Bilateral Filtering Graph Convolutional Network for Multi-relational Social Recommendation in the Power-law Networks

MINGHAO ZHAO, QILIN DENG, KAI WANG, RUNZE WU, JIANRONG TAO, and CHANGJIE FAN, Fuxi AI Lab, NetEase Games
LIANG CHEN, Sun Yat-Sen University, China
PENG CUI, Tsinghua University, China

In recent years, advances in Graph Convolutional Networks (GCNs) have given new insights into the development of social recommendation. However, many existing GCN-based social recommendation methods often directly apply GCN to capture user-item and user-user interactions, which probably have two main limitations: (a) Due to the power-law property of the degree distribution, the vanilla GCN with static normalized adjacency matrix has limitations in learning node representations, especially for the long-tail nodes; (b) multi-typed social relationships between users that are ubiquitous in the real world are rarely considered. In this article, we propose a novel Bilateral Filtering Heterogeneous Attention Network (BFHAN), which improves long-tail node representations and leverages multi-typed social relationships between user nodes. First, we propose a novel graph convolutional filter for the user-item bipartite network and extend it to the user-user homogeneous network. Further, we theoretically analyze the correlation between the convergence values of different graph convolutional filters and node degrees after stacking multiple layers. Second, we model multi-relational social interactions between users as the multiplex network and further propose a multiplex attention network to capture distinctive inter-layer influences for user representations. Last but not least, the experimental results demonstrate that our proposed method outperforms several state-of-the-art GCN-based methods for social recommendation tasks.

CCS Concepts: • **Information systems** → **Collaborative and social computing systems and tools**; • **Mathematics of computing** → *Graph algorithms;*

Additional Key Words and Phrases: Social recommendation, power-law networks, multi-typed social relationships, heterogeneous graph neural network

ACM Reference format:
Minghao Zhao, Qilin Deng, Kai Wang, Runze Wu, Jianrong Tao, Changjie Fan, Liang Chen, and Peng Cui. 2021. Bilateral Filtering Graph Convolutional Network for Multi-relational Social Recommendation in the Power-law Networks. *ACM Trans. Inf. Syst.* 40, 2, Article 31 (September 2021), 24 pages.
https://doi.org/10.1145/3469799

Authors' addresses: M. Zhao, Q. Deng, K. Wang, R. Wu (corresponding author), J. Tao, and C. Fan, Fuxi AI Lab, NetEase Games; emails: {zhaominghao, dengqilin, wangkai02, wurunze1, hztaojianrong, fanchangjie}@corp.netease.com; L. Chen, SunYat-Sen University, China; email: chenliang6@mail.sysu.edu.cn; P. Cui, Tsinghua University, China; email: cuip@tsinghua.edu.cn.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
1046-8188/2021/09-ART31 $15.00
https://doi.org/10.1145/3469799

1 INTRODUCTION

As an effective tool to solve information overload, recommender systems have not only been extensively studied so far, but also have been widely used in the industries such as e-commerce, advertising, and games. However, several fundamental problems remain unresolved so far. For example, traditional collaborative filtering-based recommender systems often face the problems of data sparsity, long-tail recommendation, and so on [1, 2].

As a feasible way to alleviate the data sparsity problem, social recommendation aims to leverage user correlations implied by social networks to recommend potential items for users [3, 4]. Most existing social recommendation algorithms are based on the assumption that people tend to share similar interests and preferences with their friends. Therefore, the recommender system resorts to user correlations implied by social networks to recommend products for those who have few purchase records. Traditional studies on social recommendation mainly attempt to integrate social information with shallow matrix factorization models by well-designed social regularization and detailed overviews can be found in [3].

In recent years, **Graph Convolutional Networks (GCNs)**, which can naturally integrate both node attributes and topological structures in the networks, have proved to be effective and efficient in the representation learning for graph-structured data [5]. The fundamental mechanism of the GCNs is that each node updates its representation by aggregating information from the neighborhoods iteratively. Due to the end-to-end information aggregation, GCNs achieve great success in many graph-based tasks [6, 7], e.g., node classification [5, 8], link prediction [9], and graph classification [10]. The vanilla GCN [5] specifies fixed degree-based weights to neighboring nodes, resulting in the inflexibility to learn the different impacts of neighboring nodes during the aggregation phase. Further, GAT [11] assigns different neighboring nodes with trainable weights through a multi-head attention mechanism [12].

