Social recommendation based on social network has achieved great success in improving the performance of the recommendation system. Since social network (user-user relations) and user-item interactions are both naturally represented as graph-structured data, Graph Neural Networks (GNNs) have thus been widely applied for social recommendation. Despite the superior performance of existing GNNs-based methods, there are still several severe limitations: (i) Few existing GNNs-based methods have considered a single heterogeneous global graph which takes into account user-user relations, user-item interactions, and item-item similarities simultaneously. That may lead to a lack of complex semantic information and rich topological information when encoding users and items based on GNN. (ii) Furthermore, previous methods tend to overlook the reliability of the original user-user relations which may be noisy and incomplete. (iii) More importantly, the item-item connections established by a few existing methods merely using initial rating attributes or extra attributes (such as category) of items, may be inaccurate or sub-optimal with respect to social recommendation. In order to address these issues, we propose an end-to-end heterogeneous global graph learning framework, namely Graph Learning Augmented Heterogeneous Graph Neural Network (GL-HGNN) for social recommendation. GL-HGNN aims to learn a heterogeneous global graph that makes full use of user-user relations, user-item interactions and item-item similarities in a unified perspective. To this end, we design a Graph Learner (GL) method to learn and optimize user-user and item-item connections separately. Moreover, we employ a Heterogeneous Graph Neural Network (HGNN) to capture the high-order complex semantic relations from our learned heterogeneous global graph. To scale up the computation of graph learning, we further present the Anchor-based Graph Learner (AGL) to reduce computational complexity. Extensive experiments on four real-world datasets demonstrate the effectiveness of our model.

CCS Concepts: • Information systems → Recommender systems;
1 INTRODUCTION

In the era of information explosion, how to help users get useful information from the flood of information is playing a very important role [23]. Recommendation system is an effective way to alleviate information overload and is widely used in various fields, such as e-commerce (Taobao, Amazon) and social media (Instagram, Facebook) sites. However, when the user-item interaction is too sparse, the performance of recommendation systems tends to be poor. Therefore, researchers put forward social recommendation [41, 42] to alleviate the sparsity problem by introducing user-user social relations.

Recent years have witnessed the rapid development of social recommendation, which leverages social network as side information to effectively alleviate the problem of data sparsity [31, 53]. Conceptually, users’ preferences are often largely influenced by people around them [4, 10], including parents, friends, classmates, and so on. A social recommendation system based on users’ social relationships usually significantly improves the quality of recommendations.

Recently, there is a surge of interests in graph neural networks (GNNs) [14, 22, 44, 52], which have been proven to effectively learn node representations from graph-structured data. Since social network (user-user relations) and user-item interactions are both naturally represented as graph-structured data [8, 57], GNNs have thus been widely employed to learn the representations of users and items, which has been shown to improve the performance of the social recommendation [55, 68]. Furthermore, in order to enrich the types of the potential graphs and extract richer side information, a few existing works have considered the construction of extra item-item graph structure [9, 18, 58].

Despite the promising results current methods have achieved, there are still several severe limitations in existing approaches. First, few existing GNNs-based methods have considered a single heterogeneous global graph which takes into account user-user relations, user-item interactions and item-item similarities simultaneously. As a result, these methods may fail to capture high-order cross-semantic information and limit the delivery of messages. Second, previous methods tend to overlook the reliability of the original user-user social graph that may be noisy and incomplete [65]. The original connections often only record the social relationships between users but rather reflect the similarity of users’ preferences. For example: Bob is Ketty’s husband, Jim is Ketty’s colleague, Bob and Jim both like sports, while Ketty likes reading. However, in the user-user connections, Bob and Jim are not directly connected, while they are connected to Ketty respectively. We can learn from this example that there may be conflicts in interests between nearby neighbors, while distinct neighbors could have similar preferences. Such a topology will make the users’ preferences extracted from user-user connections deviate from the real situation, which may lead to the sub-optimal performance of downstream recommendation task. Besides, previous methods do not fully exploit the relationships between items. Though a few existing methods attempt to construct the item-item graph, they only utilize the items’ initial rating attributes or extra attributes (such as category) in an ad-hoc fashion, which barely reflects the optimized item-item graph structure with respect to downstream social recommendation.
To address these issues, we propose an end-to-end heterogeneous global graph learning framework, namely Deep **Graph Learning Augmented Heterogeneous Graph Neural Network (GL-HGNN)** for social recommendation. Our GL-HGNN aims to learn a heterogeneous global graph that makes full use of user-user relations, user-item interactions and item-item similarities in a unified perspective.

In order to obtain and optimize heterogeneous global graph structure, we present a Global Graph Learning module. We first establish item-item subgraph by calculating the similarity of the rating vectors. Then, we use graph structure learning to optimize the global graph. We propose the **Graph Learner (GL)** method to extract richer implicit relationships and filter out the explicit noisy edges in user-user relation subgraph and item-item similarity subgraph. Specifically, our GL method can learn the implicit connections between nodes by measuring the embeddings similarity between target nodes in each mini-batch and all nodes. We then combine the learned implicit graph with the initial graph to obtain the refined heterogeneous global with respect to downstream task. To capture the high-order complex semantic relations from different types of edges in the heterogeneous global graph, we also present a **Heterogeneous Graph Neural Network (HGNN)**, to model the refined heterogeneous global graph. Technically, it is crucial to scale up the computation of user-user subgraph and item-item subgraph learning, especially when the number of users or items is very large. To reduce computational complexity, we further utilize anchor-based approximation technique [6] to design a scalable Graph Learner, namely **Anchor-based Graph Learner (AGL)**, which selects the anchor nodes instead of all nodes to calculate the similarity to the target nodes. In addition, we design a joint learning method and a hybrid loss which considers both graph structure learner loss and rating loss. Through multiple epochs of optimization, we can get more refined heterogeneous global graph structures with respect to social recommendation, and more reliable representations of users and items.

To summarize, we highlight our main contributions as follows:

- We construct a heterogeneous global graph with multiple semantic relations for social recommendation. We propose a novel framework named GL-HGNN to learn the heterogeneous global graph of different relationships in a unified perspective, which can capture the complex semantic relations and rich topological information.
- We propose the Global Graph Learning module to construct item-item connections and optimize both user-user and item-item subgraph structures, so as to obtain the refined global graph with respect to the downstream social recommendation. In addition, we design an Anchor-based Graph Learner (AGL) method to scale up the proposed method, which can significantly reduce computational complexity.
- The experimental results on four real-world datasets demonstrate the effectiveness of our proposed model over state-of-the-art methods. Besides, we also conduct experiments to verify that our scalable AGL module can reduce the computational costs.

2 RELATED WORKS

2.1 Graph Neural Network

Graph Neural Networks (GNNs) [25, 56, 59] aim to utilize deep neural network technology to model graph-structured data. Graph convolutional network (GCN) [22] is a classic method, which maps the graph signal from spatial domain to frequency domain by **Graph Fourier Transform (GFT)**. GCN generates target node embeddings by aggregating the feature of neighbor nodes. However, the disadvantage of GCN is that it requires complete graph data for each layer of the network architecture. To address the limitition, **GraphSage** [14] is proposed to model the graph structure inductively. GraphSage samples neighbor nodes of target nodes, and aggregates node features...
with the way of mini-batch training. Different from GCN, GraphSage aggregates in a number of ways, such as mean aggregator, LSTM [17] aggregator and pooling aggregator. In order to learn the heterogeneous strength of edges between different nodes, GAT [44] aggregates the features of neighbors with attention mechanism, which can adaptively learn different weights of different neighbors. However, all of these methods model homogeneous graphs and do not perform well on heterogeneous graphs. Plenty of heterogeneous graph neural networks [11, 15, 20, 36, 45, 52] have been proposed to solve the problem. Due to the superiority of GNN in modeling graph-structured data, GNN is widely used in a variety of applications such as social network, chemistry and recommendation system.

