A Graph Convolutional Neural Network for Recommendation Based on Community Detection and Combination of Multiple Heterogeneous Graphs

Caihong Mu, Heyuan Huang, Yunfei Fang and Yi Liu*
Xidian University, Xi'an, China
caihongm@xidian.edu.cn, {huangheyuanxd, yiliuxd}@foxmail.com, fangyunfeixd@stu.xidian.edu.cn

2023 IEEE International Conference on Data Mining (ICDM) | 979-8-3503-0788-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDM58522.2023.00154

***Abstract*—Graph Convolutional Neural Networks (GCNs) have performed well in many recommendation scenarios. In spite of this, recommendation models based on GCNs still face problems such as insufficient information mining and high complexity for some existing models. To address the above problems, we propose a Graph Convolutional Neural Network for Recommendation Based on Community Detection and the Combination of Multiple Heterogeneous Graphs (GCN-CMHG). This model uses the community detection algorithm to detect the communities in the user-item interaction heterogeneous graph (UIIHG), finds the regional central nodes of communities, and then creates edges between the regional central node of each community and all other nodes in the UIIHG to construct the heterogeneous partial adjacent graph. Then, a Heterogeneous Partial Adjacent Auxiliary (HPAA) layer is designed to aggregate information on the heterogeneous partial adjacent graph. HPAA layer expands the influence of distant nodes on target nodes, enables target nodes to receive global information, and enhances the ability of GCN-CMHG to mine information. Specially, due to the low complexity of HPAA layer and the abandonment of redundant information, GCN-CMHG is easier to implement and train. Under the exact same experimental setting, GCN-CMHG's time consumption is only about 1/10 of another model based on GCN called Graph Convolutional Neural Network for Recommendation Based on the Combination of Multiple Heterogeneous Graphs (GCN-MHG). Experiments on multiple real-world datasets show that GCN-CMHG achieves better results compared with several advanced models. The implementation of our work can be found at https://github.com/GCNRSs/GCN-CMHG.**

Keywords—graph convolutional neural network, community detection, heterogeneous graph, recommendation

I. INTRODUCTION

The recommender system (RS) is a technology that can alleviate the information overload problem. In recent years, many studies have combined neural networks or Graph Convolutional Neural Networks (GCNs) with RSs [1]. Among them, recommendation models based on GCNs have achieved better recommendation results because they are more suitable for mining graph structure information in RSs.

However, as Mu et al. [2] analyze in their paper, these GCNs-based algorithms still face at least two challenges: unable to access distant information effectively and unable to effectively differentiate between useful and harmful node self-connections. Therefore, Mu et al. proposed a Graph Convolutional Neural Network based on the Combination of Multiple Heterogeneous Graphs (GCN-MHG) [2]. GCN-MHG improved recommendation effect and alleviated the information bottleneck problem by designing a Heterogeneous Fully Adjacent Auxiliary (HFAA) layer and the Partial Layers Node Self-connection (PLNS) strategy.

Despite the good results, GCN-MHG suffers from two disadvantages that limit its effectiveness:

Information redundancy: As shown in Fig. 1, in GCN-MHG, after stacking three GCN layers, node u_1 actually contains information from its second-hop homogenous nodes u_2~u_7. For distant nodes, connecting u_1~u_7 nodes to them and aggregating information by constructing heterogeneous fully adjacent graph will result in a large amount of information being repeatedly transmitted, which increases the computation but does not bring more information.

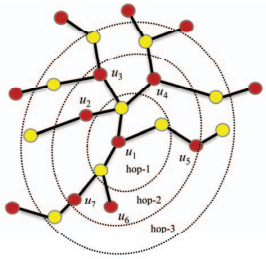

Fig. 1. Diagram of GCN-MHG transmitting information repeatedly.

High complexity: If GCN-MHG is observed from the perspective of matrix, as shown in Eq. (1), the constructed heterogeneous fully adjacent graph is actually a dense matrix containing a large number of elements as 1 (that is, all homogenous nodes are connected to each other):

$$\mathbf{A}_f = \begin{pmatrix} \mathbf{1}^{M\times M} & \mathbf{0} \\ \mathbf{0} & \mathbf{1}^{N\times N} \end{pmatrix} \tag{1}$$

where M and N represent the number of users and items respectively. $\mathbf{1}^{M\times M}$ and $\mathbf{1}^{N\times N}$ represent all-1 matrices of size $M \times M$ and $N \times N$ respectively, these all-1 matrices greatly increase the computational difficulty.