On one hand, GCNs provide powerful tools for network representation learning. On the other hand, as illustrated in Figure 1(a), data in social recommendation can be represented as a combination of heterogeneous user-user social networks (i.e., formed by interactions between users) and a user-item bipartite network (i.e., formed by interactions between users and items). Thus, GCNs can be directly applied to social recommendation. For example, DiffNet [13] utilizes the vanilla GCN to iteratively learn user representations in the user-user network. Further, both GraphRec [14] and DANSER [15] employ the GAT to learn the heterogeneous interaction strengths during information aggregation in the user-user and user-item network, respectively. It is worth mentioning that most of the above methods directly use GCN or GAT to learn user and item representations in the user and item domains, respectively.

However, through careful analysis, we find that the vanilla GCN with static normalized graph convolutional filter has limitations to learn the representations for low-degree nodes. As is well known, the degree distributions of nodes in most real networks conform to the power-law distributions, which means that most nodes in the network are long-tail nodes with small degrees [16]. As shown in Figure 1(b), the degree of item nodes in the user-item bipartite network and the user nodes in the user-user network both conform to the power-law distributions. Due to data sparsity, this may make it difficult and insufficient to learn good representations for long-tail nodes. On one hand, hub nodes usually have more supervision signals to be optimized than tail nodes. On the other hand, after theoretical analysis, we find that existing graph convolutions are inclined to strengthen hub nodes by allocating more weights for them than tail nodes during neighborhood aggregation. These factors together lead hub nodes to dominate the loss function, while the supervision signals of long-tail nodes are inevitably weakened. That is to say, the representations of the long-tail items may not be well learned, which leads to poor performance w.r.t. long-tail

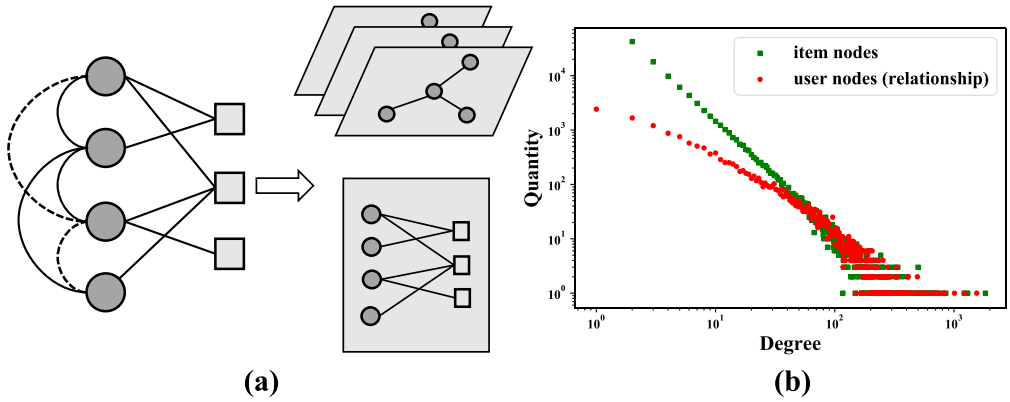

Fig. 1. (a) Example of social recommendation dataset, where circles/squares represent user/item nodes, and different types of edges between user nodes represent different types of relationships, respectively. The whole graphs consist of two heterogeneous networks: a multiplex network formed by multi-typed interactions between users (upper part) and a bipartite network formed by interactions between users and items (lower part); (b) The distributions of the degrees of user/item nodes and corresponding quantities in the user-user and user-item networks of the Epinions dataset (log-log scale).

items (see Section 5.2.2). More importantly, as the long-tail nodes occupy the main part of the network, their representations play a key role in the final performance. Previous works have revealed the power-law distribution of the items interacted by the users in the recommender systems [17–20]. Although there are many methods from different perspectives are proposed to improve the long-tail performance, in this work, we pay attention to methods based on deep learning. Prior efforts on challenging the long-tail recommendation have mainly been two-fold. One points toward data-oriented level, where auxiliary data can effectively alleviate the data sparsity problem for the long-tail items, especially when contextual information such as user and item contents are available [21, 22]. The other points towards model-oriented level. Briefly speaking, model-oriented methods are implemented by designing specific mechanisms that empower the model to optimize representations for the long-tail items [23–25]. However, as far as we know, there are few special studies on the long-tail node representation learning under the framework of GCNs. Recently, Tang et al. [26] have found that GCNs have degree-related biases that the performance of low-degree nodes is unsatisfying in node classification task. But it remains unexplored in link prediction task. Moreover, the existing methods only apply to the situation where there is only a homogeneous relationship type among users, while the heterogeneous types are ignored. In other words, they did not consider the edge information, e.g., edge type, in the period of neighboring information aggregation. Since imbalanced and sparse interactions could also show up in different types of relations [27], we also need to resolve the imbalanced and sparse issues across the types of relations. Besides, previous studies have shown that considering different ways of allocating resources (i.e., normalization) can obtain simultaneous gains w.r.t. both accuracy and diversity of recommendations [28], and different relationships can complement each other, so integrating information from multiplex relationships can improve the overall performances [29–31]. Inspired by them, in this work, we study how to leverage the specific graph convolutions to improve the long-tail node representations, as well as the situation when there exist multiple types of relationships between users.