2.2 Graph-based Recommendation

Graph neural network (GNN) is a method of projecting nodes in a generated graph into a low-dimensional vector space, based on the idea that nodes are naturally characterized by their own features and neighbors [7, 35, 61]. Based on this theory, many works use GNN to model user-item bipartite graphs in recommendation scenarios [60]. Several works [50, 64] utilize graph convolution network to extract user and item collaborative feature. LightGCN [16] removes feature variation and nonlinear activation on the basis of GCN, which not only simplifies the model, but also extracts more collaboration information. In addition, plenty of models [43, 54, 60] also use graph attention network (GAT) to aggregate the heterogeneous strength of collaborative information, so that the target node representation is more suitable for downstream recommendation strength. Moreover, many other models [3, 46, 49, 51] introduce Knowledge graph into user-item bipartite graph to extract user and item features at a finer granularity with GNNs.

However, when the number of user-item interactions is sparse, the GNN-based recommendation system does not show superior performance. Social recommendation, by introducing user-user social information, can effectively alleviate this limitation.

2.3 Graph Structure Learning

As GNNs rely heavily on the good quality of the original graph, graph structure learning method was proposed to alleviate this limitation. LDS [12] proposed to model each edge inside the adjacency matrix. SLC [67] proposed to extend the traditional convolutional neural network (CNN) to graph domains and learn the graph structure for traffic forecasting. IDGL [6] jointly and iteratively learned graph structure and graph embedding based on node features. Jin et al. [21] proposed a general framework Pro-GNN, which can jointly learn a structural graph and a robust graph neural network model from the perturbed graph. HGSL [69] generated three kinds of graph structures to fuse an optimal heterogeneous graph. PTDNet [29] pruned task-irrelevant edges by penalizing the number of edges in the sparsified graph with parameterized networks. TRI [5] proposed the topological relational inference to integrate higher-order graph information to GNNs and for systematically learn a local graph structure. GEN [47] implements the estimation of graph to maximize the posterior probability based on Bayesian inference. VIB-GSL [38] utilizes information theory to advance the Information Bottleneck (IB) principle for graph structure learning, providing a more elegant and universal framework for mining underlying task-relevant relations. CoGSL [27] extracted two basic views as inputs and used an adaptive technique to obtain the final view by fusing estimated views refinedly reestimated by a view estimator.

However, most of these models are applied to node classification or graph-level prediction tasks. To fill the gap, we propose to adopt the graph structure learning to improve the quality of the heterogeneous global graph for social recommendation.
2.4 Social Recommendation

With the popularity of social platforms, social recommendation, based on the assumption that users’ preferences and choices are always influenced by people related to them [32, 33], has become one of the hottest areas in recommendation research. Early research mainly employed matrix factorization-based methods for recommendation, such as SoRec [30], TrustFM [63] and TrustSVD [13]. Recently, deep learning-based methods have become the most successful methods in recommendation research. Plenty of recent works [10, 34, 37] have applied deep learning to social recommendation tasks and achieved promising performance.

In recent years, a lot of works [2, 8, 55] transform user-user relations and user-item interactions to graph-structured data, and employ the graph neural network (GNN) to learn better user and item representations. MPSR [26] extracts hierarchical user preferences and assign friends’ influence with different trust levels from different perspectives. DISGCN [24] proposes to utilize GCN to disentangle both social homophily-aware user interests and social influence. Yu, et al. [66] propose a multi-channel hypergraph convolutional network to enhance social recommendation by leveraging high-order user relations. In addition, to capture connections among items and enhance the performance of social recommendation, several efforts adopted the item attributes to construct item-item graph. For example, GraphRec+ [9] and DANSER [58] leveraged the item’s collaborative information to build item-item graph, while KCGN [18] utilized inter-dependent knowledge of items to construct graphs. SMIN [28] designs a metapath-guided heterogeneous graph neural network, to investigate the potential of jointly incorporating social- and knowledge-aware relational structures.

However, few of these methods extract high-order cross-semantic information by modeling a joint heterogeneous global graph in a unified perspective, which includes three kinds of semantic relations: user-user, user-item and item-item.

3 DEFINITION AND PRELIMINARY

In this paper, we define $U = \{u_1, u_2, \ldots, u_N\}$ and $V = \{v_1, v_2, \ldots, v_M\}$ as the sets of users and items, separately. The user-user relations can be defined as $G_{uu} = \{U, E_u\}$, in which $E_u$ is the set of edges, and $(u_i, u_n, r_u)$ in $E_u$ represents $u_i$ is related to $u_n$. And the user-item interactions can be represented as the user-item graph with $K$ kinds of edges $G_{uv} = \{U, V, E_v\}$. The edge in $E_v$ is defined as $(u_i, v_j, r_v)$, which indicates that the user $u_i$ rates the item $v_j$ as $k$. Let $U(u_i)$ denote the set of users related to user $u_i$. In addition, $V^K(u_i)$ is defined as the set of items that the user $u_i$ rates $k$ to, while $U^K(v_j)$ as the set of users who give a rating $k$ to $v_j$. The mathematical notations of this paper are summarized in Table 1.

**Problem Formulation.** Let $p_i, q_j \in \mathbb{R}^D$ denote initial embeddings of the target user $u_i$ and item $v_j$. Given user-user relations and user-item interactions, the task is to predict the explicit score $\hat{r}_{ij}$ that user $u_i$ will rate item $v_j$.

4 METHODOLOGIES

4.1 Overview

Figure 1 provides the overall architecture of our model. We aim to construct a heterogeneous global graph, and extract cross-semantic relations and rich topological information from it. We first build item-item subgraph $G_{vv} = \{V, E_v\}$ by similarity between items. The edge $(v_j, v_m, r_v)$ in $E_v$ means items $v_j$ and $v_m$ are similar. We also define $V(v_j)$ to denote the set of items similar to item $v_j$. We combine these three graphs $\{G_{uu}, G_{uv}, G_{vv}\}$ into a heterogeneous global graph $G$, which contains two kinds of nodes, three kinds of semantic paths and $K + 2$ kinds of edges, i.e., user-user relation edge, $K$ kinds of rating edges, and item-item similarity edge. In order to get a
Table 1. Notation

| Symbols | Definitions and Descriptions |
|---------|-----------------------------|
| $U$     | the set of users             |
| $V$     | the set of items             |
| $R$     | the rating matrix            |
| $p_i$   | the initial embedding of the user $u_i$ |
| $q_j$   | the initial embedding of the item $v_j$ |
| $U^k(u_i)$ | the set of users related to user $u_i$ to rate $k$ |
| $V^k(v_j)$ | the set of items that the user $u_i$ rates $k$ to |
| $r_{ij}$ | the rating value that user $u_i$ will rate item $v_j$ |
| $A_x$   | the adjacency matrix         |
| $G_x$   | the graph                    |
| $p^{(t)}_i, q^{(t)}_j$ | the representations of user $u_i$ and the item $v_j$ after the propagation of $t$-th HGNN layer |
| $K_I$   | the hyper-parameter to create initial item-item edges |
| $\lambda_w$ | the hyper-parameter which is used to combine the learned matrix with the initial matrix |
| $L$     | the truncation length to create refined graph |
| $W, b$  | the trainable weight and bias in neural network |

Fig. 1. The overview of our model GL-HGNN. We first establish the item-item subgraph structure. We employ the Graph Structure Learner to update and optimize the graph structure. We utilize HGNN to model the global graph to extract complex cross-semantic information. The output embeddings are sent to the predictor for prediction. We design a hybrid loss including Graph Learner loss and rating loss for training.

better graph structure with respect to the downstream task, capture implicit connections and filter out possible noise, we design the Graph Learner to optimize user-user (u2u) and item-item (i2i) connections. Moreover, the refined global graph is passed as input to a heterogeneous graph neural network to distill high-order complex semantic information. We employ a rating predictor to predict the score that target user will rate the candidate item. We design a hybrid loss to train our model.
4.2 Global Graph Learning

Figure 2 shows the architecture of Global Graph Learning module, which constructs the heterogeneous global graph and optimizes the graph structures. It should be noted that there is usually no connection information between items in the raw data, but item-item connections can enrich the graph structure and improve the receptive field, which allows us to extract more information of both users and items. Therefore, we need to construct the item-item edges first. Then, we employ Graph Learner (GL) to optimize the user-user and item-item subgraph by adding or removing the edges with the method of calculating the similarity of node embeddings [6]. We will introduce the details below.