In addition, from the perspective of time complexity, when a HFAA layer in GCN-MHG is stacked, the time complexity is $O((M^2+N^2)h)$, where h represents the dimension size of the embedding. Whereas, when a GCN layer is stacked, the time complexity is just $O(|R^+|h)$, where $|R^+|$ is the number of user-item interactions, and $|R^+| << (M^2+N^2)$. Therefore, it can be found that the high time complexity of GCN-MHG will limit its application in real world.

Considering the disadvantages of the existing GCN-based recommendation algorithms, this paper proposes a Graph Convolutional Neural Network for Recommendation Based on Community Detection and the Combination of Multiple Heterogeneous Graphs (GCN-CMHG). This algorithm first uses the community detection (CD) algorithm to detect the

*Corresponding author.

979-8-3503-0788-7/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDM58522.2023.00154

communities in the user-item interaction graph and finds the regional central nodes of communities. Then, the regional central node of each community is connected with all other nodes in the user-item interaction heterogeneous graph (UIIHG) to construct the heterogeneous partial adjacent graph. A heterogeneous partial adjacent auxiliary (HPAA) layer is designed to aggregate the information on the graph. On the basis of GCN-MHG, GCN-CMHG not only reduces the negative impact of redundant information on the algorithm, but also greatly reduces the complexity of algorithm training, so that GCN-CMHG has the similar or better effect compared to GCN-MHG, while the running time is only about 1/10 of the original time.

The main contributions of this paper are as follows: (1) This paper analyzes the advantages and disadvantages of GCN-MHG in detail and discusses how to further improve its recommendation effect while reducing its time complexity. (2)This paper proposes a new graph construction method and improves the model effect without significantly increasing the complexity. (3)This paper designs experiments to compare the effects of GCN-CMHG and GCN-MHG, and to verify the performance of GCN-CMHG and its variants when stacking different GCN layers, as well as the impact of different usage methods of HPAA layer and PLNS strategy on recommendation effect.

II. Related Work

Graph Convolutional Neural Network: Topology graph, also known as graph, is a kind of structured data that defines nodes and their relationships, which is widely applied in RSs [3] and other scenarios. GCN is a deep learning technique which has been highly praised for its ability to fully mine the structural information [4]. A single GCN layer can learn information from the first-hop neighbors of nodes, while after stacking multiple layers of GCN, information from multi-hop neighboring nodes can also be transmitted. Since the data of RSs generally consists of the interactions between users and items which can be easily modeled as a graph, a lot of work in recent years began to explore how to better apply graph-based algorithms to the RSs [5,6,7].

Community Detection: The CD problem is a Non-deterministic Polynomial (NP) problem focusing on how to discover communities in the graph [8]. The CD algorithm detects community structure in the graph by dividing nodes based on their tightness or similarity in features. Early CD algorithms mostly belong to heuristic-rule-based solving strategies [9]. Later, more and more work began to seek the optimal community from the perspective of modularity [10] and modularity density [11]. Moreover, CD algorithms based on label propagation have also been widely applied [12]. The application fields of CD are very extensive [13].

III. Methodology

In this section, each component of GCN-CMHG is described in detail according to their order of occurrence, then the training method is given, and the complexity is analyzed.

A. GCN-CMHG

As shown in Fig. 2., the details of GCN-CMHG are as follows: 1) Initial embeddings are obtained through random initialization, and user-item interaction information is used to construct the UIIHG; 2) LPA [12] is used to detect communities on the constructed UIIHG, and a node with the highest degree is selected from each community as the regional central node of the community; 3) Edges are created between the regional central node of each community and all other nodes in the UIIHG to construct the heterogeneous partial adjacent graph; 4) K GCN layers are stacked on the UIIHG to aggregate information. When the first several GCN layers are stacked, the node self-connections are discarded. When the last GCN layer is stacked, the node self-connections are retained. 5) The embeddings obtained from the K-th GCN layer are used as input, and the information on the heterogeneous partial adjacent graph is aggregated as the HPAA layer; 6) The final embeddings are obtained using layer combination; 7) The top n items that the user is most likely to like are predicted.

B. Regional Central Nodes of Communities

A regional central node of a community refers to a node located in the key position in the graph that can serve as the representative of other nodes within a community. In this paper, the CD algorithm is used to directly split the UIIHG into several non-overlapping communities, so that the internal nodes of the same community are closely connected.

It is worth noting that most of the existing CD algorithms indirectly construct user-user or item-item graph [13] based on user-item interaction information, social information or attribute information, which will cause information loss. Therefore, the proposed GCN-CMHG directly detects communities on the UIIHG. After obtaining several non-overlapping communities, the regional central nodes are selected from each community, which are connected with all other nodes in the UIIHG. Therefore, after aggregating information through GCN layers, all nodes can obtain information from other nodes within and outside their communities through these regional central nodes, which help GCN-CMHG alleviate the problem of not being able to obtain information from distant nodes and the information bottleneck problem while reducing the complexity.

