix represents the rating of user i on item j. If user i has no rating on item j or its rating is not observed, then M_{ij} is represented by 0. The task of matrix completion for recommendation systems is to predict the true rating of element 0 of the rating matrix based on the user and item feature vectors, as well as the existing rating matrix [13].

Inspired by pre-trained models, GNN-based matrix completion methods can divide link prediction into upstream and downstream tasks. In the upstream task, graph node embeddings are learned through weighted graph convolutional networks as encoders. In the downstream task, multilayer nonlinear neural networks are used as decoders to predict ratings based on the node embeddings of users and items [32]. Figure 2 illustrates the flow of upstream and downstream tasks in GNN-based matrix completion for recommendation systems.

V. OUR MODEL

In order to realistically consider various rating densities and mitigate over-smoothing in the graph convolution matrix completion process, we propose the JK-DMC model, whose forward propagation process is shown in Figure 3. Firstly, a graph is constructed based on the rating matrix and node features to be aggregated. The jumping knowledge connection is introduced to alleviate over-smoothing when increasing the number of aggregation layers. Also, to maintain the stability of node features in each aggregation layer, the output

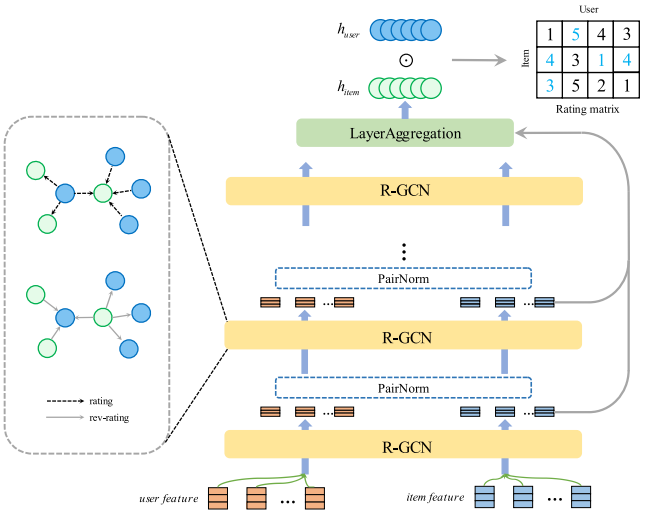

FIGURE 3. The complete structure of JK-DMC.

of each layer is PairNorm normalized. Finally, we use multiple layer aggregation mechanisms to select the output of each layer of jumping knowledge connection and propose an inter-layer output fusion strategy based on a self-attention mechanism. The aggregated node embeddings are then used to reconstruct the ratings between users and items through decoders.

A. EMBEDDING LAYER

We preprocess the original user features in the embedding layer to obtain the feature vector for each user and item. Suppose the original feature sequence of $item_i$ is $I^{in}{}_i = \{q_1, q_2, \ldots, q_m\}$. First, a one-hot vector of length d_{in} is used to represent the original feature $q_{in} \in R^{d_{in}}$ of the item. At this point, q_{in} is high-dimensional and sparse, and we put q_{in} into an embedding layer to reduce the feature dimension and obtain the embedding vector $q_{out} = Embedding(q_{in}) \in R^{d_{out}}$. The original features are embedded and put into a fully connected layer for further dimensional organization, and then processed by sigmoid activation function and BatchNormalization at the output of the fully connected layer.

After processing each original feature of item j in the abovementioned way, they are concatenated to obtain a complete vector I_i^{out}. For the original user features, we follow the same process. It is worth noting that overfitting, which is prevalent in graph tasks, may make the model generalize poorly to the testing data and aggravate the cold-start problem [33]. Therefore, in this paper, a method of randomly masking the input node features based on a certain ratio is used to alleviate the cold-start problem during the training process [19], [34], and the masked node feature matrix $F_{drop} \in R^{N\times C}$ is:

$$F_{drop} = F \times N_{MASK}. \tag{7}$$

where $F = \{I_1, \ldots, I_N\} \in R^{N\times C}$ is the input node feature matrix and $N_{MASK} \in R^{N\times C}$ is the mask matrix used to zero out the feature vectors in F randomly at a certain ratio. The mask matrix is randomly initialized and randomly masked for the input features during training, but we retain all node features during inference.

B. HETEROGENEOUS CONSTRUCTION

The recommendation scenario contains two roles, user and item, which are considered as two types of nodes. Since there are different levels of ratings for user to item, the connections between the two nodes are considered as multiple classes of edges. Modelling the user-item interaction results in a graph structure where both node and edge have multiple categories, and are therefore a heterogeneous graph.

In the upstream task, JK-DMC generates new node embeddings by adding multiple GNN node aggregation layers. The graph structure composed of recommendation data contains both user and item nodes, and there are different kinds of connections between nodes according to the ratings.

We use relational graph convolutional network(R-GCN) [36] as the node aggregation rule in the rating prediction process. Compared with normal GCNs, GCNs focus only on message transfer between nodes and cannot be adaptive to different edge types. R-GCN is specifically designed to handle multi-relational data in realistic knowledge bases, and R-GCN introduces relation-specific transformations on top of GCN. For each type of relationship present in the system, i.e., the edge type in the graph, R-GCN uses a separate weight matrix for processing. Thus during the neighborhood aggregation of a node, node features from the same connection type are treated separately, which allows the node to perceive the corresponding relationship type. In the recommendation problem, the interaction between user and item is more than just information transfer. Different ratings of item by user represent different types of interactions and reflect the interest preferences of user on item. Therefore, R-GCN is suitable for learning diverse user-item ratings in recommendation systems, which helps users' adaptive selection of items.

Let the edge from user to item be of type rating-r and the edge from item to user be of type rev-rating-r, with r representing the rating value. In this paper, we construct a directed weighted graph $G(I, U, R)$ for feature aggregation of nodes using the user-item rating relationship [13], [35], where I and U represent item and user nodes, respectively, and R is the edge, i.e., the rating relationship. We use an independent weight matrix W_r for each type of rating r. Node features from the same connection type will be treated separately from others during the neighborhood aggregation of a node. Let $h_{user_i}^{(l)}$ be the node embedding obtained by $user_i$ aggregation at layer l, then the aggregation result $h_{user_i}^{(l+1)} \in R^{N\times d}$ at layer $l+1$ is:

$$h_{user_i}^{(l+1)} = \sigma\Big(\sum_{r\in R}\sum_{item_j\in N_{user_i}^r}\frac{1}{c_{ij}}W_r^{(l)}h_{item_j}^{(l)} + W^{(l)}h_{user_i}^{(l)}\Big). \tag{8}$$

where $N_{user_i}^r$ is the set of indexes in the neighborhood of node $user_i$ under $r \in R$ and $c_{ij} = |N_{user_i}^r|$ is the normalization constant.

C. ADAPTIVE JUMPING GRAPH NEURAL NETWORK NODE AGGREGATION

In recommendation systems, for matrix completion models that sequentially connect multiple GNN layers for node aggregation, increasing the number of GNN layers can lead to a degradation of model performance, i.e., over-smoothing. This phenomenon is caused by two factors. First, similar to deep CNN models, a large number of parameters brought by too many layers can lead to severe overfitting. However, unlike residual networks [37] that can effectively improve the generalization performance of deep CNN models, this approach does not play a significant role in deep GNN models. Second, unlike CNN models, which process data in Euclidean space, GNNs are affected by the graph topology during node aggregation. Specifically, the radius of influence of nodes is different for nodes that have high degree and are near the center of the graph compared to nodes that have low degree or are at the edge of the graph. For nodes that have high degree and are near the center of the graph, they have a strong ability to influence other nodes, and when the number of aggregation layers is high, it can lead to too much similarity between other nodes and such nodes. On the other hand, for nodes that have low degree or are at the edge of the graph, using fewer aggregation layers will lead to insufficient information aggregation. The effect of graph topology on GNN matrix completion is more obvious in the recommendation scenario due to the diffusion effect of information propagation.

