Received 15 April 2024, accepted 11 May 2024, date of publication 17 May 2024, date of current version 24 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3402210

RESEARCH ARTICLE

Cliques of Graph Convolutional Networks for Recommendation

ZHENYE PAN AND YAHONG CHEN
School of Mathematics and Computer Science, Lishui University, Lishui 323000, China
School of Computer Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China

Corresponding author: Yahong Chen (yhchen0611@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 12371349, and in part by Zhejiang Provincial Natural Science Foundation of China under Grant LY21A010002.

ABSTRACT Graph neural networks have become a popular technique for collaborative filtering. However, most related work is based on user-item bipartite graphs, which can generate a large amount of noise due to the broad and elusive interests of users. To address this problem, we propose a novel generalized insertion framework (CGCN) that directly captures cliques in the item-item co-occurrence graph and considers them as the basic units of the user's higher-order semantics. The method inserts the structural information in these item-item co-occurrence graphs as an insertion module into the original user-item bipartite graph propagation process, thus providing additional useful information to learn better feature representations. By utilizing the strong proximity relationships between different items in these cliques, the method is able to discover the user's potential higher-order semantics. We experimentally evaluate two improved variants of the framework on three commonly used public datasets, and the results show significant performance improvements. The method is able to better discover users' latent true intentions and achieve better recommender system performance by introducing clique information in the item-item co-occurrence graph.

INDEX TERMS Graph neural networks, clique, collaborative filtering, recommender systems.

I. INTRODUCTION

With the rapid development of e-commerce, online news, and social media, personalized recommendation has become an indispensable and important tool for many enterprises [1], [2], [3], [4], [5]. The core of the personalized recommendation task is to accurately match users with candidate items, so as to recommend content that users are more likely to interact with. Collaborative filtering (CF) [6], [7], [8], [9], which is commonly used for personalized recommendation, is based on the idea of learning the embeddings of users and items from their historical interaction information, and computing the scores of the two based on the pairwise similarity between the embeddings of the user and the item, and using the scores as the basis for recommendation [10].

Recent studies have shown that user-item relationships can be naturally represented as graph structures, such as user-item bipartite graphs or item-item co-occurrence graphs. Many recent studies have used graph neural networks (GNNs) to learn node embedding representations of users and items [11], [12]. GNNs can naturally handle the interaction between users and items and learn embedding representations of users and items by aggregating relevant neighborhood information, thus achieving good performance gains on many recommendation tasks [13].

Most of the current GNN-based CF models [14], [15], [16] are directly based on the basic user-item bipartite graph, on which different constructions of message-passing layers are explored, including the processes of neighborhood aggregation, message passing, and feature transformation. However, in addition to user-item relationships, studies have shown that other relationships (e.g., item-item relationships, user-user relationships) are also beneficial to improve the performance of GNN-based CF models [17], [18]. User-item bipartite graphs based on user-item bipartite graphs are able to implicitly learn these different relationships,

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wang.

but this often leads to noise and over-smoothing problems. For example, the Gowalla dataset, which is widely used in collaborative filtering work, has an average degree of the graph of about 23. Since user nodes are alternately connected to item nodes in a user-item bipartite graph, conveying information about another item neighboring one item (which, in the interpretable sense, means the user interacting with that item, the other item with which it interacts) requires 2-hops, and this implies introducing information about more than 500 item nodes, which will inevitably introduce noise, as well as making it difficult to distinguish from a large number of item nodes the neighboring nodes that are more important to that node. To alleviate these problems, some related works [17], [18], [19], directly use item-item co-occurrence graphs with user-user co-occurrence graphs to explicitly learn the interrelationships between different users and different items, and integrate the additional information learned from the different relationship graphs for recommendation tasks.

However, propagating information directly in the item-item co-occurrence graph or in the user-user co-occurrence graph is very difficult. This is because edges in the co-occurrence graph tend to be much denser than those in the user-item bipartite graph. Taking Gowalla as an example, in Gowalla's item-item co-occurrence graph, the average degree of each item node is about 1033, which is 45 times that of the original graph. Too dense edges often lead to training difficulties with over-smoothing problems. Meanwhile, for an item node, it is unfair to treat all its neighbor nodes equally, and in real application scenarios, there are often a large number of episodic user-item interactions, which will lead to much noise. On the other hand, how to fuse information from multiple graphs is also challenging, and related work [17], [18] tends to use complex graph encoders and attention mechanisms for aggregation, which increases training difficulty and leads to poor interpretability.

Graph neural networks are able to naturally capture higher-order connectivity, and the results of different layers of graph neural network aggregation can be interpreted as having unique meanings [1], [20]. For example, for an item node, its one-hop neighbor aggregation is the aggregation of all users who have interacted with the item, and its two-hop neighbor aggregation is the aggregation of other interacted item nodes of users who have interacted with the node. However, there is some clustering and structural information present in the graph itself that cannot be captured explicitly, and which may imply higher-order semantic information, the use of which is often more beneficial [18], [21]. For example, a user who has purchased a CPU, memory sticks, and motherboard may represent the user's true intention to purchase a computer. Recommending other components of a computer, such as a hard disk, to the user at this point may be more consistent with the user's true intent. However, traditional GNN-based recommendation models do not utilize this higher-order semantic information to make recommendations, but instead make recommendations through other items purchased by other users who have purchased these items.

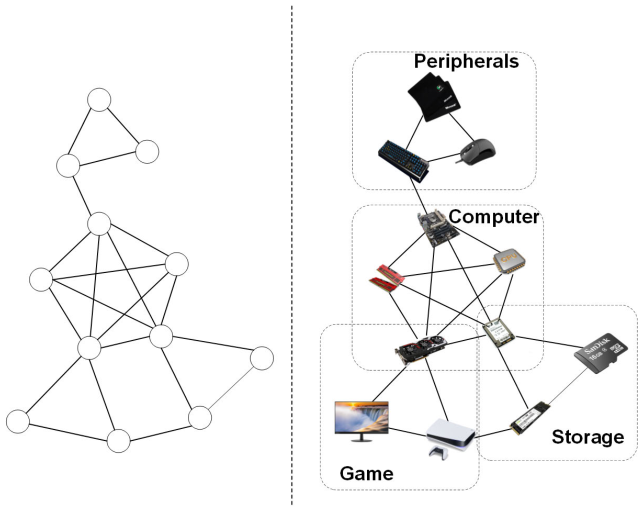


FIGURE 1. Clique structures in the item-item co-occurrence graph. Our approach introduces different cliques to reflect different real intentions of the user, e.g., the items in a clique in the figure on the right may have similar roles (Storage), potential use intentions (Game), or constitute a whole (Computer).