LPA [12] is used to divide graphs into communities here. LPA uses graph structure as a guide and can be completed only in linear time. The basic idea is to update the node's label according to the labels of the node's neighbors so that the node can be divided into different communities. The details of LPA are as follows. In a graph composed of nodes and edges, the LPA first assigns a unique label to each node. Taking the node u_1 as an example, the frequency of label occurrence of all u_1's neighbors is recorded according to Eq. (2):

$$La_{u_1}(t) = func(La_{u_{h_1}}(t),...,La_{u_{h_{m-1}}}(t), La_{u_{h_m}}(t-1),...,La_{u_{h_{N_{u1}}}}(t-1)) \quad (2)$$

where t is the number of iterations, the first m-1 nodes are the nodes that have completed updating in the t iteration, and *func* is the function that updates the label of node u_1 to the label that appears the most times in its neighbors. The label with the most frequent occurrence is taken as its label, and nodes are divided into different communities according to labels. After several iterations, the labels of the communities become stable, and the nodes of the same community are given the same label. Finally, the node with the highest degree is selected from each community as the regional central node of the community.

C. GCN Layer Using PLNS Strategy

Similar to GCN-MHG, GCN-CMHG stacks K GCN layers on a UIIHG to aggregate information. To eliminate the impact of redundant features that not only waste time but also reduce the recommendation effect, GCN-CMHG also adopts the PLNS strategy to retain node self-connections in partial layers.

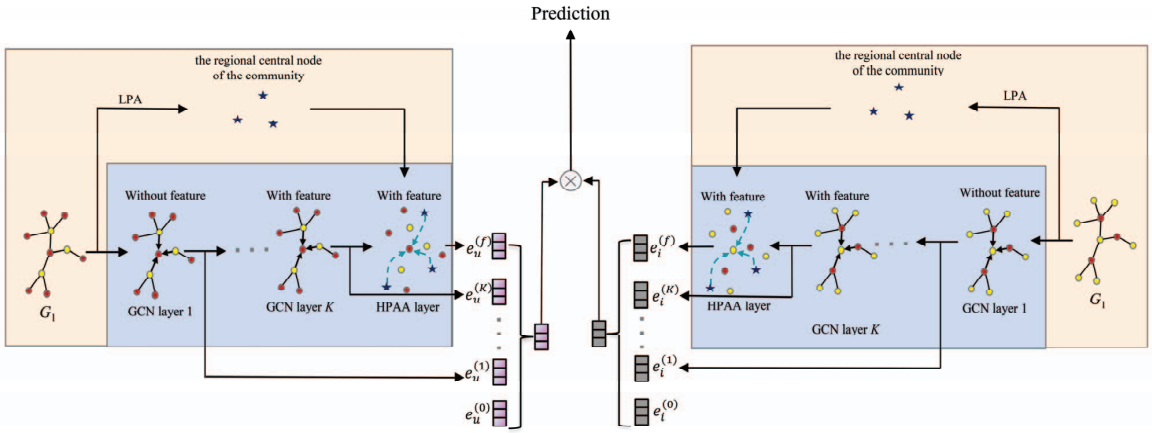

Fig. 2. Overall framework of GCN-CMHG.

Specifically, GCN-CMHG first stacks K-1 GCN layers that do not contain node self-connections [5,6]. The operation of feature transformation and nonlinear activation is abandoned, and the information transmission rules of the k-th layer are defined as shown in Eq. (3):

$$\mathbf{e}_u^{(k)} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} \mathbf{e}_i^{(k-1)}, \quad \mathbf{e}_i^{(k)} = \sum_{u \in N_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_u|}} \mathbf{e}_u^{(k-1)} \tag{3}$$

where $\mathbf{e}_u^{(k)} \in \mathbb{R}^d$ and $\mathbf{e}_i^{(k)} \in \mathbb{R}^d$ represent the d-dimensional embedding of the user node u and the item node i in the k-th layer, respectively, and N_u (N_i) represents the set of items (users) interacting with user u (item i) in the UIIHG G_1.

As the number of stacking layers increases, more and more meaningful neighbor information is gathered on nodes, and the damage of redundant features such as ID to recommendation effect is gradually offset. The GCN layer, which keeps node self-connections, is stacked as the K-th layer [6,7,14]. The information transmission rule of this layer for user u *is* defined as shown in Eq. (4):

$$\mathbf{e}_u^{(k)} = \mathbf{e}_u^{(k-1)} + \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} \mathbf{e}_i^{(k-1)} \tag{4}$$

We can obtain $\mathbf{e}_i^{(k)}$ by using the analogous information transmission rule.