**Item-item Connections Construction.** We utilize the rating matrix $R \in \mathbb{R}^{N \times M}$ to calculate the cosine similarity between items following previous work [9]. For the rating matrix $R$, we take the $j$-th column vector $e_j$ as the item $v_j$ vector. The similarity calculation formula is denoted as:

$$\text{score}(v_j, v_m) = \frac{e_j \cdot e_m}{\|e_j\|\|e_m\|}.$$

For each item, we choose the most similar $K_I$ items to create the edges. In this way, we construct the item-item connections.

**Graph Learner.** Due to the noise or lack of possible information in the original graph structure, we propose to adapt Graph Learner to optimize the input u2u and i2i subgraphs’ topologies.

For u2u Graph Learner, the input is the subgraph $G_{uu}$ with node set $\{u_1, u_2, \ldots, u_N\}$ and embedding set $\{p_1, p_2, \ldots, p_N\}$. For the target node $u_i$, we apply the multi-perspective method [6] to calculate the similarity between $u_i$ and all nodes, where each perspective calculates one part of the semantics captured in the vectors automatically by trainable parameters. The calculation is as follows:

$$\text{sim}(u_i, u_n) = \frac{1}{F} \sum_{f=1}^{F} \text{sim}^f(u_i, u_n), n = 1, 2, \ldots, N$$

where $F$ is the number of perspectives. For each perspective, we can choose one from three methods, which are called weighted cosine, attention, and add attention:

$$\text{sim}^f(u_i, u_n) = \text{cosine} \left( W^c_f p_i, W^c_f p_n \right)$$

$$\text{sim}^f(u_i, u_n) = \left( W^a_f p_i \right)^T \left( W^a_f p_n \right)$$

Fig. 2. The structure of proposed Global Graph Learning. We construct the i2i graph first. We calculate the similarity of nodes to get the implicit graph. We combine the implicit graph with initial graph.
\[ \text{sim}^{\ell}(u_i, u_n) = \sigma \left( w_f^T p_i + w_f^T p_n \right) \] (4)

Equation (2) is the principle of \textit{weighted cosine}. Equation (3) shows the calculation method of \textit{attention}. Equation (4) presents the principle of the \textit{add attention}. \(W_f^u, W_f^v\) and \(w_f^T\) are the trainable parameters and \(\sigma\) is the ReLU function. During the experiment, we mainly use the \textit{weighted cosine} method, and the other two methods will be compared in the ablation study.

For all the target user nodes in one batch, the initial adjacency matrix with all nodes is \(A_u \in \mathbb{R}^{B \times N}\), where \(B\) is the number of target user nodes in the current batch. And we can obtain a new learned implicit adjacency matrix \(A'_u\) with similarity calculation. Though the initial graph may be noisy or missing information, it still contains rich valuable topological information. Therefore, we employ a weight value \(\lambda_w\) to combine the implicit matrix with the initial matrix:

\[ \tilde{A}_u = \lambda_w A'_u + (1 - \lambda_w) A_u \] (5)

Each element in the refined matrix \(\tilde{A}_u\) represents the similarity of two nodes. In order to prevent information redundancy caused by too many edges, we set a truncation length \(L\). For each target node, we truncate the first \(L\) nodes with the highest similarity to establish new connections, and the remaining nodes are not connected to the target node. In this way, we can get the refined subgraph \(G'_{uu}\).

Similarly, for the input i2i subgraph \(G_{vv}\), we can apply the same method to get the refined subgraph \(G'_{vv}\).

\textbf{Anchor-based Graph Learner.} In the real world, the number of nodes is often very large. For target nodes, if we calculate the similarity of all the nodes to them, the costs of computation are high. Inspired by [6], we proposed a scalable Anchor-based Graph Learner (AGL). Next, we take the item-item subgraph as an example. For target item nodes, we randomly select \(H_l(H_u \ll M)\) nodes as the anchor nodes set \(\{v_{m_1}, v_{m_2}, \ldots, v_{m_{H_u}}\}\). We can get the initial adjacency matrix between target nodes and anchor nodes \(A_{v,\text{anchor}} \in \mathbb{R}^{B \times H_u}\) from the initial connections. We calculate the target-anchor similarity matrix \(A'_{v,\text{anchor}}\) as Equation (1). Then we use the weight value \(\lambda_w\) and the truncation length \(L\) to calculate the refined item-item subgraph \(G'_{vv}\). Similarly, we can randomly select \(H_u(H_u \ll N)\) user nodes as the anchor nodes set, and employ AGL to get the refined user-user subgraph \(G'_{uu}\).

\subsection*{4.3 Heterogeneous Graph Neural Network}

In this subsection, we discuss how to extract user and item latent features in a unified perspective, based on the refined global graph \(G' = \{G'_{uu}, G_{uv}, G'_{vv}\}\) learned by Global Graph Learning. The global graph includes three kinds of semantic paths: user-user relations, user-item interactions and item-item similarities. Inspired by RGCN [36], we employ a Heterogeneous Graph Neural Network (HGNN) to extract high-order information and fuse different semantic information.

We employ \(T\)-layer HGNN to model our refined heterogeneous global graph to distill cross-semantic information. For the target user \(u_i\) and target item \(v_j\), the initial input embeddings of the first layer are \(p_i^{(0)} = p_i\) and \(q_j^{(0)} = q_j\). Let \(p_i^{(t)}\) and \(q_j^{(t)}\) denote the representations of user \(u_i\) and the item \(v_j\) after the propagation of \(t\)-th layer. We next introduce the user node aggregation and item node aggregation in each layer of HGNN.

\textbf{User node aggregation.} Generally, for each user node in the refined heterogeneous global graph, there exits one type of edges \(r_u\) connecting the user neighbors and \(K\) types of edges \(r_k\), \((k \in \{1, 2, \ldots, K\})\) connecting the item neighbors. For the user-user social semantic connections, we
aggregate the features of user neighbors as follows:

$$p_{i,u}^{(t+1)} = b_{u}^{(t+1)} + \sum_{u_n \in U(u_i)} \frac{1}{c_{l,n}} W_{u}^{(t+1)} p_{n}^{(t)}$$

(6)

c_{l,n} = \sqrt{|U(u_i)| |U(u_n)|}

(7)

where $W_{u}^{(t+1)}$ is a trainable transformation matrix, $b_{u}^{(t+1)}$ is the bias vector, and $c_{l,n}$ is the normalization coefficient.

Similarly, we perform user $u_i$ node aggregation based on $K$ types of user-item rating connections. Specifically, for each type of edges $r_k$, we also aggregate neighbor items under the same rating level as follows:

$$p_{i,k}^{(t+1)} = b_{k}^{(t+1)} + \sum_{v_m \in V^k(u_i)} \frac{1}{c_{l,m}} W_{k}^{(t+1)} q_{m}^{(t)}$$

(8)

c_{l,m} = \sqrt{|V^k(u_i)| |V^k(v_m)|}, k \in \{1, 2, \ldots, K\}

(9)

For user $u_i$, we accumulate all messages propagated by different $K + 1$ types of edges $[p_{i,u}^{(t+1)}, p_{i,1}^{(t+1)}, \ldots, p_{i,K}^{(t+1)}]$. Then, we aggregate the information of these $K + 1$ embeddings:

$$p_i^{(t+1)} = \sigma \left( \frac{1}{K+1} \left( p_{i,u}^{(t+1)} + \sum_{k=1}^{K} p_{i,k}^{(t+1)} \right) \right)$$

(10)

$\sigma$ is the ReLU function, $p_i^{(t+1)}$ is the output embedding of user $u_i$ in $t + 1$-th HGNN layer. It is worth noting that, for the current layer, we integrate two kinds of semantic paths (user-user, user-item) information into the user’s features, while the item features already contain the item-item semantic information after $t$ layers aggregation. Therefore, the user’s features can also fuse item-item semantic information by the multi-layer HGNN.