Are there ways to learn using only one graph and skillfully utilize the information in other graphs? Most existing collaborative filtering recommendation models based on graph neural networks apply graph neural networks directly to the original user-item bipartite graph. Although powerful, graph neural networks are able to naturally learn structural information from graphs, there may be validity issues in this recommender system. Unlike traditional graph neural network research objects, where ordinary graph nodes and their own neighbors belong to the same kind of nodes with similar attributes, the two-part graph in the recommender system is constructed by the user and the project of two different types of nodes alternatively connected. This construction presents significant differences in the connectivity and attributes of the nodes.

As a consequence, the original graph structure may prove inadequate for effectively learning user and item representations. Numerous researchers have proposed methods to address the limitations of the original bipartite graph structure by enhancing it. Nevertheless, determining how to enrich the bipartite graph structure and establishing criteria for determining whether to create new edges between nodes appears to pose significant challenges.

In this paper, we leverage the cliques within the item-item co-occurrence graph to identify item nodes with the highest degrees of similarity. By directly propagating their feature information from these cliques to the user nodes, we aim to enhance recommendation accuracy and effectiveness.

Clique is an important structure in complex graph analysis in different fields, such as social networks [22] and protein networks [23]. Many existing works in graph theory have mentioned the properties of cliques [24], [25]. A clique can be considered as a complete subgraph of the original graph, i.e., there exists an edge between any two vertices in the clique. Different sizes of cliques are widely present in the item-item co-occurrence graph. Nodes in a clique tend to be homogeneous in one way or another, while heterogeneity exists between different cliques. These cluster structures can be utilized for recommendation tasks. Specifically, an item node can be contained in a number of different cliques; different cliques co-occur with different sets of trait attributions of the items and are able to express different attributes of the items; whereas a single clique contains multiple items, reflecting the proximity of these items in terms of a certain attribute. Fig.1 visualizes the nature of cliques in the item-item co-occurrence graph.

Therefore, we extract cliques from the item-item co-occurrence graph and use the features of all the item nodes within the clique to obtain the features of that clique. The higher-order features of the item node are then derived by calculating the features of all the cliques to which the item node belongs. Finally, we combine the user's features, the item's features, and the item's higher-order features to predict the user's preference for the item. The inner product score of a user's features and an item's features reflects the user's preference for that item, while the inner product score of a user's features and an item's higher-order features reflects the user's preference for the higher-order semantics of that item, resulting in a combined score that can be used to better perform recommendation tasks.

The general propagation process is depicted in the upper section of Fig.2. Here, the red node represents the target node,

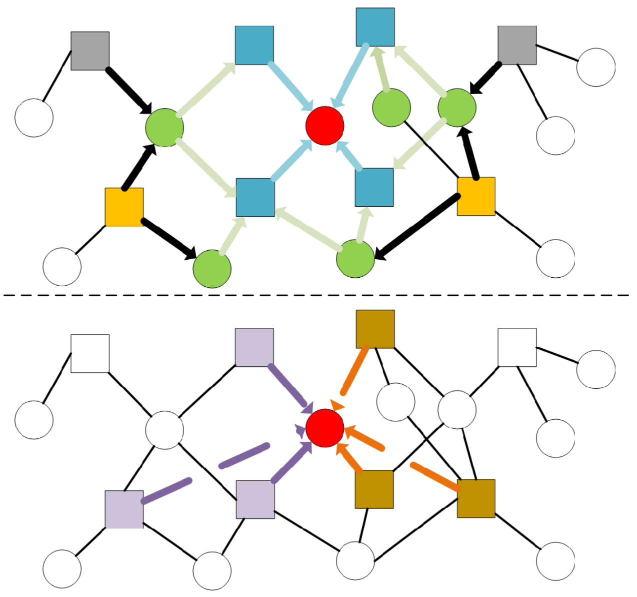


FIGURE 2. Comparative graph of propagation processes. The top half is the original propagation process and the bottom half is the new propagation process.

while the blue, green, yellow, and gray nodes symbolize its 1-hop, 2-hop, and 3-hop propagating neighbors, respectively. It's observable that the information from the target node iteratively spreads from its 3-hop neighbors.The updated propagation process is illustrated in the lower section of Fig.2. Here, the purple and brown nodes respectively represent nodes within two distinct cliques. It can be observed that in the upper part of Fig.2, the two gray nodes have been excluded, while the two yellow nodes are retained. This decision is based on the closer relationships between the yellow nodes and the other blue nodes; at least, they are connected to two blue nodes through a user node, collectively forming two separate triangular cliques. Conversely, the gray nodes are more likely to be incidental links stemming from users' broad interests and are thus excluded.Our aim is precisely to eliminate these edges that arise from incidental user interests and retain only those items that have closer connections.

The main contributions of the paper are summarized as follows:

1) We extract cliques from the item-item co-occurrence graph and estimate the user's preference for nodes in the cliques by considering these cliques as the user's hidden higher-order semantic units. Thus, we solve a series of problems caused by the overly dense item-item co-occurrence graph, as well as the problems of poor interpretability and fusion difficulties caused by the simultaneous use of multiple graph structures.

2) We propose a concise, novel, and generalized graph neural network-based recommender system framework, CGCN, which inserts an insertion module that uses cliques to estimate the user's higher-order semantic preferences for items directly into the original recommender system. Thereby, the framework indirectly uses the structural information in the item-item co-occurrence graph while still using only the user-item bipartite graph, which enables better learning of the feature representations of user and item nodes.

Given that the proposed generalized framework focuses on improving GNN recommender systems based on user-item bipartite graphs that are heavily deployed in reality, we select two classic user-item bipartite graph-based recommender system models, the NGCF [15] and LightGCN [14], as the basic models to demonstrate the improvement. We use the improved framework with plug-in improvements for recommendation tasks and conduct an empirical study on three public datasets. The results show that the improved results using the framework significantly outperform the original method on all datasets.

II. RELATED WORK

In this section, we briefly review some representative GNN-based approaches and the work most relevant to the paper.

With the development of GNNs in various machine learning domains, GNNs have achieved remarkable success in recommender systems over the past few years [26], [27], [28], [29]. Ying et al. [30] first applied graph convolutional neural networks (GCNs) to recommender systems for large-scale networks, proposing a method PinSage that combines random wandering and graph convolutional method PinSage, so that the embedding contains both structural and feature information of the graph. Wang et al. [15] designed a graph-based collaborative filtering framework NGCF to propagate the embedding information in user-item bipartite graphs, so as to utilize the higher-order connectivity information of the user and the item to capture their collaborative signals. He et al. [14] found that the nonlinear activation in the NGCF and the feature transformations in NGCF are unnecessary for collaborative filtering and even lead to performance degradation, thus removing the nonlinear activations and feature transformations proposed LightGCN.

Although GNN-based CF recommendation models have achieved impressive results, however, some recent work has found that applying GNN directly to the original user-item bipartite graph has effectiveness problems, and the original graph structure may not be sufficient to learn the user and item representations [17], [31].