To alleviate over-smoothing, we introduce jumping knowledge connection [11] in the node aggregation process to select the most suitable aggregation depth for different nodes, which is adaptively adjusted according to the effective neighborhood size of the nodes. Two concepts are defined in JK-net: influence distribution and random walks. Where influence distribution is used to measure the effect of node x on node y, indicating how much the change in the original input features of x affects the representation of y in the final layer of the network. Suppose $h_x^{(0)}$ is the input feature of node x and $h_x^{(k)}$ is the hidden feature of x learned by the model at the

k th layer. Then the influence score $I(x, y)$ of node x on node y is the sum of the absolute values of the entries of the Jacobian matrix $[\frac{\partial h_x^{(k)}}{\partial h_y^{(0)}}]$, and the influence distribution of x to y is $I_x(y) = I(x, y)/\sum_z I(x, z)$. Also, random walks denote starting at a node v_0; if at step t we are at a node v_t, we move with equal probability to any neighbor of v_t (including v_t). Then the t-step random walks distribution P_t for v_0 is $Prob(v_t = i)$. As mentioned earlier, nodes have different degrees, which gives them different influence scores, and the number of model layers required for their features to radiate to a certain range is also different. The literature [11] proves theorem: assume that all paths in the graph of the model are activated with the same probability of success ρ. The influence distribution I_x in expectation of any node $x \in V$ is then equivalent to the distribution of k-step random walks on $\widetilde{G}$ starting from node x. The role of the inter-layer aggregation function in JK-net is to determine the importance of a node's sub-graph features at different ranges after looking at the features learned on all layers, rather than optimizing and fixing the same weights for all nodes. Taking the Max-pooling aggregation function as an example, on the basis of the above theorem literature [11] proves that under a k-layer JK-Net with inter-layer max pooling, the influence score $I(x, y)$ for any $x, y \in V$ is equivalent in expectation to $0, \ldots, k$-step random walks starting from x, a mixture of k-step random walks distributions whose coefficients depend on the value of the layer feature $h_x^{(l)}$. This gives the model the ability to adaptively select the layer depth based on the features $h_x^{(l)}$ of different nodes at each layer, i.e. learning the corresponding weight coefficients for each layer feature $h_x^{(l)}$ of the node. This avoids a situation where the model is overly disturbed by high-influence nodes due to the use of the same weights resulting in final node features that are indistinguishable from each other, i.e. over-smoothing. Figure 4 shows the forward propagation process of this connection.

In the process of computing node embedding, first k GNN layers are connected sequentially, and the message passing in (8) is performed in each layer, and the output of the previous layer is the input of the next layer. Instead of taking the output of the last layer as the final representation of the nodes, jumping knowledge connection performs a merging process on the set $\mathbb{C} = \{h^{(1)}, h^{(2)}, \ldots, h^{(k)}\}$ of outputs of all layers to obtain the final node embedding matrix. The merging process selects the aggregation results for each layer, and selects the aggregation results with the appropriate depth for the nodes according to their radius of influence, which allows the model to be more adapted to the topology of the recommendation datasets.

For the output set $\mathbb{C} = \{h^{(1)}, h^{(2)}, \ldots, h^{(k)}\}$, the output results of the k GNN layers can be selected using a variety of aggregation functions to compute the final node embedding. The selectable aggregation functions include Concatenation, Max-pooling, and Average-pooling [11]. Among them, Concatenation is the simplest aggregation method which

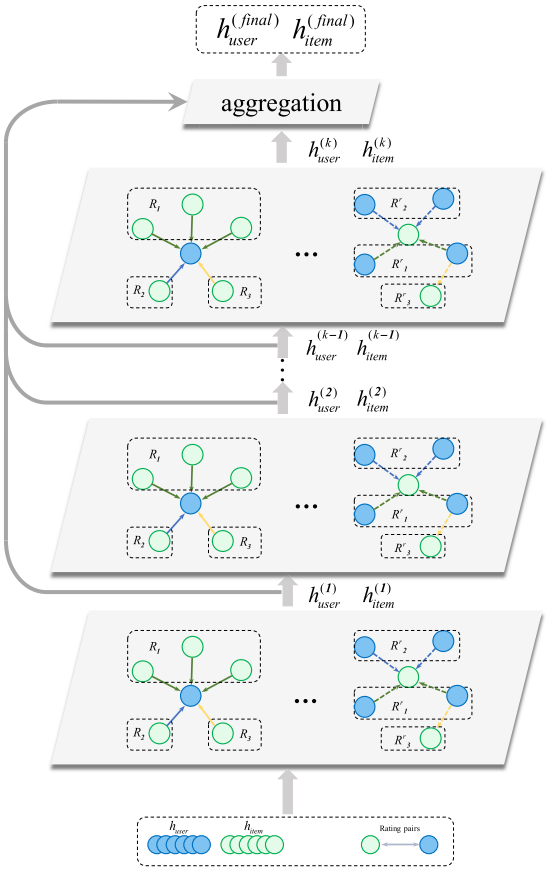

FIGURE 4. Introducing jumping knowledge connection in the node aggregation process. The node representation is calculated from the output of the last layer and the output of each intermediate layer.

directly concatenates the output set and then performs a linear transformation on the concatenated results. Concatenation is expected to be used for small graphs and graphs with regular structure, because such graphs do not require strong adaptive capabilities. Max-pooling and Average-pooling are two types of pooling operations. Max-pooling applies to the feature vector coordinate dimension, where the largest value in each feature coordinate is selected for each layer. Average-pooling is to average all elements in $\mathbb{C}$ and calculate the average value of each layer's output, including the hidden layer. Additional parameters are not required for Max-pooling and Average-pooling, and the calculation process is as follows:

$$Max(\mathbb{C}) = \max(\{h^{(1)}, h^{(2)}, \ldots, h^{(k)}\}). \tag{9}$$

$$Average(\mathbb{C}) = \frac{1}{k}\sum_{i=1}^{k} h^{(i)}. \tag{10}$$

In addition, we have also designed an aggregation function based on multi-headed self-attention [38]: Attention-pooling. The multi-headed self-attention module can better learn sequence information and has stronger adaptability when dealing with complex sequences. Specifically, the selection

Algorithm 1 JK-DMC Node Aggregation

1: Input: The heterogeneous graph $G(I, U, R)$;
2: Output: Node representations $h^{(final)} = \{h^{(final)}_{user}, h^{(final)}_{item}\}$ by aggregation.
3: Initial input feature $h^{(0)} = \{h^{(0)}_{user}, h^{(0)}_{item}\}$ of the user and the item;
4: Matrix of rating A^R, Number of convolution layers k;
5: LayerAggregation={Concatenation, Max-pooling, Average-pooling, Attention-pooling}.
6: for i = 1,...,k do:
7: $h^{(i-1)} = PairNorm(h^{(i-1)})$
8: $h^{(i)} = AGGREGATE^i(h^{(i-1)})$
9: end for
10: Node representation of every layer $\mathbb{C} = \{h^{(1)}, h^{(2)}, \ldots, h^{(k)}\}$.
11: Selection of results $h^{(final)} = LayerAggregation(\mathbb{C})$.
12: return $h^{(final)}$.

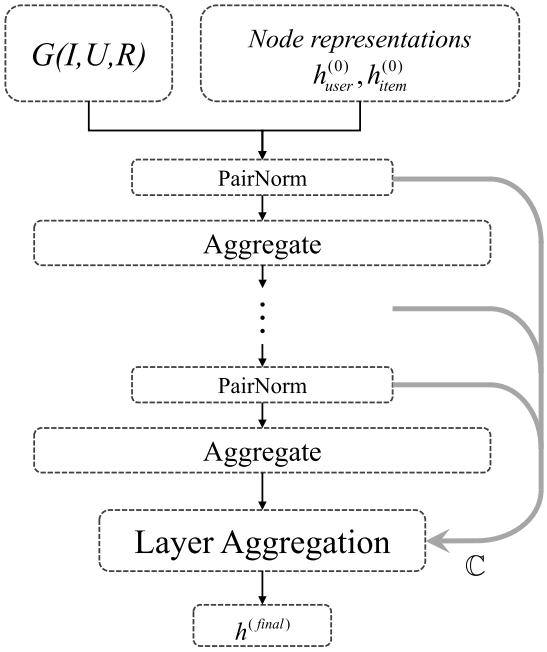

FIGURE 5. Node aggregation process based on JK-net and PairNorm.

process for the output set $\mathbb{C}$ can be formulated as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V, \tag{11}$$

$$h_i = Attention(\mathbb{C}W_i^Q, \mathbb{C}W_i^K, \mathbb{C}W_i^V), \tag{12}$$

$$Attention - pooling(\mathbb{C}) = [h_1; h_2; \ldots; h_H]W^O. \tag{13}$$

where H is the number of self-attention module heads and $d_k = \frac{d}{H}$, $W_i^Q, W_i^K, W_i^V \in R^{d\times d_k}$, $W^O \in R^{Hd_k\times d}$ is the parameter matrices. The complete procedure of JK-DMC node aggregation is shown in Algorithm 1 and Figure 5.

D. INTER-LAYER NORMALIZATION

During the process of node aggregation using multilayer graph convolutional networks, we apply normalization to the output from each GCN layer before passing the results to the next layer of the network. We use PairNorm [12] for this purpose. PairNorm is a normalization layer that improves the robustness of the algorithm. Its main idea is to maintain a constant total pairwise feature distance between the output results of each network layer. This ensures that nodes from different clusters are less likely to fuse features with each other, thereby preventing nodes from different clusters from becoming indistinguishable.