D. HPAA Layer

After obtaining the regional central nodes of the communities, only the edges between the regional central node of each community and all other nodes are retained in the heterogeneous fully adjacent graph to construct the heterogeneous partial adjacent graph, and the information aggregation layer on the heterogeneous partial adjacent graph is called the HPAA layer. For example, as shown in Fig. 3. The dots in different colors represent the user and item nodes, respectively, and the five-pointed star represents the regional central node of the community. Taking the target node u_1 as an example, it can be found that the HPAA layer creates temporary channels between user u_1 and all the regional central nodes of communities, allowing the regional central nodes of communities in the whole graph to transmit information to user u_1 through these temporary channels. This not only makes the distance between user u_1 and the regional central nodes of communities closer, but also brings the target user node u_1 closer to all nodes in the communities where the regional central nodes are located.

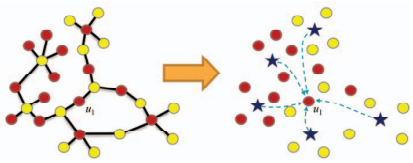

Fig. 3. Schematic diagram of the information aggregation using the HPAA layer.

In addition, the HPAA layer also increases the number of the information transmission channels, which can help GCN-CMHG alleviate the cold start and information bottleneck problem. Similar to the heterogeneous fully adjacent graph, the user-item interaction edges are not retained in the heterogeneous partial adjacent graph, so as to better examine the influence of information from the regional central nodes of communities on improving the recommendation effect, and to reduce the over-smoothing problem caused by over-reliance on the user-item interaction edge to aggregate information. In order to enable the algorithm to make full use of the neighbor information obtained through the previous GCN layer, the HPAA layer retains the node self-connections by adding the identity matrix to the matrix representing the heterogeneous partial adjacent graph.

When the HPAA layer is added after the last GCN layer, the embeddings containing the neighbor information obtained from the aggregation of the previous K GCN layers will be regarded as the node features of the heterogeneous partial adjacent graph. At this time, node features become meaningful due to the large amount of gathered neighbor information, so GCN-CMHG also retains node self-connections in the HPAA layer. The information transmission rules of this layer are defined as shown in Eq. (5) and (6):

$$\mathbf{e}_u^{(f)} = \mathbf{e}_u^{(K)} + \sum_{cen \in N_u^{(f)}} \frac{1}{\sqrt{|N_u^{(f)}|}\sqrt{|N_{cen}^{(f)}|}} \mathbf{e}_{cen}^{(K)} \tag{5}$$

$$\mathbf{e}_i^{(f)} = \mathbf{e}_i^{(K)} + \sum_{cen \in N_i^{(f)}} \frac{1}{\sqrt{|N_i^{(f)}|}\sqrt{|N_{cen}^{(f)}|}} \mathbf{e}_{cen}^{(K)} \tag{6}$$

where $\mathbf{e}_u^{(f)} \in \mathbb{R}^d$ represents the d-dimensional embedding of the user node u in the HPAA layer, *cen* represents the regional central nodes of communities, $N_u^{(f)}$ and $N_i^{(f)}$ represent the set of the regional central nodes of communities that interact with user u and item i in the heterogeneous partial adjacent graph G_2 respectively. $N_{cen}^{(f)}$ represents the set of user/item nodes that interact with the regional central node in the heterogeneous partial adjacent graph G_2.

E. Layer Combination and Prediction

To make full use of the embeddings obtained from different layers, GCN-CMHG further combines the initial embeddings and the embeddings obtained from each layer as the final embeddings, as shown in Eq. (7):

$$\mathbf{e}_u^* = a_f \mathbf{e}_u^{(f)} + \sum_{k=0}^{K} a_k \mathbf{e}_u^{(k)}, \ \mathbf{e}_i^* = a_f \mathbf{e}_i^{(f)} + \sum_{k=1}^{K} a_k \mathbf{e}_i^{(k)} \tag{7}$$

where a_k and a_f represent the importance of the embeddings output from the k-th GCN layer and HFAA layer to the formation of the final embeddings, which can be used as either manually adjusted hyperparameters or trainable model parameters. For convenience, in this paper, we let $a_f = a_k = 1/(K+1)$,where K is the number of GCN layers.