One solution strategy to this problem is to enrich the original graph structure by adding edges. Multi-GCCF proposed by Sun et al. [17] adds edges between the two neighbors of the original graph to obtain a user-user co-occurrence graph and an item-item co-occurrence graph, and uses all three graphs simultaneously for the recommendation task. The DHCF proposed by Ji et al. [8] introduces hyperedges and constructs user-item hypergraphs to capture explicit mixed higher-order correlations. UltraGCN proposed by Mao et al. [18] similarly uses an item-item co-occurrence graph to enrich the interaction information between users and items, but they argue that the user-user co-occurrence graph is not beneficial due to the diversity and complexity of users' interests. Another strategy is to represent the enriched features of users and items by introducing virtual nodes. Wang et al. [32] considered user-item relationships at a finer granularity and proposed DGCF, which represents nodes from different perspectives by introducing virtual intent nodes and decomposing the original graph into subgraphs corresponding to each intent. Li et al. [5] represented nodes from different perspectives by clustering similar users and items with the clustering centers as new nodes to create new coarsened user-item bipartite graphs to explicitly capture the hierarchical relationships between users and items. Jiang et al. [33] proposed TGIN for click-through rate prediction task by sampling triangles on item-item co-occurrence graphs and treating these triangles as the most basic user interest units.

Inspired by these instructive studies, in the paper, we sample cliques on the item-item co-occurrence graph, utilize these cliques to show the hidden real intentions behind the users, and propose a general recommendation framework CGCN for better implementation of recommendation tasks.

III. PROPOSED APPROACH

In this section, we explain the process of capturing cliques in the item-item co-occurrence graph and how this information can be utilized for recommendation tasks. The framework diagram for the overall process is illustrated in Fig.3.

A. CAPTURE CLIQUES FROM ITEM-ITEM CO-OCCURRENCE GRAPHS

To begin with, we must obtain the item-item co-occurrence graph with weights from the user-item bipartite graph. In recommender systems, there are typically a series of interaction relationships between users and items. Let U be the set of users, N be the set of items, and $R \in \mathbb{R}^{N \times M}$ be the user-item interaction matrix, where N is the number of users and M is the number of items. If user u has interaction behavior with item i, then $R_{ui} = 1$; otherwise $R_{ui} = 0$. The adjacency matrix A of the user-item bipartite graph can then be expressed as:

$$A = \begin{bmatrix} \mathbf{0} & R \\ R^{\mathsf{T}} & \mathbf{0} \end{bmatrix} \tag{1}$$

We can get the item-item co-occurrence graph $G = \{V, E\}$ by coarsening all user nodes in the user-item bipartite graph, however, this implies a huge time overhead. Fortunately, the adjacency matrix of the item-item co-occurrence graph can be obtained directly through matrix operations:

$$A = R^T R \tag{2}$$

It is important to note that, for any two points in G, the weights of the edges between them are equal to the number of common neighboring user nodes in the user-item bipartite graph for those two items.

In the next step, we sort all the edges of each node in graph G according to their weights. We then set a window size to filter the edges of each node, keeping only those edges with the highest weights that are connected to neighbors that are more important to that node. To achieve this, we remove the lowest-weighted edges in each node by randomly cycling through them until the number of edges per node is less than or equal to the window size. This approach has the advantage of allowing smaller degree vertices to retain more edges for the next step of sample cliques.

Finally, we partition the clique K_n from the item-item co-occurrence graph G. K_n is a complete subgraph of G with edges between any two vertices in K_n. In practice, we sequentially sample cliques starting from a larger size up to a sample size of 3. For a larger clique that has already been sampled, we do not sample smaller subcliques of it. This is because find cliques in a graph is an NP-complete problem. Additionally, there are $k - 1$ k-cliques in a k-clique, which can lead to efficiency and duplication problems. The exact algorithmic pseudo-code steps are shown in Algorithm 1

In Fig.4, we show a case where the item-item co-occurrence graph is first obtained from the projection, then the edges of the item-item graph are sampled, and finally

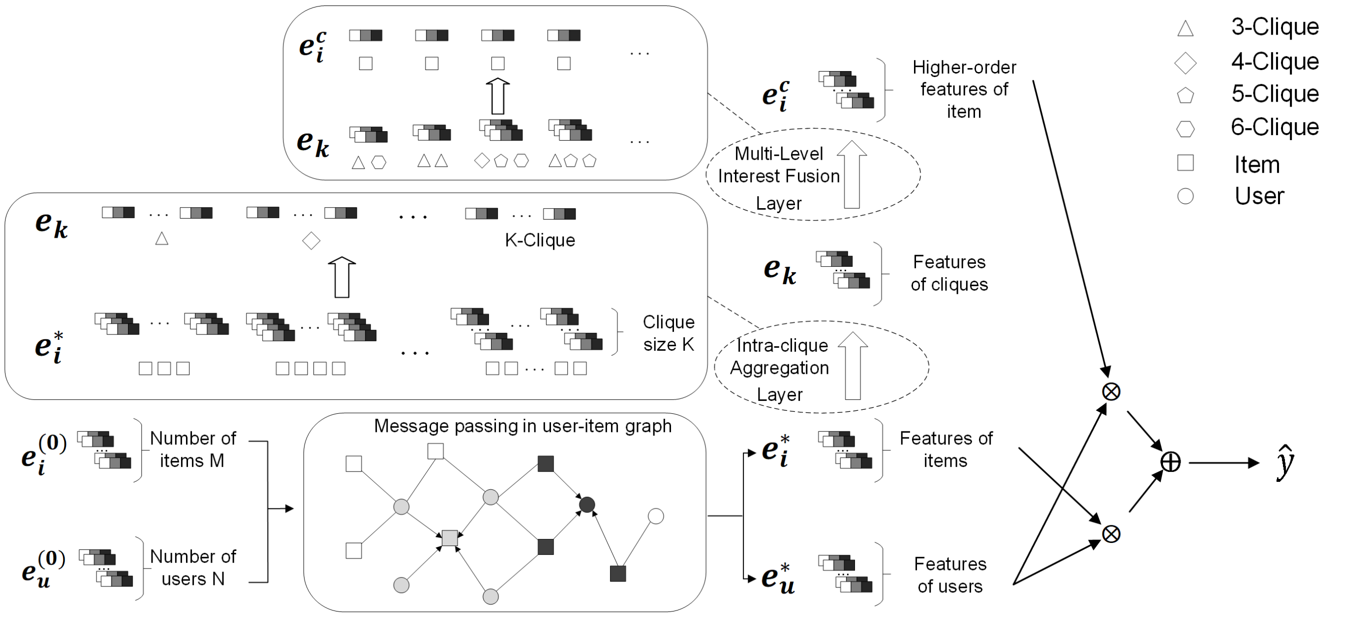


FIGURE 3. Framework Architecture Diagram.Most important in the diagram are the two layers intra-clique Aggregation Layer and multi-Level Interest Fusion Layer, which together form the plug-in module. They calculate the higher-order features of items by utilizing the information in the cliques with the features of the items.