PairNorm normalization processes the output of each layer of node aggregation, which is also the input of the next layer of node aggregation, such that the total pairwise squared distance (TPSD) of the node embedding after each processing is the same. Let the input and output of PairNorm be X and $\tilde{X}$, respectively. We want to ensure that $TPSD(\tilde{X}) = C$, where C is a constant and more specifically, a hyperparameter that can be adjusted according to the dataset. $TPSD(X)$ can be written as:

$$\begin{aligned} TPSD(X) &= \sum_{i,j\in n} ||x_i - x_j||_2^2 \\ &= 2n^2(\frac{1}{n}\sum_{i=1}^{n}||x_i||_2^2 - ||\frac{1}{n}\sum_{i=1}^{n}x_i||_2^2). \end{aligned} \tag{14}$$

where n is the number of nodes. The normalization process for X can be divided into two steps: centering and rescale:

$$x_i^c = x_i - \frac{1}{n}\sum_{i=1}^{n}x_i. \tag{15}$$

$$\tilde{x}_i = s\cdot\frac{x_i^c}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}||x_i^c||_2^2}} = s\sqrt{n}\cdot\frac{x_i^c}{\sqrt{||X^c||_F^2}}. \tag{16}$$

First (15) is centering on x_i to get the embedded vector x_i^c after translation, and the term $||\frac{1}{n}\sum_{i=1}^{n}x_i^c||_2^2$ in (14) will be 0 after processing, thus, $TPSD(X^c)$ is the squared Frobenius norm of X^C, which is $O(nd)$. The output $\tilde{x}_i$ of PairNorm can be obtained by (16).

Rescale allows $TPSD(\tilde{X})$ to be changed to a constant with respect to n and s, where s is the hyperparameter that can be used to determine C:

$$\begin{aligned} TPSD(\tilde{X}) &= 2n\sum_i ||s\cdot\frac{x_i^c}{\sqrt{\frac{1}{n}\sum_i||x_i^c||_2^2}}||_2^2 \\ &= 2n\frac{s^2}{\frac{1}{n}\sum_i||x_i^c||_2^2}\sum_i||x_i^c||_2^2 = 2n^2s^2 \end{aligned} \tag{17}$$

E. RATINGS FORECAST

In the upstream task, we obtain the node embeddings h_{user_i} and h_{item_j} of users and items by aggregating graph nodes containing jumping structures and PairNorm. Next, we will

predict users' preferences for different items based on node embedding vectors in the downstream task by constructing a decoder capable of reconstructing node links. Using a decoder structure similar to the GCMC [13] model, we let the set of possible ratings between nodes be $R = \{1, 2, \ldots, |R|\}$ and the predicted preference rating of $user_i$ for $item_j$ be $\hat{r}_{ij}$, which is calculated as:

$$h_{user_i} = \sigma(W_u h_{user_i}), \quad h_{item_j} = \sigma(W_i h_{item_j}). \tag{18}$$

$$\begin{aligned} p(\hat{r}_{ij} = r) &= sofmax(h_{user_i}^T Q_r h_{item_j}) \\ &= \frac{e^{h_{user_i}^T Q_r h_{item_j}}}{\sum_{s \in R} e^{h_{user_i}^T Q_s h_{item_j}}} \end{aligned} \tag{19}$$

$$\hat{r}_{ij} = \sum_{r \in R} r \cdot p(\hat{r}_{ij} = r). \tag{20}$$

where $W_u \in R^{d \times d_{user}}$, $W_i \in R^{d \times d_{item}}$ are two trainable projection matrices, projecting h_{user_i} and h_{item_j} to the same length d, which are then processed by the activation function $\sigma(\cdot)$ respectively. Q_r is the parameter matrix corresponding to the rating r. The output after multiplying with h_{user_i} and h_{item_j} will be fed into the softmax function to produce a probability $p(\hat{r}_{ij} = r)$ that $\hat{r}_{ij}$ is equal to the rating $r \in R$ and finally $\hat{r}_{ij}$ is obtained by weighting the sum of the possible ratings with their corresponding probabilities.

F. MODEL TRAINING

The ratings in the training and testing data are discrete integer values, and the loss is calculated during training using a cross entropy loss function and back propagation. Let the true rating between $user_i$ and $item_j$ be r_{ij}, and the predicted rating be $\hat{r}_{ij}$, then the loss between the predicted and true values is:

$$L = -\frac{1}{|\mathbb{Q}|} \sum_{[i,j] \in \mathbb{Q}} \sum_{r=1}^{|R|} I[r = r_{ij}] \log(p(r = \hat{r}_{ij})). \tag{21}$$

where $\mathbb{Q}$ is the set of rating pairs observed in the training set and the function $I[a = b] = \begin{cases} 1, & a = b, \\ 0, & a \neq b. \end{cases}$. In the back propagation process, we use Adam algorithm to minimize the loss L of the predicted samples. In addition, to alleviate the possible overfitting during the training process, we regularize the model parameters with L_2 regularization [39] and add penalty terms to the loss function to make the learned parameter smaller. The penalty term is set as the product of the sum of squares of the weight parameter and a positive constant to obtain a new loss function $L = L + \frac{\lambda}{2|\mathbb{Q}|}||\omega||^2$, where λ is the hyperparameter to be set and ω is the weight parameter vector.

VI. EXPERIMENTS

In this section, we experimentally validate the performance of our model with different node degree distribution and analyze the effect of hyperparameters on the model performance. Our experiments are conducted under Ubuntu OS with NVIDIA GeForce RTX 2080 Ti GPU. The experimental environment is based on Python 3.7, Pytorch 1.9.0 and DGL 0.9.0. DGL provides versatile control over message passing with automatic batch processing and a highly tunable sparse matrix kernel for speed optimization.

TABLE 1. Dataset of rating prediction.

Dataset	U	I	R	C	D
ML-100K	943	1682	1,...,5	100K	6.30%
ML-1M	6040	3706	1,...,5	1M	4.50%
Douban	3000	3000	1,...,5	130K	1.50%

A. DATASET

The rating prediction task in this paper is performed on the MovieLens and Douban. The MovieLens dataset is sourced from the research project GroupLens at the University of Minnesota and is widely used in fields such as information filtering and recommendation systems. The dataset contains different numbers of rating pairs, i.e., 100K, 1M. Statistics for ML-100K, ML-1M and Douban are shown in Table 1, where U and I represent the number of users and items in both datasets, R stands for the rating categories, C denotes the number of rating pairs, and D represents the average rating density of users.

B. EXPERIMENT SETTINGS

We consider the dataset ML-100K as a small dataset and ML-1M as a large dataset. For the small dataset task, we set the node embedding dimension of the GCN hidden layer to 300, the final output dimension of node embedding to 75, and the random masking ratio of node features to 0.5. For the large dataset task, we set the node embedding dimension of GCN hidden layer to 500, the final output dimension of node embedding to 75, and the random masking ratio of node features to 0.3.

Also for all tasks, we use the ReLU function as the activation function in the experiments, Adam as the optimizer, and set the learning rate to 0.02, which gradually decays with a decay rate of 0.5 and finally stops at 0.001. The maximum threshold for gradient clipping is set to 0.9 and the early stopping patience is set to 1000. The parameters to be optimized in the model are initialized by Xavier [40] method, which aims to keep the variance of the inputs and outputs as consistent as possible and prevent all output values from converging to 0. The parameters W initialized by the Xavier method follow the uniform distribution shown in (22), where n_i and n_{i+1} denote the input dimension size and output dimension size of the i-th layer, respectively. Other important hyperparameters, such as the number of convolutional layers and the number of attention heads, will be determined by experimental comparison in Section VI-E.

$$W \sim U[-\frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}, \frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}]. \tag{22}$$

In this study, we use the root mean square error RMSE and the mean absolute error MAE as measures the effectiveness of the model. The RMSE and MAE reflect the deviation between the predicted rating $\hat{r}_i$ and the true rating r_i. The smaller the deviation is, the better the model works. The RMSE and MAE are calculated in the following way:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{r}_i - r_i)^2}. \tag{23}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{r}_i - r_i|. \tag{24}$$

In addition, in the inference stage, we add another metric Recall@10 to test the prediction effect of the model. Larger value of Recall@10 metric indicates better inference effect.

C. COMPARISON MODELS

In order to verify the validity of the model, some advanced algorithms are selected for comparison. To ensure a comprehensive comparison, different baselines are selected, and we consider both traditional models and advanced models. In addition, the choice of the baseline is also affected by the dataset and metrics used in this study. We classify the selected baseline models into three categories: traditional collaborative filtering algorithms(PMF, BiasMF, DST-HRS), neural network-based recommendation algorithms(ConvMF, Factorized EAE, NNMF, NMTR, DBC-AF, RConvMF&VConvMF, JODIE); and graph learning-based recommendation algorithms(GCMC, sRMGCNN, IGMC, STAR-GCN, GRALS, DeepCoevolve, CoPE). A detail description of all baseline models is provided below:

PMF [41]: Probabilistic Matrix Factorization (PMF) is a collaborative filtering recommendation model that models the user preference matrix as the product of two low-rank user and movie matrices, and extends it to include adaptive priors on movie and user feature vectors, which is suitable for handling large-scale, sparse and unbalanced data.