Then, as shown in Eq. (8), the inner product is calculated to complete the prediction of interaction probability:

$$\hat{y}_{ui} = (\mathbf{e}_u^*)^T \mathbf{e}_i^* \tag{8}$$

where $\mathbf{e}_u^*$ and $\mathbf{e}_i^*$ are the user and item embeddings, and $(\bullet)^T$ represents the transpose of the embedding.

F. Model Training

In this paper, the Bayesian personalized ranking (BPR) loss function is used as the loss function of GCN-CMHG, as shown in Eq. (9):

$$Loss = \sum_{(u,i,j) \in O} -\ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \beta \| \mathbf{E}^{(0)} \|^2 \tag{9}$$

where $O=\{(u,i,j)|(u,i)\in R^+,(u,j)\in R^-\}$ represents the paired training samples composed of positive sample item i and negative sample item j with and without interactive information with user u, R^+ represents positive sample set and R^- represents negative sample set. Negative samples are randomly selected, with a ratio of one-to-one to positive samples, and β is the L2 regularization coefficient.

G. Time Complexity Analysis

The time complexity of GCN-CMHG can be divided into CD algorithm LPA and GCN-based algorithm. LPA has the approximate linear time complexity O(t(M+N+|E|)), where t represents the number of iterations of the LPA, and |E| represents the number of edges. The time complexity of GCN-based algorithm can be further divided into three parts: for GCN layers that do not retain node self-connections, the time complexity is O(|R^+|h). For GCN layers that retain node self-connections, the time complexity is O(|R^+|h+(M+N)h). For the HPAA layer, the time complexity is O(|C|(M+N)h), where |C| represents the number of the communities. In GCN-MHG, the time complexity of the HFAA layer is O((M^2+N^2)h). Due to |C|(M+N) << (M^2+N^2), the proposed GCN-CMHG achieved a significant improvement in time efficiency, making it more widely applicable. Therefore, when there are K layers of GCN and only retain node self-connections in the last layer, the overall complexity of GCN-CMHG is O(t(M+N+|E|))+O((K-1)|R^+|h)+O(|R^+|h+(M+N)h)+ O(|C|(M+N)h).

IV. EXPERIMENTS

In this section, we first introduce the experimental setup. Then GCN-CMHG is compared with seven advanced comparison baselines on five datasets. A detailed comparison is made between GCN-CMHG and GCN-MHG. Finally, ablation experiments are conducted on GCN-CMHG.

A. Experimental Setup

To evaluate the model performance, several state-of-the-art algorithms are used for comparison, including the traditional classic model-based CF algorithm BPR-MF [15], the GCN-based CF algorithms GCN [14], GC-MC [6], NGCF [7], DGCF [16], LightGCN [5] and GCN-MHG [2].

We implement experiments on five datasets: Last.fm [5], Flixster [17], Douban [17], Amazon [18] and Yelp2018 [5]. The first three datasets are used by GCN-MHG, while the other two larger datasets can verify the performance of GCN-CMHG on large-scale datasets. Statistics for all five datasets are shown in Table I. Normalized Discounted Cumulative Gain (NDCG) and Recall, which are commonly used in recommendation systems, are used as evaluation indexes.

TABLE I. DATASETS STATISTICS.

Dataset	Users	Items	Interaction	Density
Last.fm	1892	4489	52668	0.0062
Flixster	3000	3000	26173	0.0029
Douban	3000	3000	136891	0.0152
Amazon	14432	28242	631762	0.0016
Yelp2018	31668	38048	1561406	0.0013

For convenience, GCN-CMHG uses the same parameters as GCN-MHG. Specifically, we use Adam as the optimizer, the default number of stacked GCN layers is 3, the learning rate is 0.001, the batch size is 2048, the length of the recommendation list is 20, the L2 regularization parameter is 0.0001, the embedding size is fixed at 64, and the embedding is initialized using the Xavier. The averaged experimental results are obtained through five independent experiments.

B. Overall Performance

In this section, GCN-CMHG is first compared with other algorithms including GCN-MHG. The experimental results are shown in Table II. The best results are shown in bold, the second best results are underlined and the third best results are shown in square brackets. The improvements of GCN-CMHG, GCN-MHG compared with LightGCN are also recorded. The numbers in brackets in the table are Standard Deviations of five results. By observing Table II, the following conclusions can be drawn:

(a) Among all comparison algorithms, LightGCN has excellent performance, and is only inferior to GCN-MHG and GCN-CMHG. The performance of GCN-MHG is second only to that of GCN-CMHG on most datasets. This is because the HFAA layer used by GCN-MHG expands the influence of the distant node on the target node, enabling the target node to receive global information.