**Item node aggregation.** The target item $v_j$ also involves in two semantic paths: item-item similarity and user-item interactions including $K$ types of edges. Similarly, for the $t + 1$-th layer, we propagate different messages from $K + 1$ types of edges and obtain $K + 1$ embeddings $[q_{j,u}^{(t+1)}, q_{j,1}^{(t+1)}, \ldots, q_{j,K}^{(t+1)}]$ of $v_j$. Then we aggregate these embeddings into the output embedding $q_j^{(t+1)}$:

$$q_j^{(t+1)} = \sigma \left( \frac{1}{K+1} \left( q_{j,u}^{(t+1)} + \sum_{k=1}^{K} q_{j,k}^{(t+1)} \right) \right)$$

(11)

After $T$ layers of HGNN, we can extract high-order and cross-semantic information from the refined heterogeneous global graph, which enables us to distill more latent features of users and items.

### 4.4 Rating Predictor

The initial embeddings and output of each HGNN layer constitute the user $u_i$ embedding lists $[p_{i,0}^{(0)}, p_{i,1}^{(0)}, \ldots, p_{i,T}^{(T)}]$ and item $v_j$ embedding lists $[q_{j,0}^{(0)}, q_{j,1}^{(0)}, \ldots, q_{j,T}^{(T)}]$. We design the shared attention mechanism to get the final user and item latent embeddings. For the user $u_i$, the final embedding is defined as follows:

$$p_i^* = \sum_{t=0}^{T} \alpha_{t,u} p_{i}^{(t)}$$

(12)

$$\alpha_{t,u} = \frac{\alpha_{t,u}'}{\sum_{t'=0}^{T} \alpha_{t',u}'}$$

(13)
\[ \alpha'_{t,u} = s^T \sigma \left( W_s p^{(t)}_i + b_s \right) \]  

(14)

where \( W_s, s \) and \( b_s \) are the shared trainable parameters, \( \sigma \) is the ReLU function. And the final embedding \( q^*_j \) of item \( v_j \) can be calculated in the same way. In this paper, we focus on the rating prediction task in social recommendation. Dot product and multi layer perceptron (MLP) are both common predictors. In fact, dot product is more often used for rank tasks, while for rating prediction task, most existing methods (e.g., GraphRec, DANSER) adopt the MLP-based predictor. So, we design the predictor based on multi layer perceptron (MLP):

\[ \hat{r}_{ij} = \text{MLP}([p^*_i, q^*_j]) \]  

(15)

where \([,]\) is the concatenation operation.

4.5 Model Training

To better train our model, we design a special loss function, which contains two aspects of loss: (i) Graph Learner loss, and (ii) rating loss.

Graph Learner Loss. In our work, the updated graph structure plays an important role in rating prediction. In order to obtain the better graph topology with respect to the social recommendation task, we design the Graph Learner (GL) loss through graph regularization \([1, 6]\). For the u2u GL, we can get the refined adjacency matrix \( \hat{A}_u \in \mathbb{R}^{B \times N} \). Generally, graph regularization is often applicable for symmetric adjacency matrix. Since \( \hat{A}_u \) is not symmetric, we first transform it to be symmetric as follows:

\[ \hat{A}_u = \tilde{A}_u \Delta^{-1} \tilde{A}_u^T \]  

(16)

where \( \Delta \in \mathbb{R}^{N \times N} \) (\( \Delta_{ii} = \sum_{k=1}^{N} A_{u,ki} \)) is a diagonal matrix. As we all know, that values change smoothly among adjacent nodes is a widely applied assumption. Therefore, we utilize \( \hat{A}_u \) and initial user feature matrix \( P \) to design the smoothness loss as follows:

\[ L(\hat{A}_u, P) = \frac{1}{2B^2} \sum_{i,n} \hat{A}_{u,in} \|p_i - p_n\|^2 = \frac{1}{B^2} \text{tr}(P^T LP) \]  

(17)

where \( \text{tr}(\cdot) \) indicates the trace of a matrix, \( L = D_u - \hat{A}_u \) is the graph Laplacian, and \( D_u = \sum_{n} \hat{A}_{u,in} \) denotes the degree matrix. However, only minimizing the smoothness loss will cause over-smoothing, so we impose constraints \([6]\) to control smoothness as follows:

\[ C(\hat{A}_u) = -\frac{\beta_1}{B} 1^T \log(\hat{A}_u 1) + \frac{\beta_2}{B^2} \|\hat{A}_u\|_2 \]  

(18)

where \( 1 \) indicates the vector in which elements are 1, and \( \|\hat{A}_u\|_2 \) indicates the Euclidean norm of \( \hat{A}_u \). We then define the overall Graph Learner loss of u2u GL as the sum of the previously defined losses:

\[ L^u_G = \beta L(\hat{A}_u, P) + C(\hat{A}_u) \]  

(19)

\( \beta \) is a non-negative hyper-parameters.

While for u2u AGL, we can convert the refined adjacency matrix \( \hat{A}_{u,\text{anchor}} \in \mathbb{R}^{B \times H_u} \) to the symmetric matrix \( \hat{A}_{u,\text{anchor}} \) as Equation (16). And we can rewrite Equation (19) to define the Anchor-based Graph Learner loss:

\[ L^u_G = \beta L(\hat{A}_{u,\text{anchor}}, P) + C(\hat{A}_{u,\text{anchor}}) \]  

(20)

We also calculate i2i GL or AGL loss \( L^i_G \) by the same method.
Rating Loss. For the task of rating prediction, we adopt mean square error (MSE) loss function as:

\[ L_r = \frac{1}{B} \sum_{i,j} \| \hat{r}_{ij} - r_{ij} \|^2 \]  \hspace{1cm} (21)

where \( r_{ij} \) is the ground-truth value. For our model, we apply a hybrid loss to jointly learn the parameters:

\[ L = L_r + \gamma_u L^u_G + \gamma_v L^v_G + \eta \Omega(\Theta) \]  \hspace{1cm} (22)

\( \gamma_u, \gamma_v \) and \( \lambda \) are non-negative hyper-parameters. \( \Theta \) is the trainable parameters, \( \Omega(\cdot) \) denotes the L2 regularization. Through multiple epochs of optimization, we can iteratively learn an optimized global graph structure with respect to the social recommendation as well as reliable user and item features.

4.6 Model Complexity Analysis

GL-HGNN. As for GL-HGNN, the computational cost of the Graph Learner is \( O(E(N + M)D) \) for \( N \) user nodes, \( M \) item nodes and \( E \) missing user-item rates to be predicted. The computational cost of HGNN is \( O(\tau X(M + N)D) \), where \( \tau \) denotes the number of layers and \( X \) indicates the average neighbors of each node. The rating task costs \( O(EdD) \) where \( d \) is the hidden size, while the computational complexity of the hybrid loss is \( O(E(N + M)D) \). The overall cost is about \( O((E + E')(N + M)D + EdD) \). Due to \( E = a_1M = a_2N = \frac{a_2}{2}M + \frac{a_1}{2}N \), where \( a_1 \ll M \) and \( a_2 \ll N \) denote the average frequency of each user and each item, respectively, we assume that \( O(E) = O(M + N) \). If we assume that \( d \ll N + M \), the overall time complexity is \( O((N + M)^2D) \).

AGL-HGNN. As for AGL-HGNN, the computational cost of the Anchor-based Graph Learner is \( O(E(H_u + H_v)D) \), while computing node embeddings by HGNN costs \( O(TX'(M + N)D) \), where \( X' \) indicates the average neighbors of each node. The rating task also costs \( O(EdD) \), and computing the hybrid loss costs \( O(E(H_u + H_v)D) \). As \( H_u, H_v, d \ll (N + M) \), the overall time complexity of AGL-HGNN is \( O(TX'(N + M)D) \), which is linear to the number of user and item. Therefore, AGL-HGNN can significantly reduce the computational complexity.