To this end, we attempt to build multi-relational social recommendation systems based on the GCNs. To be specific, we model user-item and user-user interactions as a bipartite network and a

multiplex network, respectively. We first propose a novel **B**ilateral **F**iltering **G**raph **C**onvolutional **N**etwork **(BFGCN)** in the bipartite network and extend it to the homogeneous user-user network. Then based on the BFGCN structure, we additionally utilize a **M**ultiplex **A**ttention **N**etwork **(MAN)** to jointly learn user representations from different edge types. We name the whole framework as **BFHAN**, which means **B**ilateral **F**iltering **H**eterogeneous **A**ttention **N**etwork. To sum up, our contributions can be summarized as follows:

- We point out that the existing graph convolutional filters have limitations in node representation learning due to the long-tail degree distribution of the power-law networks and theoretically analyze the convergence values of stacking different graph convolutional filters. Further, we propose a novel graph convolutional filter, which adaptively adjusts the aggregation weights of neighboring nodes by combining the row- and column-normalized adjacency matrices, to improve node representation learning in the power-law networks.
- We extend the existing social recommendation methods from considering single-typed user relationships to multi-typed situations. Particularly, we model the heterogeneous relationships between users as a multiplex network and propose a multiplex attention network to integrate representations from different network layers.
- We demonstrate that the proposed method outperforms several state-of-the-art GCN-based social recommendation methods and heterogeneous graph convolutional networks with extensive experiments. We also conduct an ablation study to further verify the effectiveness of each component of our proposed method.

The rest of the article is organized as follows: In Section 2, we briefly review the related works. In Section 3, we introduce the definitions and notations used in this work. In Section 4, we explain each component of the proposed models in detail. In Section 5, we conduct several experiments to examine the proposed models. We conclude our work and future directions in Section 6.

2 RELATED WORK

In this section, we briefly review relevant researches for GCNs in both homogeneous and heterogeneous networks, MF-based and GCN-based social recommendation.

2.1 Graph Convolutional Networks

Recent years have witnessed great advancements in **Graph Convolutional Networks (GCNs)** for graph representation learning [6, 7]. Inspired by the first-order Chebyshev filter, the graph convolutional filter of vanilla GCN [5] is a symmetric-normalized adjacency matrix (with self-connections). Other existing filters are mainly row-normalized or column-normalized adjacency matrix that is inspired by random walk on graphs [32] or PageRank [33], respectively. To distinguish different influences of neighboring nodes during neighborhood aggregation, GAT [11] uses an attention mechanism to learn the weights between the source node and its neighbors. A more detailed survey on GCNs and their applications can be found in Reference [34]. Since the degree distributions of the real-world networks usually follow the power-law forms, representation learning for the long-tail nodes plays a crucial role for downstream tasks. However, few works have investigated the effects of different graph convolutional filters on the learning ability of the nodes in the power-law networks.

For another perspective, real-world networks are usually heterogeneous networks, which means that nodes or edges have multiple types. While GCN and GAT are designed for homogeneous networks and are not suitable for heterogeneous networks. To this end, the studies for heterogeneous graph convolutional networks have been put forward to tackle these situations. For example, for the user-item bipartite networks in the recommendation scenario, GCMC [35] proposes a graph

Table 1. Comparison of the Compositions of GCN-based Methods for Social Recommendation and Heterogeneous GCNs in Terms of Learning Paradigm and Relationship Type

	attention mechanism	heterogeneous relationship type
DiffNet	×	×
DiffNet++	✓	×
SoRecGAT	✓	×
GraphRec	✓	×
DANSER	✓	×
RGCN	×	✓
HAN	✓	✓

auto-encoder framework based on first-order neighborhood aggregation on the bipartite network. NGCF [36] exploits high-order collaborative information between users and items through stacking multiple GCN layers. For multi-relational networks in the knowledge graphs, RGCN [9] is proposed to learn the intrinsic multi-typed relationships between entities with semantic-aware matrix transformations. Further, HAN [37] extends GAT to heterogeneous information networks. It applies hierarchical GAT layers to capture the weights of the neighboring nodes and meta-paths at both node-level and semantic-level. More recently, Yang et al. [38] summarize the latest advancements in the representation learning of heterogeneous networks by providing a unified framework together with several benchmark methods and datasets for further studies.