BiasMF [42]: BiasMF interprets ratings from the perspective of biases, and the observed ratings are decomposed into four components: global mean, item bias, user bias, and user-item interaction. This allows each component to interpret only those parts that are relevant to it.

DST-HRS [43]: DST-HRS extracts embeddings by capturing the semantics of textual information from the perspective of solving the sparsity problem of recommendation data. And further integrates these embeddings into probabilistic matrix factorization(PMF), thus effectively exploiting the semantics of item text information to overcome the sparsity problem.

ConvMF [44]: Convolutional Matrix Factorization (ConvMF) is a context-aware recommendation model that combines a convolutional neural network (CNN) with probabilistic matrix factorization(PMF). ConvMF captures the contextual information of a document and improves the accuracy of rating prediction.

Factorized EAE [45]: Factorized EAE is a matrix completion algorithm based on parameter sharing, which proves that Permutation Equivariant is important in limiting the expressiveness of the model and achieves good generalization performance on the matrix extrapolation task.

NNMF [46]: NNMF replaces the inner product in matrix factorization with a multilayer feedforward neural network model to achieve advanced performance by optimizing the network to obtain fixed latent features.

NMTR [47]: NMTR is a neural network model that focuses on learning user preference from multi-behavior data. NMTR learn a data-dependent interaction function for each behavior type, and correlate the model prediction of each behavior type in a cascaded way.

DBC-AF [48]: DBC-AF aims to develop a movie recommendation system by the use of density based clustering (DBC) with artificial flora (AF). The model gets rid of sparsity problem by applying the content-boosted collaborative filtering technique.

JODIE [65]: JODIE employs two recurrent neural networks to not only update user and item embeddings at every interaction, but also to model the future embedding trajectories of users/items.

RConvMF&VConvMF [49]: RConvMF and VConvMF are based on convolutional matrix factorization to extract textual and multi-level visual features from descriptive texts and posters, respectively. RConvMF integrate the recurrent structures into convolutional layers to further improve the quality of word representations. VConvMF combines both textual features and multi-level visual features by convolutional neural networks.

GCMC [13]: GCMC represents the interaction data as a user-item bipartite graph and considers matrix completion for recommendation systems from the perspective of link prediction of graphs.

sRMGCNN [50]: sRMGCNN combines a multi-graph convolutional neural network and an RNN and it has a constant number of parameters, which is independent of the matrix size.

IGMC [16]: IGMC is an inductive matrix completion method that learns local graph patterns and can generalize to new local graphs for inductive learning.

STAR-GCN [19]: STAR-GCN learns node representations in recommendation scenarios by stacking reconstructed graph convolutional network structures. The previous limitation on the number of tandem graph convolution modules is alleviated by stacking or recursive multi-block graph encoders.

GRALS [64]: GRALS relies on efficient Hessian-vector multiplication schemes, provides a scalable algorithm for matrix completion graph with structural information.

DeepCoevolve [66]: DeepCoevolve is an RNN-based deep coevolutionary network model for learning user and item features based on their interaction graphs.

CoPE [67]: CoPE uses graph neural networks based on ordinary differential equation to model information propagation and more complex evolution patterns, and uses metalearning to ensure fast adaptation to the latest interactions.

TABLE 2. Result of ML-100K.

Model	RMSE	MAE	Recall@10
BiasMF	0.9170	-	-
ConvMF	0.9450	0.7440	-
Factorized EAE	0.9200	-	-
NNMF	0.9070	-	-
NNTR	0.9110	-	-
DBC-AF	-	0.7120	-
GCMC	0.9127	0.7163	-
sRMGCNN	0.9290	-	-
IGMC	**0.9050**	-	-
DeepCoevolve	-	-	0.069
JODIE	-	-	0.074
CoPE	-	-	**0.081**
JK-DMC(Concat)	0.9135	0.7159	0.066
JK-DMC(Max-pooling)	0.9076	0.7121	0.071
JK-DMC(Average-pooling)	0.9065	0.7112	0.072
JK-DMC(Attention -pooling)	0.9058	**0.7106**	0.077

TABLE 3. Result of ML-1M.

Model	RMSE	MAE	Recall@10
PMF	0.8971	-	-
BiasMF	0.8450	-	-
DST-HRS	0.8460	-	-
ConvMF	0.8531	0.6700	-
Factorized EAE	0.8600	-	-
NNMF	0.8430	-	-
RConvMF	0.8453	-	-
VConvMF	0.8361	-	-
GCMC	0.8425	0.6603	-
IGMC	0.8570	-	-
DeepCoevolve	-	-	0.030
JODIE	-	-	0.035
CoPE	-	-	0.049
JK-DMC(Concat)	**0.8346**	**0.6534**	0.053
JK-DMC(Max-pooling)	0.8465	0.6607	0.048
JK-DMC(Average-pooling)	0.8423	0.6586	0.043
JK-DMC(Attention-pooling)	0.8375	0.6562	**0.056**

D. RESULTS AND ANALYSIS

1) RESULTS ANALYSIS

For the dataset ML-100K, the performance of JK-DMC and baseline models in u1.base(training)/u1.test(testing) is shown in Table 2.

Where the training set accounts for 80% and the test set accounts for 20% of the dataset. Firstly, for the baseline model, the results of ConvMF and GCMC are obtained in the local environment in order to make a more fair comparison with the model proposed in this paper and to reduce the impact of environmental factors on the results. Other results of the baseline models are reported from the relevant literature. Our model JK-DMC uses four different aggregation methods, where the best results are obtained in Attention-pooling. Overall, the best performing model in terms of RMSE and MAE metrics is IGMC, which is an advanced matrix completion algorithm with its predominance in RMSE. However, JK-DMC also achieves comparable results, and JK-DMC based on Attention-pooling and Average-pooling is better than all the baseline models except IGMC. In terms of the Recall@10 metric, the best performance is achieved by CoPE, and JK-DMC(Attention-pooling) achieves the second best performance among all models.

The experimental results on the dataset ML-1M are shown in Table 3. The setup and the baseline models for this task are basically the same as ML-100K. The results suggest that the best performing model is the JK-DMC(Concat) model based on Concatenation aggregation, which is better than other models in both RMSE and MAE metrics. In addition, JK-DMC(Attention-pooling) also outperforms all other baselines. In terms of Recall@10 metric, JK-DMC is also ahead of the other comparison models. Among them, the JK-DMC model based on Attention-pooling and Concatenation both achieve the leading level.

The results of the experiments on the dataset Douban are shown in Table 4. Again 80% of the data are classified as the training set and 20% of the data are classified as the test set. The four JK-DMC models are compared with STAR-GCN, sRMGCNN, GRALS, GCM, and IGMC. In the results, JK-DMC based on Attention-pooling achieves the best performance. IGMC achieves the next best level, while JK-DMC (Max-pooling) and JK-DMC (Avg-pooling) achieve an accuracy close to that of IGMC.

TABLE 4. Result of Douban.

Model	RMSE	MAE
STAR-GCN	0.7270	-
sRMGCNN	0.8010	-
GRALS	0.8330	-
GCMC	0.7340	-
IGMC	0.7210	-
JK-DMC(Concat)	0.7480	0.5414
JK-DMC(Max-pooling)	0.7263	0.5228
JK-DMC(Average-pooling)	0.7238	0.5201
JK-DMC(Attention-pooling)	**0.7176**	**0.5171**

2) DATA SIZE ANALYSIS

As mentioned in Figure 1, different datasets have different distributions of node degrees. JK-DMC wants to address the negative impact of this nodal degree imbalance on the recommendation effect. Therefore, we also consider the effect of dataset size on model effectiveness. We randomly select data with proportions [20%,40%,60%,80%,100%] in the dataset ML-100K for experiments using GCMC and JK-DMC, respectively. Among them, we choose Attention-pooling, which has the best performance in Table 2, as the aggregation function of JK-DMC. The experimental results are shown in Figure 6. The recommendation effects of both JK-DMC and GCMC are gradually improved in 20%-100% ML-100K, but JK-DMC shows a higher improvement. At 20%-60%, GCMC shows better results. However, starting from 60%, JK-DMC outperforms GCMC and keeps leading at 60%-100%. The potential reason is that JK-DMC takes more into account the over-smoothing arising from the difference in the influence of