5 EXPERIMENTS

In this section, we will detail the settings of our experiment and present the experimental results. To fully demonstrate the superiority of our model, we conduct experiments to verify the following four research questions (RQ):

- **(RQ1)**: Compared with the state-of-the-art models, does our model achieve better performance?
- **(RQ2)**: What are the impacts of key components on model performance?
- **(RQ3)**: How does the setting of hyper-parameters (such as the truncation length in Graph Learner) affect our model?
- **(RQ4)**: How can Global Graph Learning module improve the performance of our model?

5.1 Experiment Setup

5.1.1 Datasets. We conduct experiments on several public social recommendation benchmark datasets Ciao\(^1\) [40], Epinions\(^2\) [39] and Flixster\(^3\) [19], which all contain rating information and social networks. The detailed statistics of the dataset are given in Table 2.

\(^1\)http://www.ciao.co.uk
\(^2\)http://www.epinions.com
\(^3\)https://www.flixster.com
Table 2. Statistics of Datasets

| Dataset   | Ciao-5       | Ciao-28       | Epinions     | Flixster     |
|-----------|--------------|---------------|--------------|--------------|
| # of Users| 2,248        | 10,994        | 22,164       | 147,612      |
| # of Items| 16,861       | 112,802       | 296,277      | 48,794       |
| # of Ratings| 36,065      | 304,493       | 922,267      | 8,196,067    |
| # of Ratings Density| 0.095%     | 0.025%        | 0.014%       | 0.114%       |
| Rating Range| [1, 5]      | [1, 5]        | [1, 5]       | [0, 5, 5]    |
| # of Social Links| 52,907     | 131,427       | 362,433      | 2,442,886    |
| # of Social Links Density| 1.047%     | 0.108%        | 0.073%       | 0.011%       |

- **Ciao**: *Ciao* is derived from a popular social networking e-commerce platform. We process two available versions of the Ciao datasets, separately called Ciao-5 and Ciao-28. Ciao-5 collects 5 categories of items and their corresponding users, while Ciao-28 contains all 28 categories of items and users. The rating range is [1, 5] with the step size 1.

- **Epinions**: *Epinions* comes from a well-known social based product review platform. On the website, users can rate purchases and add other users as friends. The rating values contain five discrete numbers, which are \{1, 2, 3, 4, 5\}.

- **Flixster**: *Flixster* comes from a popular movie review website, where people can add others as friends to create the social network. We can generate user-item interactions and user-user social relationships through ratings and friend lists. The range of rating value is [0.5, 5] with the step size 0.5.

For each dataset, we select 20% as the test set, 10% as valid set and remaining 70% as training set.

5.1.2 Evaluation Metrics. About the rating prediction and binary implicit feedback, these two tasks are both most common tasks in recommendations with practical significance, which are mainly determined by the benchmark datasets themselves. Without rating information, binary implicit feedback is often adopted to obtain the order list. The benchmark datasets (e.g., Ciao, Flixster) in social recommendation provide rating data, which is more valuable than common user interaction logs. Thus, many mentioned advanced models (e.g., SR-HGNN, DANSER) in the paper still adopted the rating prediction task, in order to more accurately model users’ fine-grained preferences for items. For these reasons, we also adopt rating prediction task in our work. In order to better evaluate the performance of models on the rating prediction task, we employ two widely used metrics, namely **RMSE (root mean square error)** and **MAE (mean absolute error)** [48]. The two metrics both indicate the error between the predicted value and the ground-truth, while RMSE is more sensitive to outliers.

5.1.3 Baselines. To evaluate the performance of our model, we select seven representative models, including classic and state-of-the-art (SOTA) social recommendation models as follows:

- **SoRec** [30]: This method is a classical method based on matrix factorization, which learns users’ feature vectors by decomposing the scoring matrix and the social relation matrix.

- **TrustMF** [63]: According to the direction of trust, this model maps users to the trusted space and the trustee space, by matrix factorization.

- **TrustSVD** [13]: This is one matrix factorization-based model, aggregating friends embeddings into target users embeddings to learn explicit and implicit information.

- **DSCF** [10]: This method proposes a deep learning-based framework, which captures the influence of distant social relationships on target users.
Table 3. Comparison of Features of Different Baselines and our Models

| Model       | DNN-based Model | GNN-based Model | w/ i2i edge | w/ GSL | w/ auxiliary loss function |
|-------------|-----------------|-----------------|-------------|-------|---------------------------|
| SoRec       | ✗               | ✗               | ✗           | ✗     | ✗                         |
| TrustMF     | ✗               | ✗               | ✗           | ✗     | ✗                         |
| TrustSVD    | ✗               | ✗               | ✗           | ✗     | ✗                         |
| DSCF        | ✔               | ✗               | ✗           | ✗     | ✗                         |
| GraphRec    | ✔               | ✔               | ✗           | ✗     | ✗                         |
| SR-HGNN     | ✔               | ✔               | ✗           | ✗     | ✗                         |
| DANSER      | ✔               | ✔               | ✗           | ✗     | ✗                         |
| GraphRec+   | ✔               | ✔               | ✔           | ✗     | ✗                         |
| SMIN        | ✔               | ✔               | ✔           | ✗     | ✗                         |
| GL-HGNN     | ✔               | ✔               | ✔           | ✗     | ✗                         |
| AGL-HGNN    | ✔               | ✔               | ✔           | ✗     | ✗                         |

- **GraphRec**[^4]: This method jointly captures the user-item interaction and opinion between users and items from user-item graph, and learns the heterogeneous social relationship between users from user-user graph.
- **SR-HGNN**[^5]: This method proposes the hierarchically structured graph neural networks to model the user-user and user-item graphs, to preserve both global structure of social dependencies and multi-typed interactive patterns.
- **DANSER**[^6]: This method constructs a large graph that contains user-user, item-item, and user-item sub-graphs. By modeling this large graph, it learns the dynamic and static attributes of users and items, and then fuses the dual attributes to predict users’ ratings on target items through one fusion strategy.
- **GraphRec+**[^9]: On the basis of Graphrec, Graphrec+ not only models user-item and user-user graphs, but adds item-item graph to aggregate information between similar items.
- **SMIN**[^7]: This method also constructs the global graph with three semantic relations. To investigate the potential of jointly incorporating social- and knowledge-aware relational structures, it designs a metapath-guided heterogeneous graph neural network.

Comparison of features of different baselines and our methods is shown in Table 3, where “w/ GSL” indicates whether to use graph structure learning and “w/ auxiliary loss function” indicates whether to use auxiliary loss in addition to loss function related to recommendation.

5.1.4 Parameters Setting. We implement our model based on PyTorch and DGL. We set the embedding dimension $D = 64$, and the batch size as 128. For all trainable parameters, we initialize them with a Gaussian distribution with an average of 0 and a standard deviation of 0.01. We use mini-batch Adam optimizer to train the model parameters with initial learning rate of 0.001. In order to prevent over-fitting, we add dropout layers with a probability value of 0.4 during training. In construction of item-item edges, we select top 20 items for each item to build connections, according to the similarity cosine values. For the Graph Learner, we search the weight $\lambda_w$ of learned implicit graph structure in $[0.1, 0.3, 0.5, 0.7, 0.9]$. The number $F$ of perspectives of node similarity calculation in the Graph Learner, is tuned in the set of $[1, 2, 3, 4]$. For the truncation length $L$ in Graph Learner, we obtain the optimal value in the range $[20, 40, 60, 80, 100]$ through the grid search. In addition, we set the number of graph neural network layers in range of $[1, 2, 3, 4]$.