2.2 Social Recommendation and Matrix Factorization-based Methods

As a feasible way to alleviate the data sparsity problem faced by many recommender systems, social recommendation has attracted broad attention both in academia and industry [39]. Under the assumption that the individuals tend to have similar preferences or be influenced by their friends, social recommendation resorts to user correlations implied by social networks to improve recommendation performance. Traditional social recommendation methods are mainly based on **matrix factorization (MF)** with well-designed regularization terms. For example, SoRec [21] proposes a **probabilistic matrix factorization (PMF)** model [40] that combines information from user-item and user-user interactions. SocialMF [41] incorporates the mechanism of trust propagation into MF model with the assumption that the representations of users depend on their connected neighbors. TrustMF [42] is a social CF method that captures the influence of trust propagation by a truster model and a trustee model, respectively; TrustSVD [43] is a trust-based matrix factorization model that extends SVD++ [44] with trust information. More recently, SREPS [45] considers that users may show different preferences in different scenarios and proposes an essential preference space to describe the user's multiple preferences. Finally, a comprehensive survey of early works on social recommendation can be found in References [3, 46].

2.3 Social Recommendation based on GCNs

Since data in social/non-social recommendation can be naturally represented as heterogeneous networks, attempts on the GCN-based recommendation algorithms have recently emerged. For example, GCMC [35] is a graph auto-encoder framework that applies the graph convolutions to the user-item bipartite network. However, GCMC only performs graph convolution in the user-item network and ignores the interactions between users. To leverage the social information generated by user interactions, DiffNet [13] utilizes multiple GCN layers in the user-user homogeneous network to model the social diffusion process and could be seamlessly integrated into classic CF

models. Then, DiffNet++ [47] is proposed to fuse both influence and interest diffusion in the social and interest network, respectively. Further, to consider the heterogeneous interaction strengths between neighboring nodes, SoRecGAT [48] uses multi-head and multi-layer attention mechanisms to learn the node representations in a unified space. For explicit predictions, GraphRec [14] employs an attention mechanism to learn representations for users and items by aggregating information from the user-item and the user-user networks. DANSER [15] proposes dual graph attention networks to learn representations for static and dynamic social effects in the user and item domains and then combines them with a policy-based fusion layer. Another important direction is to leverage the adversarial framework to optimize the learning process for social recommendation [49, 50]. For example, DASO [49] harnesses adversarial learning to generate hard "fake" samples rather than negative sampling to optimize the representations of the users and items better. However, all the above methods use the vanilla GCN with fixed graph convolutional filter or GAT to aggregate information from the neighboring node and only consider homogeneous relationship type between users and cannot capture different impacts from heterogeneous types. Finally, we summarize the structures of the above methods in terms of attention mechanism and heterogeneous relationships in Table 1 to clarify the differences more clearly.

3 PRELIMINARY

3.1 Definitions and Notations

Definition 1. Heterogeneous Network is a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote the set of nodes and edges in $\mathcal{G}$, respectively. $\mathcal{G}$ is associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{O}$ and an edge type mapping $\psi : \mathcal{E} \rightarrow \mathcal{R}$. If $|\mathcal{O}| + |\mathcal{R}| > 2$, then the network is named as heterogeneous; otherwise, homogeneous.

Definition 2. Bipartite Network is a network $\mathcal{BG} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ whose nodes can be divided into two disjoint and independent sets $\mathcal{U} = (u_1, u_2, \ldots, u_i)$ and $\mathcal{V} = (v_1, v_2, \ldots v_j)$ such that each edge $e \in \mathcal{E}$ connects a node in $\mathcal{U}$ to one in $\mathcal{V}$, i.e., $\mathcal{E} \subseteq \mathcal{U} \times \mathcal{V}$. B is the adjacency matrix of the bipartite network, where $B_{ij} = 1$ if $(u_i, v_j) \in \mathcal{E}$; otherwise, $B_{ij} = 0$.