Algorithm 1 Finding All Cliques in a Graph G_{sample}

Require: Simple graph G_{sample}, maximum clique size *max*, minimum clique size *min*

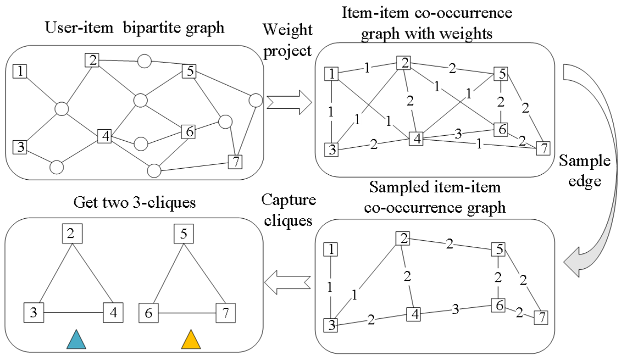
Ensure: Collection of cliques of different sizes $\{K_{\max}, \ldots, K_{\min}\}$

```
1:  $Nodes \leftarrow$ out-of-order vertices from $G_{\text{sample}}$
2:  $p \leftarrow max$
3:  while $p \geq min$ do
4:      while $Nodes \neq \emptyset$ do
5:          $node \leftarrow$ select a node from $Nodes$
6:          $neighbors \leftarrow$ gets all of $node$'s neighbors
7:          while $|neighbors| \geq p$ do
8:              $K_i \leftarrow$ use search algorithms to find a clique
9:              $neighbors \leftarrow K_i \setminus \{neighbors\}$
10:             add $K_i$ to $K_p$
11:         end while
12:         remove $node$ from $Nodes$
13:     end while
14:     $p \leftarrow p - 1$
15: end while
```

the cliques are obtained from the sampled item-item bipartite graph.

B. EMBEDDING LAYER

An item may be contained in more than one clique, reflecting the different higher-order semantics to which it belongs. However, too many cliques can lead to efficiency problems as well as noise, so it is necessary to keep only a certain number of cliques for each item node. Thus, we only keep



User-item bipartite graph
Weight project
Item-item co-occurrence graph with weights
Sample edge
Get two 3-cliques
Capture cliques
Sampled item-item co-occurrence graph

FIGURE 4. Capture cliques from item-item co-occurrence graph.

those cliques that are important to reflect its diversity and at the same time balance efficiency. The embedding layer is consistent with mainstream recommendation models [7], [14], [15].The improved framework also uses the embedding vector $e_u \in \mathbb{R}^d$ $\left(e_i \in \mathbb{R}^d\right)$ to describe the features of a user u(item i), where d denotes the size of the embedding. The embeddings of all users can be represented as:

$$E_u = [\mathbf{e}_{u1}, \cdots, \mathbf{e}_{uN}] \tag{3}$$

where N is the number of users.Similarly, the embedding of all items can be represented as:

$$E_i = [\mathbf{e}_{i1}, \cdots, \mathbf{e}_{iM}] \tag{4}$$

where M is the number of items. The embedding layer setup of our framework is consistent with the mainstream recommendation model setup. There are no additional embedding layer parameters that need to be learned throughout the framework, which reflects the generality of our proposed plug-in framework.

C. MESSAGE PASSING

The main idea of GNN is to iteratively aggregate feature information from neighbors during the propagation process and integrate the aggregated information with its own representation [21]. In general, the process of aggregation can be represented as follows:

$$n_v^{(l)} = \text{Aggregator}_l\left(\left\{e_u^{(l)}, \forall u \in \mathcal{N}_v\right\}\right) \tag{5}$$

The process of Information update can be represented as follows:

$$e_v^{(l)} = \text{Updater}_l\left(e_v^{(l-1)}, \mathbf{n}_v^{(l)}\right) \tag{6}$$

where $e_v^{(l)}$ denotes the node feature representation of node v at layer l, $\mathbf{n}$ is the set of all neighbors of the node, and $Aggregator_l$ and $Updater_l$ denote the aggregation and information update operation functions at layer l.

The GNN then operates by stacking different layers of information through multiple iterations to obtain the final representation of the nodes for the prediction task. Most existing works [14], [15], [18], [32] focus on improving this information propagation process and propose a large number of excellent propagation models. The improved plug-in framework is able to directly adapt to these excellent propagation models, and the propagation process on user-item bipartite graphs can be represented as:

$$(\mathbf{e}_u^*, \mathbf{e}_i^*) = \text{GNNmodel}\left(\mathbf{e}_u^{(0)}, \mathbf{e}_i^{(0)}\right) \tag{7}$$

where $\mathbf{e}_u^*$ the final user node feature representation obtained by propagation in the user-item bipartite graph, $\mathbf{e}_i^*$ denotes the final item node feature representation obtained by propagation in the user-item bipartite graph, and GNNmodel is the propagation model used on the user-item bipartite graph.

D. INTRA-CLIQUE AGGREGATION LAYER

Given a set of cliques generated by relevant and varied behaviors between users and items, we use these cliques to represent units of interest behind the user.A clique consists of multiple items and can be represented as follows:

$$\left\{i_1, \cdots, i_p\right\} \in \text{Clique}_j \tag{8}$$

where $\left\{i_1, \cdots, i_p\right\}$ is all the items that make up Clique_j, and p is the size of the Clique_j.After the propagation in the previous step, the feature of item $i_1, \cdots, i_p$ at this point is obtained as $e_{i1}^*, \cdots, e_{ip}^*$.We use a simple pooling operation to aggregate the internal information within the cliques and generate a representation of that Clique_j.

$$e_{kj} = \text{Average}\left(e_{i1}^*, \cdots, e_{ip}^*\right) \tag{9}$$

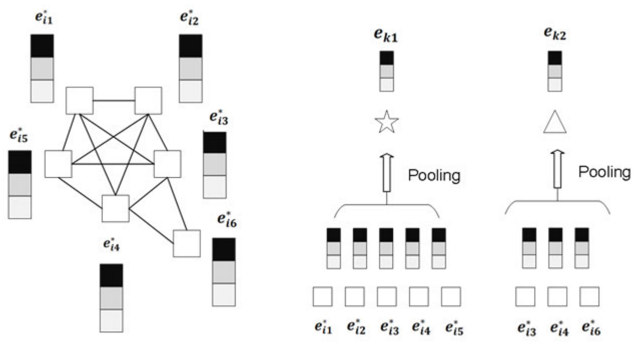


FIGURE 5. Intra-clique aggregation layer illustration.The figure depicts one 5-clique and one 3-clique, formed by aggregating 5-item and 3-item nodes, respectively.

where $e_{kj} \in \mathbb{R}^d$,represents the final embedding of Clique_j, which also representation the higher-order characteristics of all the internal item nodes within that clique.Average denotes the mean pooling operation, i.e., averaging the features of all the item nodes in the clique. In addition, other pooling operations with replacement invariance such as maximum pooling, minimum pooling, etc. are also available to adapt different representation needs.