[^4]: https://github.com/wenqifan03/GraphRec-WWW19
[^5]: https://github.com/xhcdream/SR-HGNN
[^6]: https://github.com/qitianwu/DANSER-WWW-19
[^7]: https://github.com/SocialRecsys/SMIN
Table 4. Performance Comparison of Different Models on the Four Datasets

| Models  | Ciao-5 | Ciao-28 | Epinions | Flixster |
|---------|--------|---------|----------|----------|
|         | RMSE   | MAE     | RMSE     | MAE      | RMSE     | MAE      | RMSE     | MAE      |
| SoRec   | 1.0288 | 0.7847  | 1.1881   | 0.8571   | 1.1964   | 0.9045   | 1.1237   | 0.8323   |
| TrustMF | 1.0182 | 0.8004  | 1.1506   | 0.8799   | 1.1723   | 0.8832   | 1.0594   | 0.8132   |
| TrustSVD| 0.9796 | 0.7845  | 1.0986   | 0.8480   | 1.1394   | 0.8601   | 1.0402   | 0.8097   |
| DSCF    | 0.9785 | 0.7651  | 1.0932   | 0.8391   | 1.1295   | 0.8532   | 1.0220   | 0.8034   |
| GraphRec| 0.9226 | 0.7006  | 1.0503   | 0.8157   | 1.1036   | 0.8485   | 0.9360   | 0.7161   |
| SR-HGNN | 0.9207 | 0.6993  | 1.0471   | 0.8070   | 1.1005   | 0.8219   | 0.9310   | 0.7095   |
| DANSER  | 0.9038 | 0.6857  | 1.0496   | 0.8102   | 1.0821   | 0.8164   | 0.9400   | 0.7113   |
| GraphRec+| 0.9191| 0.7065  | 1.0477   | 0.8088   | 1.0943   | 0.8377   | 0.9225   | 0.7054   |
| SMIN    | 0.9050 | 0.7014  | 1.0416   | 0.7932   | 1.0814   | 0.8171   | 0.9198   | 0.7110   |
| GL-HGNN | 0.8615 | 0.6497  | 1.0320   | 0.7763   | 1.0709   | 0.8017   | 0.9142   | 0.6993   |
| AGL-HGNN| 0.8676 | 0.6535  | 1.0330   | 0.7758   | 1.0727   | 0.8092   | 0.9091   | 0.6899   |

The smaller the RMSE and MAE, the better the performance. The bold data indicates the best performance of our methods, and the underlined data indicates the best performance of baselines. All improvements of our model over the best baseline are statistically significant ($p < 0.01$) with paired t-test.

Table 5. Model Scalability Study with Running Time (Seconds)

| Datasets | DANSER | GraphRec+ | SMIN | GL-HGNN | AGL-HGNN |
|----------|--------|-----------|------|---------|----------|
| Ciao-5   | 8.81   | 27.19     | 6.80 | 10.41   | 5.32     |
| Ciao-28  | 342.08 | 498.40    | 217.42 | 303.44 | 175.05   |
| Epinions | 605.09 | 683.18    | 481.05 | 575.95 | 398.32   |

In addition, we also apply AGL module in our experiments. We define the anchor rate $\tau = H_u/N = H_v/M$. We test the value of $\tau$ in the set $[0.01, 0.02, 0.05, 0.1, 0.15, 0.2]$.

For all the baselines, in order to achieve the best performance, we selected the best parameters of baselines strictly following the strategy and search range reported in the original papers.

5.2 Results: RQ1

The experimental results of the baseline models and our models on four datasets are shown in Table 4. For each model, we ran it five times with different run-time seed settings and reported the average scores. Besides, we record the average epoch runtime (seconds) of SOTA baselines and our methods. We implement models with the same embedding size and the same batch size on the same GPU serve. The results are shown in Table 5. Based on the comparison in the tables, we summarize our findings:

- Our model GL-HGNN comprehensively outperforms all the baseline models on the four datasets. Different from the SOTA methods: GraphRec+, DANSER and SMIN, our method can capture high-order features and different semantic information by modeling heterogeneous global graph with three kinds of edges: user-user, user-item, and item-item. In addition, to obtain a better graph structure for social recommendation, GL-HGNN employs Graph Learners to optimize initial u2u and i2i connections, which makes the graph structure with respect to downstream recommendation task. Besides, compared with GL-HGNN, AGL-HGNN can achieve comparable results, even better ones sometimes.
- Moreover, as shown in Table 5, our methods, especially AGL-HGNN, have a great advantage in efficiency. All models construct multiple subgraphs, including user-user graph, item-item...
### Table 6. Results of the Ablation Study

| Models                                  | Ciao-5 | Epinions       |
|-----------------------------------------|--------|----------------|
| GL-HGNN                                 | 0.8615 | 1.0709         |
| GL-HGNN-Attention                       | 0.8642 | 1.0745         |
| GL-HGNN-Add Attention                   | 0.8692 | 1.0738         |
| GL-HGNN-w/o u2u GL                      | 0.8708 | 1.0746         |
| GL-HGNN-w/o i2i GL                      | 0.8654 | 1.0764         |
| GL-HGNN-w/o GLs                         | 0.8813 | 1.0795         |
| GL-HGNN-w/o GLs&i2i edges               | 0.9171 | 1.1086         |
| GL-HGNN-GAT                             | 0.8718 | 1.0736         |
| GL-HGNN-GraphSAGE                       | 0.8890 | 1.1089         |

graph, and user-item graph. These SOTA methods use GNNs to model the subgraphs respectively, while our method uniformly models the global graph formed by the subgraphs, which shows that this unified modeling approach is more efficient. In addition, compared with Graph Learner (GL), our Anchor-based Graph Learner (AGL) can significantly reduce the time cost of graph structure learning without reducing performance.

- Among all the baselines, the performance of deep learning methods is better than that of traditional methods, which shows that deep learning methods have a stronger learning ability for user relations and user-item interaction signals. Moreover, the GNN-based models achieve better results than other models without graph structure. It proves the effectiveness of GNN for social recommendation. Furthermore, GraphRec+, DANSER and SMIN achieve better performance than other models without i2i subgraph construction. That suggests that adding extra i2i connections into the user-item graph can be helpful for social recommendation.

#### 5.3 Ablation Study: RQ2

In order to verify the effectiveness of some key modules, we conduct a series of ablation experiments on the Ciao-5 and Epinions datasets. The results are shown in Table 6. The impact of nodes similarity calculation. Firstly, we compare different calculation methods of nodes similarity in the Graph learner by replacing weighted cosine with attention and add attention. As can be seen in Table 6, it is clear that the weighted cosine method is the best one of three methods to capture similar attributes between nodes.

The impact of the Global Graph Learning. Besides, we explore to evaluate the effectiveness of the critical modules of Global Graph Learning. We delete each module of Global Graph Learning to observe the change of model performance, e.g., removing the u2u GL module and removing the i2i GL module. We can observe that the Graph Learner module is pivotal for the model performance by seeing “GL-HGNN-w/o GLs”. These results demonstrate that a more suitable graph structure with respect to the downstream task plays an important role. In addition, without i2i connection information, the model performance declines to a certain extent, which shows that capturing implicit item relations from the user rating matrix is valuable for the rating prediction task.

The impact of the HGNN Module. We next conduct some experiments to study the effectiveness of the HRGCN layer. To be more specific, we employ GAT [52] (which reports the best result when the number of heads is 4 in range of [1,2,4,6,8]) and GraphSage [14] to replace GCN for subgraph node aggregation with the same kind of edge. The performance of GraphSage is not as good as the other two methods, which illustrates that graph topology is important in this task and
that the loss of normalization factors may lead to the unbalance influences of nodes with huge degree differences. GAT can achieve comparable results but not better ones, which illustrates that weighting the similarity of nodes is unnecessary because the refined graph may introduce the noise relations with high attention and the reuse of similarity may amplify this kind of noise.

5.4 Hyper-parameter Sensitivity: RQ3

Global Graph Learning The performance of the Global Graph Learning is mainly affected by four important hyper-parameters, i.e., the weight value $\lambda_w$ of learnt implicit graph structure, the number of perspectives $P$, the truncation length $L$ and the anchor rate $\tau$.