TABLE II. OVERALL COMPARISON

Dataset / Method	Lastfm		Flixster		Douban		Amazon		Yelp2018	
	NDCG	***Recall***	***NDCG***	***Recall***	***NDCG***	***Recall***	***NDCG***	***Recall***	***NDCG***	***Recall***
BPR-MF	0.1869	0.2439	0.0906	0.1512	0.0845	0.1285	0.0581	0.0754	0.0462	0.0571
	(*0.0038*)	(*0.0050*)	(*0.0028*)	(*0.0025*)	(*0.0013*)	(*0.0019*)	(*0.0005*)	(*0.0002*)	(*0.0015*)	(*0.0023*)
GCN	0.1682	0.2275	0.0675	0.1344	0.0851	0.1287	0.0381	0.0499	0.0368	0.0447
	(*0.0019*)	(*0.0005*)	(*0.0024*)	(*0.0020*)	(*0.0051*)	(*0.0021*)	(*0.0022*)	(*0.0025*)	(*0.0003*)	(*0.0004*)
GC-MC	0.1557	0.2130	0.0604	0.1185	0.0750	0.1125	0.0379	0.0505	0.0353	0.0438
	(*0.0075*)	(*0.0068*)	(*0.0011*)	(*0.0018*)	(*0.0005*)	(*0.0064*)	(*0.0016*)	(*0.0015*)	(*0.0003*)	(*0.0004*)
NGCF	0.1885	0.2518	0.1018	0.1782	0.0853	0.1275	0.0551	0.0703	[0.0477]	[0.0579]
	(*0.0022*)	(*0.0029*)	(*0.0028*)	(*0.0050*)	(*0.0001*)	(*0.0015*)	(*0.0009*)	(*0.0009*)	(*0.0002*)	(*0.0003*)
DGCF	0.2030	0.2623	[0.1314]	[0.2196]	0.0990	0.1456	[0.0764]	[0.0963]	0.0438	0.0536
	(*0.0023*)	(*0.0034*)	(*0.0019*)	(*0.0042*)	(*0.0016*)	(*0.0015*)	(*0.0005*)	(*0.0002*)	(*0.0001*)	(*0.0005*)
LightGCN	[0.2134]	[0.2754]	0.1281	0.2130	[0.1027]	[0.1496]	0.0807	0.1029	0.0519	0.0633
	(*0.0005*)	(*0.0012*)	(0.0005)	(0.0083)	(*0.0008*)	(*0.0015*)	(*0.0037*)	(*0.0044*)	(*0.0002*)	(*0.0001*)
GCN-MHG	**0.2195**	**0.2816**	0.1330	0.2196	0.1103	0.1562	-	-	-	-
	(*0.0011*)	(*0.0009*)	(*0.0002*)	(*0.0032*)	(*0.0000*)	(*0.0010*)	-	-	-	-
	+2.9%	+2.2%	+3.8%	+3.1%	+7.3%	+4.4%	-	-	-	-
GCN-CMHG	0.2193	0.2807	**0.1344**	**0.2220**	**0.1110**	**0.1569**	**0.0835**	**0.1064**	**0.0528**	**0.0643**
	(*0.0005*)	(*0.0013*)	(*0.0044*)	(*0.0054*)	(*0.0012*)	(*0.0016*)	(*0.0026*)	(*0.0034*)	(*0.0008*)	(*0.0006*)
	+2.8%	+1.9%	+4.9%	+4.2%	+8.1%	+4.9%	+3.4%	+3.4%	+1.8%	+1.5%

In addition, GCN-MHG retains node features in the process of information aggregation of some Layers through the PLNS strategy. This not only avoids the damage to algorithm performance caused by semantically free features in many recommendation scenarios, but also enhances the influence of non-first-order neighbor nodes.

(b) On almost all datasets, GCN-CMHG has achieved state-of-the-art effect, because the HPAA layer adopted by GCN-CMHG not only helps nodes obtain distant information better, but also reduces the transmission of repeated information by using the regional central nodes of communities to construct the graph.

C. Compared with GCN-MHG

1) Does GCN-CMHG perform better than GCN-MHG? To compare the advantages and disadvantages of GCN-CMHG and GCN-MHG, we acquire three datasets used by GCN-MHG, and test the performance of GCN-CMHG and GCN-MHG when stacking different numbers of GCN layers. In this section, by default, only an HPAA/HFAA layer is added after the last GCN layer, and by default, only the last GCN layer and HPAA/HFAA layer maintain the node self-connections. To make the comparison more intuitive, LightGCN is also tested as a comparison baseline in this section. The experimental results are shown in Fig. 4, and the following conclusions can be drawn:

(a) When the number of stacked GCN layers is small, the effect of both GCN-MHG and GCN-CMHG is worse than that of LightGCN, which may be due to the non-semantic noise information introduced from the node self-connections. However, as the number of stacked GCN layers increases, the effect of both GCN-MHG and GCN-CMHG is significantly higher than that of LightGCN. This is because LightGCN can only aggregate the information from the nearest node, which proves that retaining the information of distant nodes is positive for improving the recommendation effect.