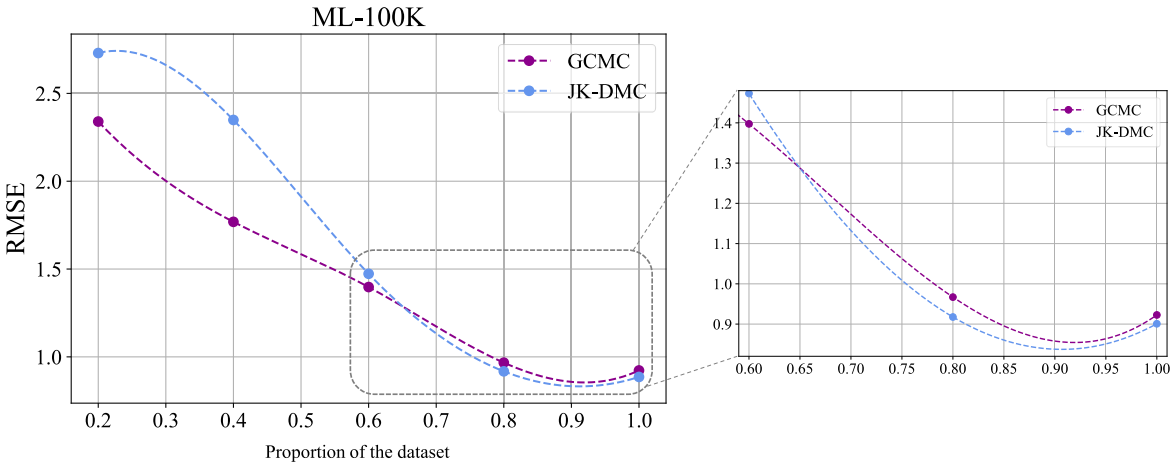

FIGURE 6. Performance of JK-DMC and GCMC at different proportions of ML-100K. The subplot on the right shows a specific comparison when the node proportion exceeds 60%.

nodes. Larger datasets have a greater variety of nodes and a higher possibility of aggregation of ordinary nodes to high nodes. This makes it easier to produce over-smoothing in large datasets, and the role of JK-DMC is more obvious in large datasets. As a result, JK-DMC achieves a suboptimal level in the relatively small ML-100K, but leads all other models in the ML-1M with larger data volume.

3) TRAINING PROCESS

In addition, the training process of JK-DMC on ML-100K and ML-1M is shown in Figure 7. Based on the performance of JK-DMC on the experimental dataset and the training process, we have the following analysis:

(1) In the results on both datasets, JK-DMC as well as the graph learning-based GCMC, sRMGCNN and IGMC outperformed the traditional collaborative filtering algorithms PMF and BiasMF, as well as the neural network-based recommendation algorithms ConvMF, Factorized EAE and NNMF, indicating the importance of utilizing high-order information of social networks in the recommendation process. On the other hand, when compared to GC-MC, sRMGCNN and IGMC, JK-DMC also achieved leading or similar results, which verifies the improvement of jumping connection structure and self-attention-based inter-layer aggregation function on recommendation matrix completion. This approach addresses the node degree imbalance prevalent in social networks by increasing the node aggregation depth through the use of jumping connection structure, enhancing the model's fitting ability. The inter-layer selection strategy based on self-attention further improves the model's adaptivity, making it more stable to noise when facing recommendation tasks of different sizes, and better able to capture the global graph structure.

(2) The comparison of the four inter-layer aggregation methods, namely Concatenation, Max-pooling, Average-pooling and Attention-pooling, reveals that models using Max-pooling and Average-pooling are more streamlined and converge faster because they do not introduce extra parameters. Therefore, JK-DMC(Max-pooling) and JK-DMC(Average-pooling) reach the optimal state faster compared to JK-DMC(Concat) and JK-DMC(Attention-pooling). In contrast, the convergence process of JK-DMC based on Concatenation and Attention-pooling is relatively slow and unstable due to the need of additional parameters, which, however, also leads to greater fitting. In ML-100K JK-DMC(Attention-pooling) achieves the best results. In the large dataset task ML-1M, JK-DMC(Concat) achieves the best and JK-DMC(Attention-pooling) achieves the second best results. Our findings suggest that the optimal results can often be obtained by using JK-DMC(Attention-pooling) in small datasets due to its greater fitting. For large datasets that consume more time and computational resources, satisfactory results can be obtained by using JK-DMC(Concat) due to its slightly lower complexity or JK-DMC(Max-pooling)/JK-DMC(Average-pooling) since they do not introduce additional parameters.

4) TRAINING TIME CONSUMPTION

Finally, we consider the time consumption by JK-DMC for model training when using different aggregation functions. We conduct experiments on ML-1M to compare with the GCMC model and the results are shown in Figure 8. The left side (red) represents the final RMSE of JK-DMC, and the right side (blue) represents the average time (sec/epoch) for JK-DMC to perform an epoch during training.

Consistent with the results in Table 2, JK-DMC (Concat) achieves the best results in ML-1M, and JK-DMC (Attention-pooling) achieves the second best results. However, JK-DMC (Attention-pooling) consumes substantially less training time than JK-DMC (Concat) while achieving results closer to those of JK-DMC (Concat). JK-DMC (Concat) consumes the most training time. It shows that the aggregation function based on multi-headed self-attention has higher training efficiency. In large-scale datasets that are

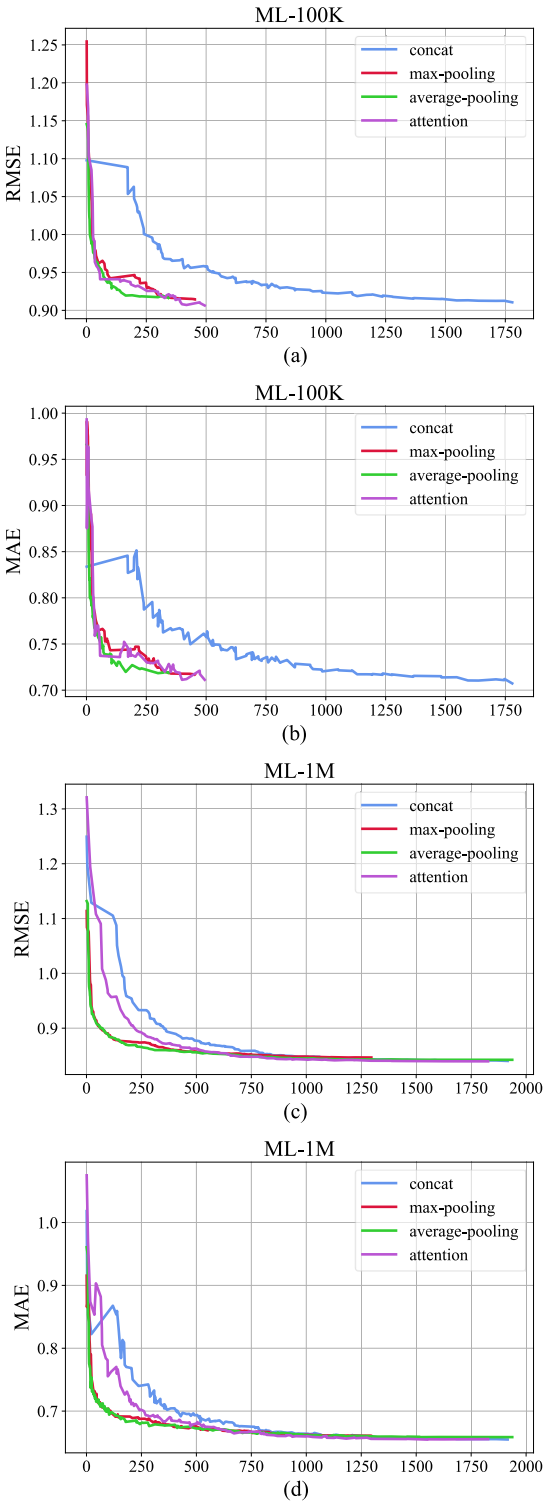

FIGURE 7. In ML-100K and ML-1M, the RMSE and MAE of the JK-DMC model on the test set converge gradually with the increase of training times. The experiments test the convergence when different aggregation methods are used, and the EarlyStopping mechanism is introduced.

prone to over-smoothing, JK-DMC (Attention-pooling) can have lower time complexity while achieving good prediction results. In addition, although JK-DMC based on Max-pooling and Average -pooling takes less time, its effectiveness lags

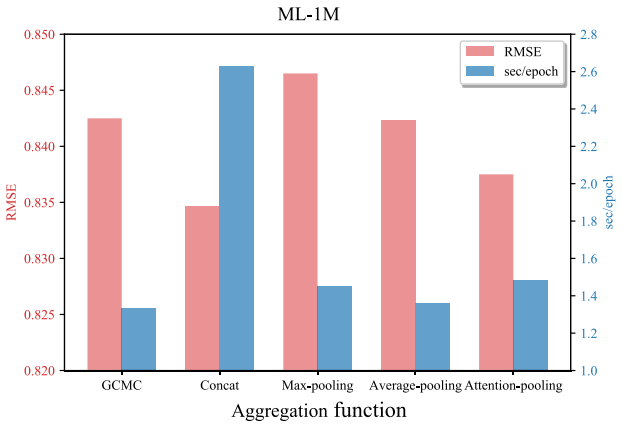

FIGURE 8. Results and training time consumption of different aggregation functions. The blue bars represent the time consumed per epoch during training and the red bars represent the RMSE of the model.

behind JK-DMC (Concat) and JK-DMC (Attention-pooling). It shows that the additional parameters introduced in the aggregation function play a role in the prediction of complex datasets. Regarding the comparison models, GCMC has a simple structure and has the lowest time consumption among all models, close to JK-DMC (Average-pooling). However, its prediction effect is also relatively backward.