It is worth noting that since cliques of different sizes are captured during the sampling process, it is necessary to exploit these heterogeneous cliques by sequentially aggregating cliques of different sizes and making them all end up having feature vectors of the same size. Finally, after aggregating all cliques of different sizes in turn, the embedding matrix representing all cliques is as follows:

$$e_k = \{e_{k1}, \cdots, e_{kl}\} \tag{10}$$

where $e_k \in \mathbb{R}^{L\times d}$,and l is the number of cliques.$e_{k1}, \cdots, e_{kl}$ are the features of $\text{Clique}_1 \cdots \text{Clique}_l$.

E. MULTI-LEVEL INTEREST FUSION LAYER

As mentioned earlier, different cliques represent different user interests. An item may be contained in several different cliques at the same time, which also represent the attributes of each different dimension of the item. Therefore, we need to synthesize all the different cliques to which an item node belongs to get a higher-order feature representation of the item. For example, item i_q may simultaneously belong to multiple different cliques.

$$i_q \in \text{Clique}_n, \cdots, \text{Clique}_m \tag{11}$$

where $\text{Clique}_n, \cdots, \text{Clique}_m$ are all cliques containing item i_q. Therefore, in order to obtain the higher-order feature representation of an item, it is necessary to derive the final higher-order feature by pooling all clique features. Again, we simply use a pooling operation to compute the higher-order features of the items.

$$e_{ip}^c = \text{Average}\left(e_{kn}, \cdots, e_{km}\right) \tag{12}$$

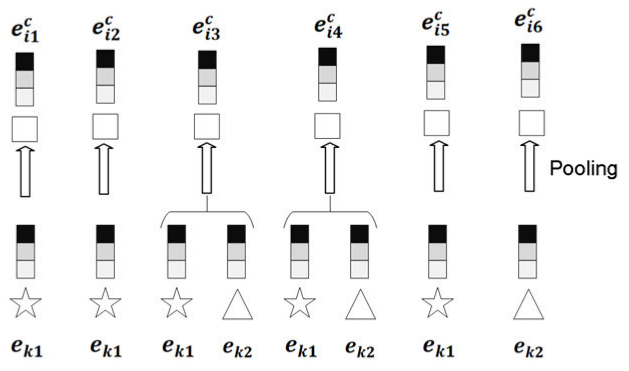


FIGURE 6. Multi-level interest fusion layer illustration. For the example in Figure 5,the feature representations of the six item nodes are obtained by aggregating them separately.

where e^c_{ip} is the final obtained higher-order feature of item i_q, and $e_{kn}, \cdots, e_{km}$ are the features of all the cliques containing item i_q. Similar to the process of mapping from item features to clique features, other pooling operations can be used in this step, but generally consistent with the pooling operation in the previous step.

The final higher-order features of an item are obtained by pooling the features of all cliques associated with it. The final higher-order feature matrix representing all items is obtained as follows:

$$e^c_i = \left\{e^c_{i1}, \cdots, e^c_{il}\right\} \tag{13}$$

where $e^c_i \in \mathbb{R}^{L\times d}$, and l is the number of items. $e^c_{i1}, \cdots, e^c_{il}$ are the features of $i_1, \cdots, i_q$.

F. LAYER COMBINATION AND MODEL PREDICTION

In the previous two steps, we described how to compute the higher-order features of an item using the features of the item node. Together, they form our clique information insertion module. Finally, we combine the learned user features, item features, and higher-order features of items for the recommendation task. We use the inner product to estimate the user's preference for the target item:

$$\hat{y} = \mathbf{e}^{*\mathsf{T}}_u \mathbf{e}^*_i + \alpha \mathbf{e}^{*\mathsf{T}}_u \mathbf{e}^c_i \tag{14}$$

where α is the combination coefficient representing the information weight of the user's higher-order intention to balance its relative importance. The first part $\mathbf{e}^{*\mathsf{T}}_u \mathbf{e}^*_i$ represents the user's preference score for the item, and the second part $\alpha \mathbf{e}^{*\mathsf{T}}_u \mathbf{e}^c_i$ represents the user's preference score for the higher-order semantics of the item.

The overall preference is computed using just a simple inner product and a constant coefficient to balance the importance of the two parts, thus making it concise and easy to understand. We also note that better performance might be obtained by using some other way of integrating information, such as the attention mechanism. More complex ways of integrating information are explored in subsequent work.

G. OPTIMIZATION

To optimize the model parameters, related work tends to use Bayesian Personalized Ranking (BPR) loss [34] as a loss function. This loss function encourages observed interactions to have higher predictive values than unobserved interactions. The objective function is as follows:

$$Loss = \sum_{(u,i,j)\in\mathcal{O}} -\ln\sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda\,||\Theta||^2_2 \tag{15}$$

where $\mathcal{O} \in \left\{(u,i,j)|(u,i) \in \mathcal{R}^+, (u,j) \in \mathcal{R}^-\right\}$ denotes the pairwise training data, $\mathcal{R}^+$ indicates the observed interactions, and $\mathcal{R}^-$ is the unobserved interactions; $\sigma(\cdot)$ is the sigmoid function; Θ denotes all trainable model parameters, and λ controls the L_2 regularization strength to prevent overfitting.

Since the CGCN framework uses the same propagation model as the original method, the BPR loss function can be used, or it can be aligned with the loss function of the original method. If we ignore the final step of calculating the higher-order features of the item nodes, we will find that the propagation process under the framework is exactly the same as the original method. So in most cases, we will directly use the loss function of the original model. The advantage of doing so is that we do not need to adjust most of the hyperparameters, including the learning rate, regularization coefficient λ, batch size, number of layers in the propagation layer, and so on. It is entirely possible to follow the hyperparameters of the base model work under the improved framework.

IV. EXPERIMENTS

In this section, we apply the CGCN framework directly to the two most classical GNN-based CF models, NGCF [15] and LightGCN [14], and compare the results with the original models to verify the effectiveness of the improved approach.

TABLE 1. Statistics of the datasets.

Dataset	User	Item	Interaction	Density
Amazon-Book	52643	91599	2984108	0.062%
Yelp2018	31668	38048	1561406	0.130%
Gowalla	29858	40981	1027370	0.084%

A. DATASETS AND EVALUATION PROTOCOL

To ensure the fairness and comparability of the experiments, we follow the experimental setups of NGCF and LightGCN, including all three datasets in the original papers, as well as the segmentation methods for the training and validation sets. This approach is consistent with other related work on GNN-based CF.

For the evaluation protocol, we choose Recall@20 and NDCG@20 as evaluation metrics because they are widely used to measure the performance of GCN-based CF models. We consider all items that are not interacted with users as candidates and report the best results for all users.

TABLE 2. Overall Performance and Improvement Comparison.