(b) No matter how the number of stacked GCN layers changes, GCN-CMHG and GCN-MHG maintain a similar change trend. This proves that although the HFAA layer used by GCH-MHG can improve the recommendation effect by enhancing the influence of distant nodes and broadening the information transmission channel, a large number of repeated and indiscriminate calculations will introduce a large number of harmful information. In contrast, the HPAA layer used by GCH-CMHG reduces the impact of redundant information while retaining as much meaningful information as possible by only using the central nodes of communities pre-selected by the LPA.

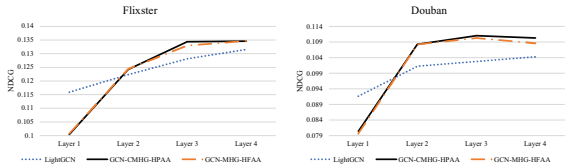

Fig. 4. Results of LightGCN, GCN-MHG and GCN-CMHG using different layers.

2) Is the time consumption of GCN-CMHG less than that of GCN-MHG? In this part, experiments are designed to compare the time consumption of GCN-CMHG and GCN-MHG. The number of GCN layers in the stack is 3 by default, only one HPAA/HFAA layer is added after the last GCN layer, and the nodes are self-connected only on the last GCN layer and the HPAA/HFAA layer by default. The experimental results are shown in Table III, and the following conclusions can be drawn:

Compared with LightGCN, the time consumed by GCN-CMHG on the three datasets is increased by +6.1%, +100.0%, and +209.5%, respectively. This indicates that stacking HPAA layers needs to cost some additional time, which is acceptable considering the improved performance. When compared with GCN-MHG, the consumption time of GCN-CMHG is only 6.7%, 7.7% and 17.7% of the former in three datasets, with an average of about 1/10. This proves that the HPAA layer can significantly reduce the running time by using the regional central nodes of communities.

TABLE III. COMPARISON EXPERIMENTS OF TIME CONSUMPTION BETWEEN GCN-CMHG-HPAA AND GCN-MHG-HFAA (*s*).

	Lastfm	**Douban**	**Flixster**
LightGCN	330	365	200
GCN-MHG	4085	4562	3507
GCN-CMHG	350	730	619

D. Ablation Experiments

1) How do different ways to use the HPAA layer affect GCN-CMHG? To eliminate interference, this part designs three variants of GCN-CMHG that do not use PLNS strategy temporarily, and sets the number of GCN layers K to 3. A variant that only adds the HPAA layer after the last GCN layer is named as HPAA-1. Adding the HPAA layer after the

last two GCN layers is called HPAA-2. Adding the HPAA layer after all three GCN layers is called HPAA-3. The experimental results are shown in Fig. 5, and the following conclusions can be drawn:

Adding HPAA layer after the GCN layer can improve the algorithm effect. However, the effect of adding HPAA layer after more GCN layers is not necessarily better, which depends on the characteristics of the dataset. When only the first several GCN layers are stacked, the regional central node of the community only contains the information of its nearest neighbor nodes. Therefore, adding HPAA to too many GCN layers cannot improve the effect, but may provide too much redundant information.

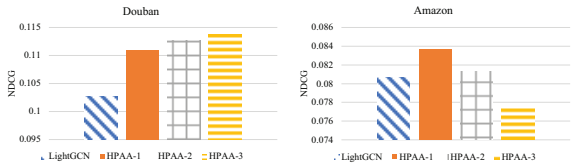

Fig. 5. Effects of different HPAA layer application methods on GCN-CMHG.

2) How do different combinations of HPAA layers and PLNS strategies affect GCN-CMHG? In this part, experiments are designed to verify the influence of the combination of different HPAA layers and PLNS strategies on the effect of GCN-CMHG. To eliminate interference, this part sets the stacking layer number K of GCN to 3, and names the nine variants in a similar way to the previous section. The experimental results are shown in Fig. 6, and the following conclusions can be drawn:

As the PLNS strategy is fixed, retaining only the HPAA layer after the last GCN layer enables GCN-CMHG to improve the effect without increasing the time consumption. As GCN-CMHG adds the HPAA layer after the last GCN layer, it retains the node self-connections in the last GCN layer (namely HPAA-1-PLNS-1), which obtains a good effect. However, considering the differences of datasets, the specific combination of HPAA layer and PLNS strategy needs to be analyzed according to the actual situation.