- **Graph Learner**: For the first three parameters, we adjust these parameters respectively for u2u and i2i Graph Learners on Epinions datasets. The results are shown in Figure 3 and we can see that: (i) For u2u and i2i Graph Learners, appropriate implicit graph weight values are required. If the weight is too large, a lot of noise may be introduced leading to sub-optimal performance. Too small weight value also hurt model performance since the learnt implicit information would become less. (ii) The increase of numbers of perspectives in GL does not necessarily lead to an increase in performance. On the contrary, too many perspectives may result in the over-fitting. (iii) As shown in Figure 3(c), the model performance reaches the best values when $L$ is 40. The performance change in the figure can indicate that too long or too short truncation will bring loss to the model effect. The most suitable truncation length should achieve the balance between effective information and irrelevant information in the graph learning.

- **Anchor-based Graph Learner**: For the anchor rate $\tau$, we perform experiments on a single NVIDIA Tesla V100 GPU on Ciao-28 and Epinions datasets. We record the training time (seconds) of each epoch and RMSE evaluation results. As we can see from Figure 4, with the increase of the anchor rate, the performance of the model improves first and then tends to be stable, while the training time is on the rise. It can be concluded that by controlling the anchor rate within a reasonable range, the model running time can be reduced without almost loss of model performance.
Fig. 4. Performance comparison and running time (seconds) with different anchor rates.

Table 7. Performance Comparisons w.r.t. $T$

| Models       | Ciao-28         |        | Epinions       |        |
|--------------|-----------------|--------|----------------|--------|
|              | RMSE | MAE   | RMSE | MAE   |
| GL-HGNN-1    | 1.0501 | 0.8072 | 1.1276 | 0.8617 |
| GL-HGNN-2    | 1.0343 | 0.7753 | 1.0912 | 0.8345 |
| GL-HGNN-3    | 1.0320 | 0.7763 | 1.0709 | 0.8017 |
| GL-HGNN-4    | 1.0398 | 0.7847 | 1.0846 | 0.8204 |

Heterogeneous Graph Neural Network. Generally, the number $T$ of layers plays an important role for the GNN. We conduct the experiments on two datasets, and Table 7 presents the results of our model with a different number of HGNN layers. From $T = 1$ to $T = 2$, the model performance is greatly improved for both datasets, which shows the necessity of the high-order interconnection. For the Epinions dataset, from $T = 2$ to $T = 3$, the performance still increases quickly. Generally, an appropriate increase in the number of layers will make information fusion deeper. However, when $T$ is too large, the performance will drop, probably because the model introduces too much noise or becomes over-smoothing.

Hyper-parameter Settings. All the hyper-parameters have been tuned on the development set. We conduct the grid search over hyper-parameters as follows: learned graph weight $\lambda_w$ in [0.1, 0.3, 0.5, 0.7, 0.9]; loss function hyper-parameters $\beta, \beta_1, \beta_2, \eta_u, \eta_v$ in [0.1, 0.2, 0.3, 0.4, 0.5]; perspective number $P$, GNN layers number $T$ in [1, 2, 3, 4]; truncation length $L$ in [20, 40, 60, 80, 100]; anchor rate $\tau$ in [0.01, 0.02, 0.05, 0.1, 0.15, 0.2]. As shown in Table 8, we show the tuned hyper-parameters for GL-HGNN and AGL-HGNN on all datasets, respectively.

5.5 Case Study: RQ4

To show the effectiveness and rationality of Global Graph Learning module, we conduct a simple case study on several users from Ciao-5 dataset. Specifically, we make a comparison between GL-HGNN and the basic GNN-based methods (HGNN) without Global Graph Learning.

Generally, we can build a graph using user-item ratings and user social relationships as the initial graph shown in the left part of Figure 5. However, we can find that there is noise in this graph structure. Although there exist social connection between user $u_4$ and $u_2$, there are huge rating differences between user $u_4$ and $u_2$ on the same item set. Besides, despite $u_4$ and $u_1$ do not have the direct social connection, their rating histories are highly overlapped. It illustrates that they may be potential friends with the similar preferences.

On the contrary, GL-HGNN propose to adopt the Global Graph Learning module to construct item-item connections and iteratively optimize the graph structure based on the initial graph. As
Table 8. Hyper-parameters for GL-HGNN and AGL-HGNN on Datasets

| Hyper-parameters | GL-HGNN | AGL-HGNN |
|------------------|---------|----------|
|                  | Ciao-5 | Ciao-28 | Epinions | Flixster | Ciao-5 | Ciao-28 | Epinions | Flixster |
| \(\lambda_w\) (u2u) | 0.1    | 0.1     | 0.3      | 0.5      | 0.1    | 0.5     | 0.5      | 0.5      |
| \(\lambda_w\) (i2i) | 0.1    | 0.5     | 0.5      | 0.3      | 0.3    | 0.5     | 0.5      | 0.3      |
| \(P\) (u2u)      | 1      | 1       | 1        | 2        | 1      | 1       | 1        | 2        |
| \(P\) (i2i)      | 1      | 1       | 1        | 1        | 1      | 1       | 2        | 1        |
| \(L\) (u2u)      | 20     | 40      | 40       | 60       | 20     | 40      | 40       | 60       |
| \(L\) (i2i)      | 40     | 80      | 40       | 40       | 20     | 40      | 40       | 40       |
| \(T\)            | 3      | 3       | 3        | 4        | 3      | 2       | 3        | 3        |
| \(\beta\)        | 0.1    | 0.1     | 0.3      | 0.1      | 0.2    | 0.1     | 0.3      | 0.3      |
| \(\beta_1\)      | 0.0    | 0.2     | 0.0      | 0.1      | 0.1    | 0.2     | 0.0      | 0.1      |
| \(\beta_2\)      | 0.0    | 0.2     | 0.0      | 0.2      | 0.1    | 0.2     | 0.0      | 0.1      |
| \(\gamma_u\)     | 0.1    | 0.1     | 0.1      | 0.1      | 0.1    | 0.1     | 0.1      | 0.1      |
| \(\gamma_v\)     | 0.1    | 0.3     | 0.1      | 0.1      | 0.1    | 0.3     | 0.1      | 0.1      |
| anchor rate       | -      | -       | -        | -        | 0.1    | 0.1     | 0.2      | 0.02     |

"u2u" denotes user-user Graph Learner, while "i2i" denotes item-item Graph Learner.

Fig. 5. Visualization of an example for the case study from Ciao-5 data. Given the social connections among five users and the corresponding user-item ratings, the prediction target is the rating of user \(u_4\) (white circular) on item \(v_1\) (green diamond). The coverage area represents the neighboring area of the target user or item. We obtain the updated graph through the Graph Learner, based on the initial graph.

shown in the right part of Figure 5, the updated graph increases the potential relationship edge and reduces noise compared with the initial graph. We utilize the initial graph and updated graph to make scoring predictions through HGNN, respectively. Given the ground-truth rating 2, HGNN with the updated graph (GL-HGNN) predicts the result as 2.97, which is closer to the ground truth label compared to the value 3.44 generated by HGNN with the initial graph. The result demonstrates the validity and rationality of our proposed Global Graph Learning module.
6 CONCLUSION

In this paper, we proposed a novel method Graph Learning Augmented Heterogeneous Graph Neural Network (GL-HGNN), which learns the heterogeneous global graph in a unified perspective for social recommendation. We argued that, as the kinds of edges in a graph and the quality of the graph structure play a very important role, we have proposed the Global Graph Learning module to construct the heterogeneous global graph with three kinds of semantic relations and optimize the graph structure. Besides, we have proposed HGNN to extract user and item latent features in a unified perspective based on the refined global graph. Our comparative experiments and ablation studies on four datasets illustrate that GL-HGNN can learn better graph structure with respect to social recommendation, and significantly improve the performance of recommendation. In addition, we propose the Anchor-based Graph Learner (AGL) to reduce the computational complexity.