Dataset	Amazon-Books		Yelp2018		Gowalla	
Method	Recall@20	NDCG@20	Recall@20	NDCG@20	Recall@20	NDCG@20
NGCF	0.0344	0.0263	0.0579	0.0477	0.1570	0.1327
NGCF+	0.0381	0.0295	0.0624	0.0518	0.1683	0.1403
Improv.	10.76%	12.17%	7.78%	8.60%	7.20%	5.73%
LightGCN	0.0411	0.0315	0.0649	0.0530	0.1830	0.1554
LightGCN+	0.0455	0.0350	0.0689	0.0564	0.1888	0.1603
Improv.	10.71%	11.11%	6.16%	6.42%	3.17%	3.15%

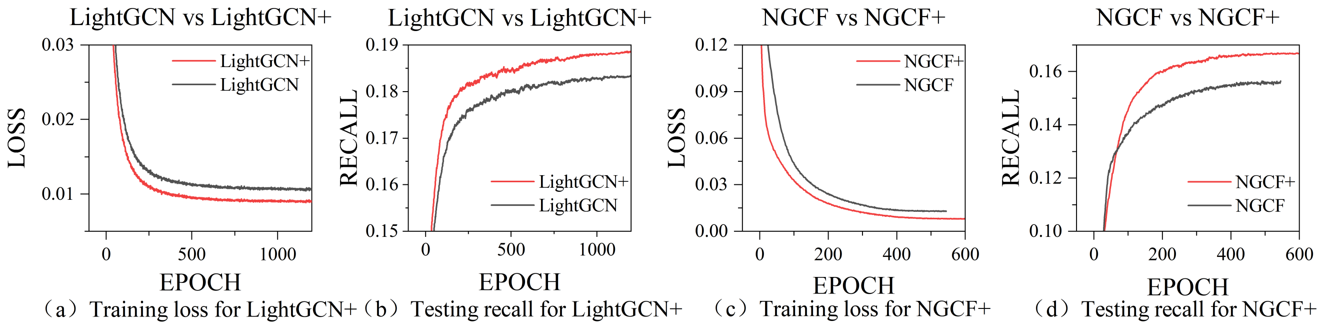


FIGURE 7. Training curves (training loss and testing recall) of LightGCN and NGCF and their three improved variants.

B. COMPARED METHODS

We use two classical models, NGCF and LightGCN, as comparison models and apply the CGCN framework to improve both models, resulting in the improved models NGCF+ and LightGCN+, respectively.

NGCF [15]: utilizes graph convolution on user-item bipartite graphs to capture collaboration signals by utilizing the higher-order connectivity information of users and items.

LightGCN [14]: removes unnecessary nonlinear activations and feature transformations for graph convolution to obtain better performance for recommendation.

NGCF+: an improved variant of NGCF using the CGCN framework.

LightGCN+: an improved variant of LightGCN using the CGCN framework.

C. HYPER-PARAMETER SETTINGS

During the experiments, we set the embedding size of all models to 64 and used the Adam optimizer with initialization methods consistent with the original method to ensure consistency with NGCF and LightGCN. For the models improved by adding the CGCN framework, we directly obtained and used the optimal parameters of the original method on different datasets from the original text or the authors' open-source code, including the learning rate, regularization coefficient λ, batch size, and the number of layers of the base model. Generally, there is no need to adjust any parameters of the original model, and the parameters that make the original model converge to the optimum can also make the improved model using CGCN converge to the optimum. This means that the cost overhead of applying our improved framework is small.

In the sampling process of cliques, we set the neighbor window of the item-item co-occurrence graph to 15, with a maximum sampling size of 10 cliques and a minimum sampling size of 3 cliques. Additionally, only up to 5 different cliques are kept for each item node to equalize the diversity and efficiency of cliques. This is because a too small neighbor window can lead to sparse regiment nodes and thus lack of expressive power in this process, while a too large neighbor window can lead to additional training costs with little gain. Since the process of finding cliques is an NP-complete problem and very large cliques are very rare, it is usually sufficient to start sampling from cliques of size 10.

The only aspect that requires adjustment is the combination coefficients α. The experimental procedure searches for the optimal parameters among the values of 0.3, 0.5, 0.7, 0.9, 1, 1.1, and 1.3. To ensure a fair comparison, we have made the open source code and associated parameter settings available on Github[1].

D. PERFORMANCE COMPARISON

Table 2 presents the performance comparison between the base model and the improved variants on the three datasets. For the two base models, the results reported in the related

[1]https://github.com/952469119/CGCN

TABLE 3. Comparison with state-of-the-art.

Dataset	Amazon-Books		Yelp2018		Gowalla	
Model	Recall	NDCG	Recall	NDCG	Recall	NDCG
Deepwalk	0.0346	0.0264	0.0476	0.0378	0.1034	0.0740
LINE	0.0410	0.0318	0.0549	0.0446	0.1335	0.1056
Node2Vec	0.0402	0.0309	0.0452	0.0360	0.1019	0.0709
NGCF	0.0344	0.0263	0.0579	0.0477	0.157	0.1327
NIA-GCN	0.0369	0.0287	0.0599	0.0491	0.1359	0.1106
LR-GCCF	0.0341	0.0258	0.0574	0.0349	0.1518	0.1259
LightGCN	0.0411	0.0315	0.0649	0.0530	0.1830	0.1554
DGCF	0.0422	0.0324	0.0654	0.0534	0.1842	0.1561
UltraGCN$_{\text{Base}}$	0.0504	0.0393	0.0667	0.0683	0.1845	0.0393
UltraGCN	**0.0681**	**0.0556**	0.0683	0.0561	0.1862	0.0556
LightGCN+	0.0455	0.0350	**0.0689**	**0.0564**	**0.1888**	**0.1603**

papers were directly followed in the paper since the exact same parameter settings and datasets provided by them were used directly in the experiments. The table shows the best metrics that can be obtained for each variant. It can be seen that NGCF+ vs. NGCF and LightGCN+ vs. LightGCN achieve significant performance improvements on all three datasets, proving the effectiveness of the improved approach.

Fig.7 plots the curves for loss and recall@20 on the Gowalla dataset, and the curves for the other two datasets have a similar trend, which will not be shown repeatedly for space reasons. Observations show that the loss of both improved models is consistently lower than that of the original model throughout the training process. Moreover, the lower loss successfully translates into higher test metrics, suggesting that the improved model has better generalization ability.

E. COMPARISON WITH STATE-OF-THE-ART

Although the generalized framework proposed in this paper focuses on plug-in improvements to GNN-based recommender systems based on user-item bipartite graphs, which are heavily deployed in reality, to further demonstrate the improvement, LightGCN+ is compared with the current state-of-the-art GNN-based recommendation models in this section.

Among them, Deepwalk [11], LINE [26], and Node2Vec [12] are all classical approaches based on graph machine learning, and NGCF [15] and LightGCN [14] are the basic models already mentioned in the previous section. NIA-GCN [35] exploits the user-item bipartite graph by explicitly modeling the relational information between neighboring nodes, thus exploiting the heterogeneous nature. LR-GCCF [27] is a generalized GCN-based model for recommending CF. DGCF [32] considers user-item relationships at a finer granularity of the user's intent and generates de-entangled user and item representations for better recommendation performance. UltraGCN$_{\text{Base}}$ and UltraGCN [18] directly approximates the limit of infinite-layer graph convolutions with a constraint loss, which also can take advantage of multiple negative samples.