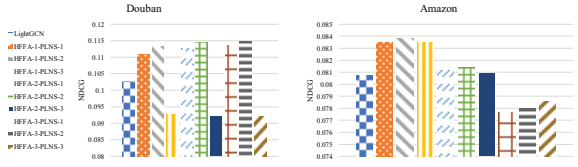

Fig. 6. Effects of different combinations of HPAA layers and PLNS strategies on GCN-CMH.

V. CONCLUSION

In this paper, a Graph Convolutional Neural Network for Recommendation Based on Community Detection and the Combination of Multiple Heterogeneous Graphs (GCN-CMHG) is proposed, which can fully mine sparse user-item interaction information at a low time, so as to improve the recommendation effect. GCN-CMHG first uses the community detection algorithm to detect communities, finds the regional central nodes of communities in the user-item interaction heterogeneous graph, and then constructs the heterogeneous partial adjacent graph. A Heterogeneous Partial Adjacent Auxiliary (HPAA) layer is designed to aggregate information on the graph. Experiments on real-world datasets show that the GCN-CMHG can not only expand the influence of distant nodes on target nodes and alleviate the information bottleneck problem, but also greatly reduce the time complexity of model training.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (Nos. 62077038, 61672405, 62176196 and 62271374).

REFERENCES

[1] R. Wang, Y. Jiang, and J. Lou, "ADCF: Attentive representation learning and deep collaborative filtering model," Knowledge-Based Systems, vol. 227, 2021, pp. 107194.

[2] C. Mu, H. Huang, Y. Liu, and J. Luo, "Graph convolutional neural network based on the combination of multiple heterogeneous graphs," in 2022 IEEE International Conference on Data Mining Workshops, 2022, pp. 724-731.

[3] X. Zhang, S. Liu, and H. Wang, "Personalized learning path recommendation for e-learning based on knowledge graph and graph convolutional network," International Journal of Software Engineering and Knowledge Engineering, vol. 33, no. 1, 2023, pp. 109-131.

[4] X. Liu, M. Yan, L. Deng, G. Li, and D. Fan, "Sampling methods for efficient training of graph convolutional networks: A survey," IEEE/CAA Journal of Automatica Sinica, vol. 9, no. 2, 2022, pp. 205-234.

[5] X. He, K. Deng, X. Wang, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in SIGKDD, 2020, pp. 639-648.

[6] R. Berg, T. Kipf, and M. Welling, "Graph convolutional matrix completion," in KDD Deep Learning Day, 2018.

[7] X. Wang, X. He, M. Wang, F. Feng, and T. S. Chua, "Neural graph collaborative filtering," in SIGKDD, 2019, pp. 165-174.

[8] Z. Chen, A. Sun, and X. Xiao, "Incremental community detection on large complex attributed network," ACM Transactions on Knowledge Discovery from Data, vol. 15, no. 6, 2021, pp. 1-20.

[9] M. Girvan, and M. Newman, "Community structure in social and biological networks," Proceedings of the National Academy of Sciences of the United States of America, vol. 99, no. 12, 2002, pp. 7821-7826.

[10] M. Newman, and M. Girvan, "Finding and evaluating community structure in networks," Physical review E, vol. 69, no.2, 2004, pp. 026113.

[11] C. Mu, J. Zhang, Y. Liu, R. Qu, and T. Huang, "Multi-objective ant colony optimization algorithm based on decomposition for community detection in complex networks," Soft Computing, vol. 23, 2019, pp. 12683-12709.

[12] U. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," Physical Review E Statistical Nonlinear & Soft Matter Physics, vol. 76, no. 3, 2007.

[13] V. Satuluri, Y. Wu, X. Zheng, Y. Qian, and J. Lin, "SimClusters: Community-based representations for heterogeneous recommendations at Twitter," in SIGKDD, 2020, pp. 3183-3193.

[14] T. Kipf, and M. Welling, "Semi-supervised classification with graph convolutional networks," in ICLR, 2017.

[15] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, 2009, pp. 452-461.

[16] Z. Liu, L. Meng, J. Zhang, and P. Yu, "Deoscillated Adaptive Graph Collaborative Filtering," arXiv preprint arXiv:2011.02100, 2020.

[17] M. Federico, B. Michael, and B. Xavier, "Geometric matrix completion with recurrent multi-graph neural networks," Advances in Neural Information Processing Systems, 2017, pp. 3700–3710.

[18] R. He, and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in WWW, 2016, pp. 507–517.