E. SENSITIVITY TO HYPERPARAMETERS

In this section, we investigate the sensitivity of the hyperparameters in the model. The hyperparameters in JK-DMC model are set to different values to oberseve the change in performance on the datasets ML-100K and ML-1M.

1) NUMBER OF NODE AGGREGATION LAYERS

First, we investigate the effect of the number of hidden layers during node aggregation on the model performance. JK-DMC model uses different aggregation methods to integrate the results of each layer, and for each aggregation method, we test the prediction when the number of hidden layers is set to [2, 3, 4, 5], and the results on ML-100K and ML-1M are shown in Figure 9.

The results show that the best performance of the four aggregation methods, i.e., Concatenation, Max-pooling, Average-pooling, Attention-pooling, on ML-100K is achieved when 4, 3, 4, 2 hidden layers are used, respectively, while the best results in ML-1M dataset are achieved when 2, 3, 3, 3 layers are used. The aggregation depth is significantly higher for these models whey they achieve the best results, and most of the results are better than in the GCMC model with only one aggregation layer, indicating that over-smoothing due to the imbalance of node degree has a significant impact on the model performance, and thus it is necessary to consider such imbalance in the recommendation scenario.

2) NUMBER OF ATTENTION HEADS H

In the JK-DMC model based on Attention-pooling inter-layer aggregation, we set the number of heads in the Multi-head self-attention module to [2,4,6,8,10,12,14,16] and observe

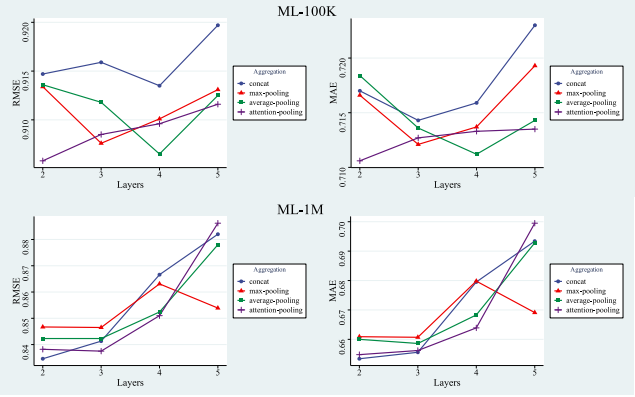

FIGURE 9. Effect of the number of GCN aggregation layers on model performance. Variation of RMSE and MAE metrics when different numbers of R-GCN aggregation layers are applied to JK-DMC based on different inter-layer aggregation methods on ML-100K and ML-1M datasets.

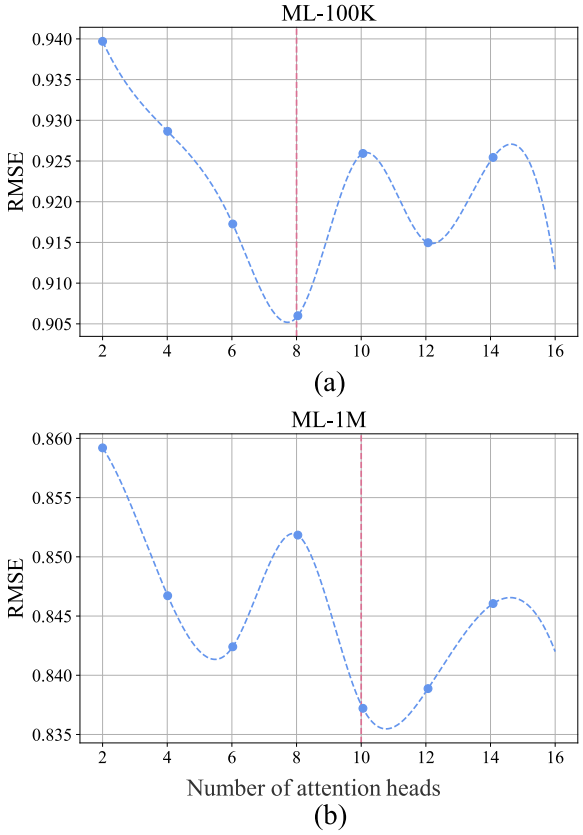

FIGURE 10. Parametric analysis of the number of Multi-headed attention heads. The optimal results are obtained when the number of attention heads of JK-DMC on ML-100K and ML-1M are set to 8 and 10, i.e. the position of the red line in the figure.

the model performance on ML-100K and ML-1M. The results are shown in Figure 10.

It can be observed that as the number of Multi-headed attention heads increases, the RMSE of the model gradually decreases in both datasets, indicating an improvement in the model's performance. This suggests that increasing the number of attention heads can enhance the model's ability to select the output of the hidden layer. However, the model's performance degrades significantly when the number of attention heads exceeds a certain threshold. Finally, the optimal results are obtained when the number of attention heads of JK-DMC on ML-100K and ML-1M are set to 8 and 10, respectively.

TABLE 5. Ablation analysis of ML-100K, ML-1M and Douban.

Models	ML-100K		ML-1M		Douban	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
JK-DMC	**0.9058**	**0.7106**	**0.8346**	**0.6534**	**0.7176**	**0.5171**
W/O PairNorm	0.9118	0.7136	0.8423	0.6586	0.7281	0.5211
W/O JK	0.9197	0.7230	0.8539	0.6691	0.7390	0.5482
W/O PairNorm&JK	0.9269	0.7293	0.8631	0.6798	0.7531	0.5590
W/O fea	0.9101	0.7137	0.8413	0.6556	0.7246	0.5313

F. ABLATION TEST

In order to investigate the contribution of each module in JK-DMC on the results, we conduct ablation experiments by removing specific components from the model and observe the results. The outcomes of these experiments are presented in Table 5. The results of JK-DMC are obtained when the optimal number of aggregation layers, aggregation method and number of Multi-headed attention heads are used, and we also consider the following settings for other models in the experiment:

W/O PairNorm: Remove the PairNorm layer between hidden layers.

W/O JK: Remove jumping knowledge Connection, hidden layers are sequentially connected, and keep PairNorm between layers.

W/O PairNorm, JK: Remove jumping knowledge connection with PairNorm. Hidden layers are sequentially connected.

W/O Fea: Remove the original feature embedding with node features randomly masked, and randomly initialize the input features.

(1) When removing jumping knowledge connection or inter-layer PairNorm normalization in JK-DMC, the performance of the model decreases significantly, indicating that these two methods can effectively alleviate the over-smoothing problem caused by graph convolution in the recommendation scenario. In recommendation scenarios, where node degrees vary dramatically, this approach can achieve outstanding performance.

(2) Simultaneous removal of jumping knowledge connection and PairNorm normalization produces a more severe degradation in the performance of JK-DMC. This reflects that keeping the total pairwise feature distances constant during inter-layer transfer can amplify the effect of jumping connections.

(3) Finally, a small decrease in model performance is also observed when randomly initializing JK-DMC input features and dropping the random masking of node embeddings, which illustrates the role of the original feature embeddings of user and item on preference selection. The random masking of node features can also alleviate the effect of overfitting and over-smoothing.

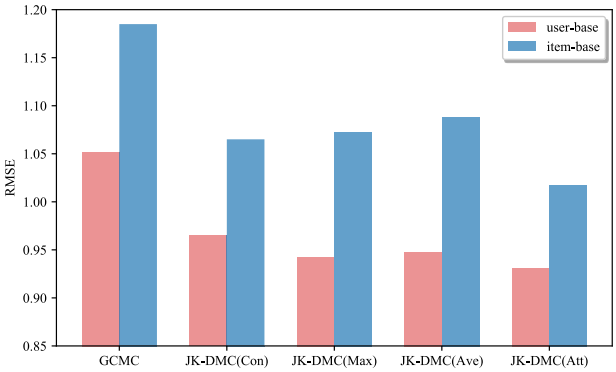

FIGURE 11. Experimental analysis of the model's cold start. Red bars represent the user-based cold start and blue bars represent the item-based cold start.