Table 3 compares the performance of the baseline and LightGCN+ variants across the three data sets. For the baseline model, the results reported in the relevant literature are directly used in this paper. The table shows the best metrics that can be obtained for each variant.

It can be observed that LightGCN+ achieves the highest performance on both the Gowalla and Yelp2018 datasets, demonstrating the effectiveness of the improved method. However, on the Amazon-book dataset, LightGCN+ still lags significantly behind UltraGCN and UltraGCN$_{\text{Base}}$, despite achieving the greatest performance improvement over LightGCN on all three datasets. This difference in performance may be attributed to the negative sampling approach and complex loss function design utilized by both UltraGCN and UltraGCN$_{\text{Base}}$.

F. EFFICIENCY COMPARISON

As mentioned in the previous section, CGCN's task is to extract the clique structure from the item-item co-occurrence graph, a task efficiently handled by pre-training static. With a reasonable neighbor size and a maximum clique size of around 10, this process typically takes only a few minutes. Because there is no need for recalculation at each epoch during training, this objectively ensures the overall efficiency of the framework.

However, CGCN introduces two new layers: the intra-clique aggregation layer and the multi-level interest fusion layer, on top of the original model, which necessitates additional computational steps. Therefore, in this section, we compare the efficiency of the improved variant with the original model to objectively assess the efficiency of the enhanced CGCN framework.To be more convincing, we compare their training efficiency from two views:

- The total training time and epochs for achieving their best performance.

- Training them with the same time to see what performance they can achieve.

Book with the same machine with one GeForce RTX 3090 GPU for all compared models.We chose the Gowalla dataset for the experiments in this section, and the hyperparameter settings are all the same as in the previous section.

From Table 4, it is evident that the two improved models, LightGCN+ and NGCF+, do require more time compared to the original model. NGCF+ takes 0.4 times longer compared to NGCF in terms of both individual epochs and total time, while LightGCN+ takes approximately 0.2 times longer compared to LightGCN. This finding is consistent with the earlier analysis, indicating that the introduction of two additional layers unavoidably results in an increase in training time, but maintains the same overall time complexity.

TABLE 4. Efficiency comparison from the first view.

Model	Time/EPOCH	EPOCH	Training Time
NGCF	30s	421	210m
NGCF+	42s	437	306m
LightGCN	11.5s	950	182m
LightGCN+	14.1s	919	216m

From Table 5, it is evident that both NGCF+ and LightGCN+ achieve better performance results within the same training duration of 180 minutes. Despite being trained with approximately 110 fewer epochs compared to NGCF, NGCF+ achieves about a 5.4% improvement in performance. Similarly, LightGCN+ achieves a 2.5% performance improvement while being trained with 184 fewer epochs compared to LightGCN.

TABLE 5. Efficiency comparison from the second view.All models were trained using the same 180 minutes.

Model	EPOCH	Recall@20	NDCG@20
NGCF	360	0.1551	0.1311
NGCF+	250	0.1635	0.1354
LightGCN	939	0.1827	0.1553
LightGCN+	755	0.1873	0.1590

Overall, the improved variant incurs a greater time cost compared to the original method in terms of individual epochs and overall training elapsed time. However, the training results achieved over a given unit of time are superior to those of the original method. Specifically, the improved variant is capable of attaining higher performance when the total training elapsed time aligns with the time required for the original model to reach convergence.

G. ABLATION STUDY

To investigate whether CGCN can benefit from multiple message passing layers, we varied the depth of the base model in message passing. In particular, we searched for the number of layers in the range 1,2,3,4. From Table 6, we can observe some interesting results.

TABLE 6. Effect of message passing layer numbers.

Layer	Model	Recall	Model	Recall
1Layer	NGCF	0.1556	NGCF+	0.1672
	LightGCN	0.1755	LightGCN+	0.1810
2Layers	NGCF	0.1547	NGCF+	0.1669
	LightGCN	0.1777	LightGCN+	0.1848
3Layers	NGCF	0.1569	NGCF+	0.1683
	LightGCN	0.1823	LightGCN+	0.1888
4Layers	NGCF	0.1570	NGCF+	0.1680
	LightGCN	0.1830	LightGCN+	0.1882

The performance of the improved models using the CGCN framework varies due to the different performance of the underlying models with different layers. The performance of all models after improvement shows significant improvement. Among them, the performance of NGCF with layers 1-4 is closer, and thus the performance of NGCF+ is also closer; while the performance of LightGCN with layers 1-4 has a larger gap, and thus the performance of LightGCN+ shows different differences. The base model with higher performance always gets better performance with its improved variants. Therefore, we believe that for base models with better performance, only the hyperparameters that enable them to achieve the best performance are needed to achieve the best results in the improved variants using CGCN.

H. HYPERPARAMETER ANALYSIS

Fig.8 illustrates the effect of the combination coefficient α on the Gowalla dataset. In general, LightGCN+ is more influenced by the hyperparameter combination coefficients, with larger optimal value gaps across different datasets. And LightGCN+ showed significant overfitting at higher α. Choosing a lower value on dense datasets tends to yield more desirable results, whereas selecting a higher value on sparser datasets leads to better outcomes. On the other hand,

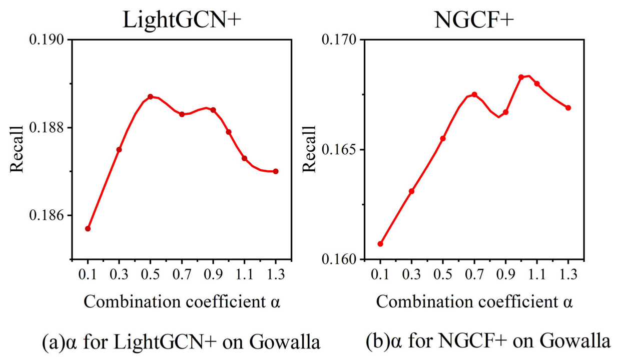


FIGURE 8. Hyperparameter analysis.

NGCF+ is less sensitive to hyperparameter combination coefficients, possibly because NGCF+ incorporates a non-linear transformation unit, allowing it to adaptively learn the weight balance between the two components.

Additionally, experimental variants of the combination coefficients in this study consistently outperformed the original model across a wide range of values. This finding validates that introducing structural information from the item-item co-occurrence graph can enhance the learning of embedded representations of users and items, resulting in improved recommendation performance.

V. CONCLUSION

In the paper, we propose a novel CGCN generalized insertion framework for the recommendation task based on collaborative filtering of graph neural networks. Our approach utilizes the structural information of the graph itself, specifically the relevant features of the clique structure in graph theory. By utilizing the clique information in the item-item co-occurrence graph, our approach ameliorates the noise problem caused by propagating the embeddings in the user-item bipartite graph and better learns the embeddings of both users and items, leading to improved performance in the recommendation task. Notably, many models designed based on user-item bipartite graphs can run under our proposed CGCN framework without hyperparameter tuning.