In the future, we plan to introduce more information, such as review or knowledge graph (KG), to capture the user-user and item-item relationships at a finer granularity, so that we can learn a higher quality heterogeneous global graph automatically.

REFERENCES

[1] Mikhail Belkin and Partha Niyogi. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. In Advances in Neural Information Processing Systems, Vol. 14. 585–591.
[2] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph convolutional matrix completion. In ICLR.
[3] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In The World Wide Web Conference. 151–161.
[4] Hongxu Chen, Hongzhi Yin, Tong Chen, Weiqing Wang, Xue Li, and Xia Hu. 2020. Social boosted recommendation with folded bipartite network embedding. IEEE Transactions on Knowledge and Data Engineering (2020).
[5] Yuzhou Chen, Baris Coskunuzer, and Yulia Gel. 2021. Topological relational learning on graphs. Advances in Neural Information Processing Systems 34 (2021), 27029–27042.
[6] Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. Advances in Neural Information Processing Systems 33, 19314–19326.
[7] Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-guided heterogeneous graph neural network for intent recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2478–2486.
[8] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In The World Wide Web Conference. 417–426.
[9] Wenqi Fan, Yao Ma, Qing Li, Jianping Wang, Guoyong Cai, Jiliang Tang, and Dawei Yin. 2020. A graph neural network framework for social recommendation. In Proceedings of the 13th ACM Conference on Recommender Systems. 305–313.
[10] Shangbin Feng, Zhaoxuan Tan, Rui Li, and Minnan Luo. 2022. Heterogeneity-aware Twitter bot detection with relational graph transformers. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36. 3977–3985.
[11] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In International Conference on Machine Learning, PMLR, 1972–1982.
[12] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2015. TrustSVD: Collaborative filtering with both the explicit and implicit influence of user trust and item ratings. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 29.
[13] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems (Long Beach, California, USA). Curran Associates Inc., Red Hook, NY, USA, 1025–1035.
[14] Jinquan Hang, Zheng Dong, Hongke Zhao, Xin Song, Peng Wang, and Hengshu Zhu. 2022. Outside in: Market-aware heterogeneous graph neural network for employee turnover prediction. In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining. 353–362.
[15] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 639–648.
[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural Computation 9, 8 (1997), 1735–1780.
[18] Chao Huang, Huance Xu, Yong Xu, Peng Dai, Lianghao Xia, Mengyin Lu, Liefeng Bo, Hao Xing, Xiaoping Lai, and Yanfang Ye. 2021. Knowledge-aware coupled graph neural network for social recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35. 4115–4122.

[19] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In Proceedings of the Fourth ACM Conference on Recommender Systems. 135–142.

[20] Di Jin, Cuiying Huo, Chundong Liang, and Liang Yang. 2021. Heterogeneous graph neural network via attribute completion. In Proceedings of the Web Conference 2021. 391–400.

[21] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiiliang Tang. 2020. Graph structure learning for robust graph neural networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 66–74.

[22] Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016).

[23] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. Computer 42, 8 (2009), 30–37.

[24] Nian Li, Chen Gao, Depeng Jin, and Qingmin Liao. 2022. Disentangled modeling of social homophily and influence for social recommendation. IEEE Transactions on Knowledge and Data Engineering (2022).

[25] Yiteng Pan, Fazhi He, and Haiping Yu. 2020. Learning social representations with deep autoencoder for recommender system. World Wide Web 23, 4 (2020), 2259–2279.

[26] Tiancheng Shen, Jia Jia, Yan Li, Yihui Ma, Yaohua Bu, Hanjie Wang, Bo Chen, Tat-Seng Chua, and Wendy Hall. 2020. PELA: Personality and emotion integrated attentive model for music recommendation on social media platforms. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 206–213.

[27] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Xingcheng Fu, Cheng Ji, and S. Yu Philip. 2022. Graph structure learning with variational information bottleneck. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 459–467.

[28] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In European Semantic Web Conference. Springer, 593–607.

[29] Tiancheng Shen, Jia Jia, Yan Li, Yihui Ma, Yaohua Bu, Hanjie Wang, Bo Chen, Tat-Seng Chua, and Wendy Hall. 2020. PELA: Personality and emotion integrated attentive model for music recommendation on social media platforms. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 206–213.

[30] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Xingcheng Fu, Cheng Ji, and S. Yu Philip. 2022. Graph structure learning with variational information bottleneck. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36. 4165–4174.

[31] Jiliang Tang, Huiji Gao, and Huan Liu. 2012. mTrust: Discerning multi-faceted trust in a connected world. In Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, 93–102.

[32] Jiliang Tang, Huiji Gao, and Huan Liu. 2012. mTrust: Discerning multi-faceted trust in a connected world. In Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, 93–102.

[33] Jiliang Tang, Huiji Gao, and Huan Liu. 2012. mTrust: Discerning multi-faceted trust in a connected world. In Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, 93–102.
GL-HGNN for Social Recommendation

[41] Jiliang Tang, Xia Hu, and Huan Liu. 2013. Social recommendation: A review. *Social Network Analysis and Mining* 3, 4 (2013), 1113–1133.

[42] Jiliang Tang, Suhang Wang, Xia Hu, Dawei Yin, Yingzhou Bi, Yi Chang, and Huan Liu. 2016. Recommendation with social dimensions. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[43] Zhulin Tao, Yinwei Wei, Xiang Wang, Xiangnan He, Xianglin Huang, and Tat-Seng Chua. 2020. MGAT: Multimodal graph attention network for recommendation. *Information Processing & Management* 57, 5 (2020), 102277.

[44] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

[45] Hongwei Wang, Hongyu Ren, and Jure Leskovec. 2021. Relational message passing for knowledge graph completion. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1697–1707.

[46] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *The World Wide Web Conference*. 3307–3313.

[47] Ruijia Wang, Shuai Mou, Xiao Wang, Wanpeng Xiao, Qi Ju, Chuan Shi, and Xing Xie. 2021. Graph structure estimation neural networks. In *Proceedings of the Web Conference 2021*. 342–353.

[48] Suhang Wang, Jiliang Tang, Yilin Wang, and Huan Liu. 2018. Exploring hierarchical structures for recommender systems. *IEEE Transactions on Knowledge and Data Engineering* 30, 6 (2018), 1022–1035.

[49] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 950–958.

[50] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 165–174.

[51] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of the Web Conference 2021*. 878–887.

[52] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*. 2022–2032.

[53] Xin Wang, Wenwu Zhu, and Chenghao Liu. 2019. Social recommendation with optimal limited attention. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1518–1527.

[54] Yifan Wang, Suyao Tang, Yuntong Lei, Weiping Song, Sheng Wang, and Ming Zhang. 2020. DisenHAN: Disentangled heterogeneous graph attention network for recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1605–1614.

[55] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2020), 4–24.

[56] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2020), 4–24.

[57] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2020), 4–24.

[58] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2020), 4–24.
[64] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 974–983.

[65] Junliang Yu, Hongzhi Yin, Jundong Li, Min Gao, Zi Huang, and Lizhen Cui. 2020. Enhance social recommendation with adversarial graph convolutional networks. IEEE Transactions on Knowledge and Data Engineering (2020).

[66] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. 2021. Self-supervised multi-channel hypergraph convolutional network for social recommendation. In Proceedings of the Web Conference 2021. 413–424.

[67] Qi Zhang, Jianlong Chang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2020. Spatio-temporal graph structure learning for traffic forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 1177–1185.

[68] Yuan Zhang, Xiaoran Xu, Hanning Zhou, and Yan Zhang. 2020. Distilling structured knowledge into embeddings for explainable and accurate recommendation. In Proceedings of the 13th International Conference on Web Search and Data Mining. 735–743.

[69] Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, and Yanfang Ye. 2021. Heterogeneous graph structure learning for graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35. 4697–4705.

Received 6 October 2022; revised 15 March 2023; accepted 15 May 2023