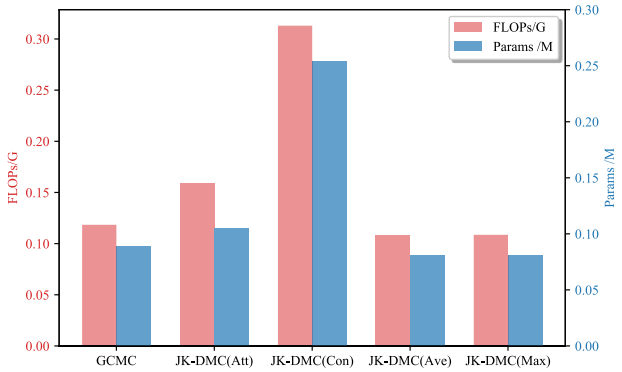

FIGURE 12. Comparison of the computational overheads of the models. Red bars represent the FLOPs of the models and blue bars represent the Params of the models.

G. COLD START EXPERIMENT

To verify the cold start capability of the proposed recommendation model and the generalization effect of this architecture in real networks, we conduct a cold start experiment using ML-100K. Using GCMC compared to JK-DMC with RMSE as the comparison metric.

The experiment divides 20% of the rating pairs in ML-100K into test set, but the users or items in the test set cannot have appeared in the train set. Therefore, in the user-based cold start experiment, we delete all the rating pairs in the training set that contain the users of the test set and obtain 47,533 training rating pairs; in the item-based cold start experiment, we delete all the rating pairs in the training set that contain the items of the test set and obtain 872 training rating pairs. The experimental results are shown in Figure 11.

In user-based cold start experiments, we simulate the scenario of a new user's first appearance in the system. The experimental results show that the scores of all four JK-DMC models have significant advantages over GCMC. Among them, JK-DMC(Attention-pooling) achieves the best result, 1.24% ahead of the second place JK-DMC(Max-pooling), and improves 12.92% over GCMC. In the item-based cold start experiments we mask certain items during the training process, and the effectiveness of several models declines as the train set size is only 872 after removing all rating pairs containing items from the test set in the train set. Among them, JK-DMC(Attention-pooling) is still the best performer, leading the second place JK-DMC(Concat) by 5.35% and improving 16.46% over GCMC. JK-DMC exhibits better cold start capability thanks to the increase of its neighbor aggregation layers, which widens the scope of message delivery. And the introduction of jumping connections alleviates the over-smoothing caused by the increase in the number of layers.

H. COMPUTATIONAL OVERHEADS

In this section, we experimentally observe the computational overheads of our model. We perform experiments on the dataset ML-100K, using JK-DMC based on four interlayer aggregation functions compared to GCMC. We use two metrics to evaluate the computational overheads of our models: FLOPs(floating point operations), which are the number of floating point operations and represent the amount of computation in the model, and FLOPs are used to measure the computational complexity of the model; Params are the total number of parameters in the model, with larger values representing more complex models. We calculate the values of FLOPs and Params for all models when performing an epoch to analyse the computational complexity of the model. The experimental results are shown in Figure 12, where red bars represent the FLOPs of the model and blue bars represent the Params of the model.

From the experimental results, JK-DMC(Avg-pooling) and JK-DMC(Max-pooling) use pooling operations for inter-layer aggregation and there are no other structural differences between them, so the FLOPs and Params values of the two models are equal. Both Avg-pooling and Max-pooling do not introduce additional parameters, which allows both models to achieve the lowest FLOPs and Params. JK-DMC (Attention-pooling) has an increase in FLOPs and Params due to the introduction of a multi-headed attention mechanism, and is slightly higher than GCMC. However, JK-DMC(Attention-pooling) achieves good performance on all three datasets, and the performance is more stable, which reflects the effect brought by Attention-pooling. The model with the highest computational complexity is JK-DMC(Concat), with much higher FLOPs and Params than the other models, and as expected, the vector concatenation results in a large number of parameters.

VII. CONCLUSION

This paper presents an overview of collaborative filtering recommendation algorithms based on graph learning. We observe the imbalance of node degree in realistic interaction networks and how it can negatively impact recommendation performance through over-smoothing during node aggregation. To address this, we propose a matrix completion algorithm that utilizes jumping connections for node feature aggregation and a layer-layer aggregation mechanism based on self-attention to adaptively select the depth of the graph

convolution network. We also use PairNorm normalization to integrate input and output data between graph convolutional layers. In experimental results on three real datasets, our model achieved leading or equivalent performance compared to several collaborative filtering recommendation models and advanced recommendation algorithms based on graph learning. Moreover, our self-attention-based approach has stronger generalization performance and is more adaptive to deep graph convolutional aggregation than other inter-layer aggregation mechanisms. Finally, we verified the importance of each model component through ablation experiments, which indicated the value of considering the degree distribution of network nodes for improving recommendation model performance. The model in this paper adaptively improves a variety of over-smoothing coping methods and applies them to the recommendation process to address the characteristics of unbalanced node degrees in the recommendation dataset. Experimental results show that the application of these techniques has significantly enhanced the model's performance. Nonetheless, as the model's complexity increases, so do its computational overheads. To mitigate this issue, we introduce a novel, attention-based aggregation mechanism in the interlayer aggregation of jumping connections. This helps to improve the model's performance while maintaining an acceptable level of computational overhead. In future work, we will consider applying parameter sharing mechanisms to reduce model complexity and further improve its generalization performance.

REFERENCES

[1] S. Wang, L. Hu, Y. Wang, L. Cao, Q. Z. Sheng, and M. Orgun, "Sequential recommender systems: Challenges, progress and prospects," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Macao, China, Aug. 2019, pp. 6332–6338.

[2] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–38, Feb. 2019.

[3] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. ICDM*, Pisa, Italy, 2008, pp. 263–272.

[4] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: A survey," *ACM Comput. Surv.*, vol. 55, no. 5, pp. 1–37, Dec. 2022.

[5] Y. Tao, C. Wang, L. Yao, W. Li, and Y. Yu, "Item trend learning for sequential recommendation system using gated graph neural network," *Neural Comput. Appl.*, vol. 35, pp. 13077–13092, Feb. 2021.

[6] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manag.*, Turin, Italy, Oct. 2018, pp. 843–852.

[7] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proc. ACM Conf. Inf. Knowl. Manag.*, Singapore, 2017, pp. 1419–1428.

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

[9] J. Chicaiza and P. Valdiviezo-Diaz, "A comprehensive survey of knowledge graph-based recommender systems: Technologies, development, and contributions," *Information*, vol. 12, no. 6, p. 232, Jun. 2021.

[10] L. A. Gonzalez Camacho and S. N. Alves-Souza, "Social network data to alleviate cold-start in recommender system: A systematic review," *Inf. Process. Manag.*, vol. 54, no. 4, pp. 529–544, Jul. 2018.

[11] K. Xu, C. Li, Y. Tian, T. Sonobe, K. I. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proc. ICML*, Stockholm, Sweden, 2018, pp. 8676–8685.

[12] L. Zhao and L. Akoglu, "PairNorm: Tackling oversmoothing in GNNs," in *Proc. Int. Conf. Learn. Represent.*, New Orleans, LA, USA, 2019, pp. 1–17.

[13] R. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, London, U.K., 2018, pp. 1–9.

[14] X. Wang, X. He, M. Wang, F. Feng, and T. S. Chua, "Neural graph collaborative filtering," in *Proc. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr.*, Paris, France, 2019, pp. 165–174.

[15] L. Hu, C. Li, C. Shi, C. Yang, and C. Shao, "Graph neural news recommendation with long-term and short-term interest modeling," *Inf. Process. Manag.*, vol. 57, no. 2, Mar. 2020, Art. no. 102142.

[16] M. Zhang and Y. Chen, "Inductive matrix completion based on graph neural networks," in *Proc. Int. Conf. Learn. Represent.*, Addis Ababa, Ethiopia, 2020, pp. 1–14.

[17] W. Guan, F. Jiao, X. Song, H. Wen, C.-H. Yeh, and X. Chang, "Personalized fashion compatibility modeling via metapath-guided heterogeneous graph learning," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2022, pp. 482–491.

[18] W. Guan, X. Song, H. Zhang, M. Liu, C.-H. Yeh, and X. Chang, "Bi-directional heterogeneous graph hashing towards efficient outfit recommendation," in *Proc. 30th ACM Int. Conf. Multimedia*, Lisboa, Portugal, Oct. 2022, pp. 268–276.

[19] J. Zhang, X. Shi, S. Zhao, and I. King, "STAR-GCN: Stacked and reconstructed graph convolutional networks for recommender systems," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Macao, China, Aug. 2019, pp. 4264–4270.