However, this observation also indicates that the improved model is still constrained by the learning capacity of the original model, and it still faces challenges related to sparse data and cold start issues. Addressing these challenges will be the focus of future research directions.

In addition to graph neural network-based approaches, many other collaborative filtering-based approaches have recently achieved impressive results using the original user-item bipartite graph. Therefore, our next research plan is to explore other related approaches that utilize clique information for the recommendation task learning.

REFERENCES

[1] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, J. Piao, Y. Quan, J. Chang, D. Jin, X. He, and Y. Li, "A survey of graph neural networks for recommender systems: Challenges, methods, and directions," *ACM Trans. Recommender Syst.*, vol. 1, no. 1, pp. 1–51, Mar. 2023.

[2] G. Albora, L. Rossi Mori, and A. Zaccaria, "Sapling similarity: A performing and interpretable memory-based tool for recommendation," *Knowl.-Based Syst.*, vol. 275, Sep. 2023, Art. no. 110659.

[3] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," 2017, *arXiv:1706.02263*.

[4] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1059–1068.

[5] Z. Li, X. Shen, Y. Jiao, X. Pan, P. Zou, X. Meng, C. Yao, and J. Bu, "Hierarchical bipartite graph neural networks: Towards large-scale e-commerce applications," in *Proc. IEEE 36th Int. Conf. Data Eng. (ICDE)*, Apr. 2020, pp. 1677–1688.

[6] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2008, pp. 426–434.

[7] 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*, 2017, pp. 173–182.

[8] S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang, and Y. Gao, "Dual channel hypergraph collaborative filtering," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 2020–2029.

[9] S. H. Park and K. Kim, "Collaborative filtering recommendation system based on improved Jaccard similarity," *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 8, pp. 11319–11336, Aug. 2023.

[10] Y. Wang, W. Ma, M. Zhang, Y. Liu, and S. Ma, "A survey on the fairness of recommender systems," *ACM Trans. Inf. Syst.*, vol. 41, no. 3, pp. 1–43, Jul. 2023.

[11] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 701–710.

[12] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 855–864.

[13] J. Yu, H. Yin, X. Xia, T. Chen, J. Li, and Z. Huang, "Self-supervised learning for recommender systems: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 1, pp. 335–355, Jan. 2024.

[14] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2020, pp. 639–648.

[15] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2019, pp. 165–174.

[16] J. Hu, B. Hooi, S. Qian, Q. Fang, and C. Xu, "MGDCF: Distance learning via Markov graph diffusion for neural collaborative filtering," *IEEE Trans. Knowl. Data Eng.*, early access, Jan. 9, 2024, doi: 10.1109/TKDE.2023.3348537.

[17] J. Sun, Y. Zhang, M. Chen, M. Coates, H. Guo, R. Tang, and X. He, "Multi-graph convolution collaborative filtering," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2020, pp. 1306–1311.

[18] K. Mao, J. Zhu, X. Xiao, B. Lu, Z. Wang, and X. He, "UltraGCN: Ultra simplification of graph convolutional networks for recommendation," in *Proc. 30th ACM Int. Conf. Inf. Knowledge Manag.*, Oct. 2021, pp. 1253–1262.

[19] F. Li, Z. Chen, P. Wang, Y. Ren, D. Zhang, and X. Zhu, "Graph intention network for click-through rate prediction in sponsored search," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieva*, Sep. 2021, pp. 961–964.

[20] X. Li, L. Sun, M. Ling, and Y. Peng, "A survey of graph neural network based recommendation in social networks," *Neurocomputing*, vol. 549, Sep. 2023, Art. no. 126441.

[21] A. Kammoun, R. Slama, H. Tabia, T. Ouni, and M. Abid, "Generative adversarial networks for face generation: A survey," *ACM Comput. Surv.*, vol. 55, no. 5, pp. 1–37, May 2023.

[22] M. E. J. Newman, D. J. Watts, and S. H. Strogatz, "Random graph models of social networks," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 1, pp. 2566–2572, 2002.

[23] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks," in *Proc. 19th Int. Conf. World Wide Web*, Apr. 2010, pp. 641–650.

[24] A. Wimmer and K. Lewis, "Beyond and below racial homophily: ERG models of a friendship network documented on Facebook," *Amer. J. Sociol.*, vol. 116, no. 2, pp. 583–642, Sep. 2010.

[25] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, 1994.

[26] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, May 2015, pp. 1067–1077.

[27] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 1, pp. 27–34.

[28] L. Yang, S. Wang, Y. Tao, J. Sun, X. Liu, P. S. Yu, and T. Wang, "DGRec: Graph neural network for recommendation with diversified embedding generation," in *Proc. 16th ACM Int. Conf. Web Search Data Mining*, Feb. 2023, pp. 661–669.

[29] Z. Sheng, T. Zhang, Y. Zhang, and S. Gao, "Enhanced graph neural network for session-based recommendation," *Exp. Syst. Appl.*, vol. 213, Mar. 2023, Art. no. 118887.

[30] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, 2018, pp. 974–983.

[31] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He, "Bias and debias in recommender system: A survey and future directions," *ACM Trans. Inf. Syst.*, vol. 41, no. 3, pp. 1–39, Jul. 2023.

[32] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T.-S. Chua, "Disentangled graph collaborative filtering," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 1001–1010.

[33] W. Jiang, Y. Jiao, Q. Wang, C. Liang, L. Guo, Y. Zhang, Z. Sun, Y. Xiong, and Y. Zhu, "Triangle graph interest network for click-through rate prediction," in *Proc. 15th ACM Int. Conf. Web Search Data Mining*, 2022, pp. 401–409.

[34] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," 2012, *arXiv:1205.2618*.

[35] J. Sun, Y. Zhang, W. Guo, H. Guo, R. Tang, X. He, C. Ma, and M. Coates, "Neighbor interaction aware graph convolution networks for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2020, pp. 1289–1298.

ZHENYE PAN was born in 1997. He is currently pursuing the master's degree with the School of Computer Science and Technology, Zhejiang University of Science and Technology. His research interests include graph neural networks and recommendation systems.

YAHONG CHEN was born in 1977. She is currently pursuing the Doctor of Science degree with Zhejiang Sci-Tech University. She is a Professor and a part-time Master's Tutor with Zhejiang Sci-Tech University. Her research interests include matrix theory, spectral extremum theory, complex networks, and related problems. She is a member of Zhejiang University Mathematics Teaching Steering Committee. She is the Executive Director of Zhejiang Higher Mathematics Teaching Research Association and the Director of Zhejiang Applied Mathematics Society. She is the Vice President of Lishui Women Science and Technology Workers Association. She is an American Mathematics Commentator.