[20] P. W. Battaglia, J. B. Hamrick, and V. Bapst, "Relational inductive biases, deep learning, and graph networks," 2018, *arXiv:1806.01261*.

[21] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *Proc. Int. Conf. Mach. Learn.*, New York City, NY, USA, 2016, pp. 86–94.

[22] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 1025–1035.

[23] N. Tremblay, P. Gonçalves, and P. Borgnat, "Design of graph filters and filterbanks," 2017, *arXiv:1711.02046*.

[24] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, 2016, pp. 3844–3852.

[25] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2011.

[26] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, Toulon, France, 2017, pp. 1–14.

[27] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. ICML*, Sydney, NSW, Australia, 2017, pp. 2053–2070.

[28] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, Vancouver, BC, Canada, 2018, pp. 1–12.

[29] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *Proc. AAAI*, New York, NY, USA, 2020, pp. 3438–3445.

[30] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," in *Proc. AAAI*, New Orleans, LA, USA, 2018, pp. 3546–3553.

[31] K. Zhou, X. Huang, Y. Li, D. Zha, R. Chen, and X. Hu, "Towards deeper graph neural networks with differentiable group normalization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 4917–4928.

[32] C. Shang, Y. Tang, J. Huang, J. Bi, X. He, and B. Zhou, "End-to-end structure-aware convolutional networks for knowledge base completion," in *Proc. AAAI*, Honolulu, HI, USA, 2019, pp. 3060–3067.

[33] J. Gope and S. K. Jain, "A survey on solving cold start problem in recommender systems," in *Proc. Int. Conf. Comput., Commun. Autom. (ICCCA)*, Greater Noida, India, May 2017, pp. 133–138.

[34] Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: Towards deep graph convolutional networks on node classification," in *Proc. Adv. Neural Inf. Process. Syst.*, Addis Ababa, Ethiopia, 2020, pp. 1–18.

[35] C. Yuan, J. Li, W. Zhou, Y. Lu, X. Zhang, and S. Hu, "DyHGCN: A dynamic heterogeneous graph convolutional network to learn users' dynamic preferences for information diffusion prediction," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases* (Lecture Notes in Computer Science), 2021, pp. 347–363.

[36] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. Eur. Semantic Web Conf.* (Lecture Notes in Computer Science), Heraklion, Greece, 2018, pp. 593–607.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.
[38] A. Vaswani, N. Shazeer, N. Parmar J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 5999–6009.
[39] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learn. Represent.*, New Orleans, LA, USA, 2019, pp. 1–19.
[40] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, Sardinia, Italy, 2010, pp. 249–256.
[41] R. R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2007, pp. 1–8.
[42] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[43] Z. Khan, N. Iltaf, H. Afzal, and H. Abbas, "DST-HRS: A topic driven hybrid recommender system based on deep semantics," *Comput. Commun.*, vol. 156, pp. 183–191, Apr. 2020.
[44] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proc. 10th ACM Conf. Recommender Syst.*, Boston, MA, USA, Sep. 2016, pp. 233–240.
[45] J. Hartford, D. Graham, K. Leyton-Brown, and S. Ravanbakhsh, "Deep models of interactions across sets," in *Proc. Int. Conf. Mach. Learn.*, Stockholm, Sweden, 2018, pp. 3050–3061.
[46] G. K. Dziugaite and D. M. Roy, "Neural network matrix factorization," 2015, *arXiv:1511.06443*.
[47] C. Gao, X. He, D. Gan, X. Chen, F. Feng, Y. Li, T.-S. Chua, and D. Jin, "Neural multi-task recommendation from multi-behavior data," in *Proc. IEEE 35th Int. Conf. Data Eng. (ICDE)*, Macau, China, Apr. 2019, pp. 1554–1557.
[48] J. Parthasarathy and R. B. Kalivaradhan, "An effective content boosted collaborative filtering for movie recommendation systems using density based clustering with artificial flora optimization algorithm," *Int. J. Syst. Assurance Eng. Manag.*, pp. 1–10, Jun. 2021.
[49] Z. Wang, H. Chen, Z. Li, K. Lin, N. Jiang, and F. Xia, "VRConvMF: Visual recurrent convolutional matrix factorization for movie recommendation," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 6, no. 3, pp. 519–529, Jun. 2022.
[50] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 3698–3708.
[51] B. Cao, J. Zhao, Z. Lv, and P. Yang, "Diversified personalized recommendation optimization based on mobile data," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2133–2139, Apr. 2021.
[52] X. Liu, J. He, M. Liu, Z. Yin, L. Yin, and W. Zheng, "A scenario-generic neural machine translation data augmentation method," *Electronics*, vol. 12, no. 10, p. 2320, May 2023.
[53] Y. Deng, W. Zhang, W. Xu, Y. Shen, and W. Lam, "Nonfactoid question answering as query-focused summarization with graph-enhanced multihop inference," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 28, 2023, doi: 10.1109/TNNLS.2023.3258413.
[54] Z. Qu, X. Liu, and M. Zheng, "Temporal–spatial quantum graph convolutional neural network based on Schrödinger approach for traffic congestion prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 8, pp. 8677–8686, Aug. 2023.
[55] Y. Wang, X. Han, and S. Jin, "MAP based modeling method and performance study of a task offloading scheme with time-correlated traffic and VM repair in MEC systems," *Wireless Netw.*, vol. 29, no. 1, pp. 47–68, Jan. 2023.
[56] Q. Ni, J. Guo, W. Wu, and H. Wang, "Influence-based community partition with sandwich method for social networks," *IEEE Trans. Computat. Social Syst.*, vol. 10, no. 2, pp. 819–830, Apr. 2023.
[57] X. Zenggang, Z. Mingyang, Z. Xuemin, Z. Sanyuan, X. Fang, Z. Xiaochao, W. Yunyun, and L. Xiang, "Social similarity routing algorithm based on socially aware networks in the big data environment," *J. Signal Process. Syst.*, vol. 94, no. 11, pp. 1253–1267, Nov. 2022.
[58] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2018, *arXiv:1810.00826*.
[59] X. Wang and M. Zhang, "How powerful are spectral graph neural networks," in *Proc. Int. Conf. Mach. Learn.*, Baltimore, MD, USA, 2022, pp. 23341–23362.
[60] Y. Wang, K. Yi, X. Liu, Y. Guang Wang, and S. Jin, "ACMP: Allen-Cahn message passing for graph neural networks with particle phase transition," 2022, *arXiv:2206.05437*.
[61] X. He, B. Hooi, T. Laurent, A. Perold, Y. LeCun, and X. Bresson, "A generalization of ViT/MLP-mixer to graphs," 2022, *arXiv:2212.13350*.
[62] X. Chen, J. Sun, T. Wang, R. Guo, L.-P. Liu, and A. Zhang, "Graph-based model-agnostic data subsampling for recommendation systems," 2023, *arXiv:2305.16391*.
[63] J.-D. Park, S. Li, X. Cao, and W.-Y. Shin, "Criteria tell you more than ratings: Criteria preference-aware light graph convolution for effective multi-criteria recommendation," 2023, *arXiv:2305.18885*.
[64] N. Rao, H F. Yu, P K. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2015, pp. 2107–2115.
[65] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Anchorage, AK, USA, Jul. 2019, pp. 1269–1278.
[66] H. Dai, Y. Wang, R. Trivedi, and L. Song, "Deep coevolutionary network: Embedding user and item features for recommendation," 2016, *arXiv:1609.03675*.
[67] Y. Zhang, Y. Xiong, D. Li, C. Shan, K. Ren, and Y. Zhu, "CoPE: Modeling continuous propagation and evolution on interaction graph," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manag.*, Oct. 2021, pp. 2627–2636.

XIAODONG ZHU (Member, IEEE) received the B.S. degree from the School of Management, Anhui University, in 2002, the M.S. degree from the School of Computer Science and Technology, Anhui University, in 2005, and the Ph.D. degree from the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, in 2009. He was a Visiting Scholar with the Swinburne University of Technology, Melbourne, Australia, from 2015 to 2016. He is currently an Associate Professor with the University of Shanghai for Science and Technology. His research interests include big data management and deep learning.

JUNYU FU received the B.S. degree in information management and information system from Qingdao University, China, in 2021. He is currently pursuing the master's degree in management science and engineering with the University of Shanghai for Science and Technology, China. His research interests include deep learning, recommendation systems, and graph neural networks.

CHEN CHEN received the Ph.D. degree from Tongji University, China. She is currently an Assistant Professor with the Business School, University of Shanghai for Science and Technology, Shanghai, China. She has published several articles in refereed journals, such as *Journal of Retailing and Consumer Services*, *Asia-Pacific Journal of Operational Research*, and *INFOR: Information Systems and Operational Research*. Her research interests include social networks, platform business models, and operations management.