Definition 3. Multiplex Network is a kind of network $\mathcal{MG} = (\mathcal{V}, \mathcal{E})$ with multiple edge types. In mathematical terms, $\mathcal{E}$ can be written as $\mathcal{E} = \cup_{r \in R} \mathcal{E}_r$, where $\mathcal{E}_r$ denotes the set of edges with type $r \in R$ and $|R| > 1$. Correspondingly, $\mathcal{A} = \cup_{r \in R} \mathcal{A}_r$, where $\mathcal{A}_r$ is the adjacency matrix of $\mathcal{MG}_r = (\mathcal{V}, \mathcal{E}_r)$.

3.2 Graph Convolutional Network

Recall that most common GCNs consist of two key components, i.e., the message function $\mathbf{M}$, which aggregates information from local neighbors' representations, and the update function $\mathbf{U}$, which combines self-representation with the aggregated information [51, 52]. Formally, the k-layer GCNs are denoted as:

$$h_{\mathcal{N}(i)}^{k-1} = \mathbf{M}(\{h_j^{k-1} | j \in \mathcal{N}(i)\}), \tag{1}$$

$$h_i^k = \mathbf{U}(h_i^{k-1}, h_{\mathcal{N}(i)}^{k-1}), \tag{2}$$

where h_i^k is the embedding vector of node i at k-th layer and $\mathcal{N}(i)$ is the set of nodes that are adjacent to node i. There are many kinds of message and update functions that have been proposed so far. Among them, the vanilla GCN motivated by the localized first-order approximation of spectral graph convolution [53] is defined as follow:

$$H^k = \sigma(\hat{A} H^{k-1} W^{k-1}), \tag{3}$$

where H^k is the hidden embedding matrix of all nodes at the k-th graph convolutional layer and $H^0 = X$ is the node feature matrix. $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, with $\tilde{A} = A + I$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$; I is the identity matrix; W^{k-1} is the weight matrix at (k-1)-th layer; $\sigma(\cdot)$ is the activation function, e.g., ReLU. In the vanilla GCN, both **M** and **U** functions are weighted sum pooling, where the weights are precomputed values. Another popular representative is GAT, which employs multi-head attention to learn the weights of different neighboring nodes during the aggregation phase:

$$H_i^k = \sigma\left(\sum_{j \in \mathcal{N}(i) \cup i} attention(H_i, H_j) H_j^{k-1} W^{k-1}\right). \tag{4}$$

It is worth noting that most current studies for GCNs are designed for homogeneous networks, however, the study for heterogeneous networks remains relatively limited, e.g., bipartite network and multiplex network.

4 METHODOLOGY

4.1 Embedding Layer

For a user-item bipartite network $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$, $\mathcal{U} \in \{u_i | i = 1, 2, \ldots, N\}$ denotes user nodes and $\mathcal{V} \in \{v_j | j = 1, 2, \ldots, M\}$ denotes item nodes. Followed by Reference [13], we define $U_i \in \mathbb{R}^d$ and $V_j \in \mathbb{R}^d$ as the embedding vectors of the user node u_i and the item node v_j, where d is the embedding size. The initial user representation is:

$$U_i^0 = \sigma(X_{u_i} W_U + E_{u_i}), \tag{5}$$

where W_U is the weight matrix and σ is the activation function (i.e., ReLU). X_{u_i} is the attribute vector of the user node u_i and E_{u_i} represents its free embedding vector that acts as complementary features and can be optimized by back-propagation algorithm. For the case when the user node has no attribute information, X_{u_i} can be simply replaced by a zero matrix. Likewise, we can get the initial representation of the item node v_j:

$$V_i^0 = \sigma(X_{v_i} W_V + E_{v_i}). \tag{6}$$

4.2 Bilateral Filtering GCN

In the early days, some collaborative filters such as **Heat Conduction (HC)** and **Mass Diffusion (MD)** were proposed to improve the diversity, including inter-list diversity and intra-list novelty, while maintaining the accuracy of recommendations [28, 54, 55]. The proposed collaborative filters have some advantages in improving the long-tail performance. As shown in Figure 2, HC, which mimics the heat diffusion across the user-item bipartite network, helps improve the diversity of recommendations, whereas MD, which propagates information in a manner akin to a random-walk process, can obtain high accuracy of recommendations. Formally, HC employs a row-normalized transition matrix, while MD employs a column-normalized one. These two processes can be formalized as follows:

$$W_{v_i v_j}^{HC} = \frac{1}{d_{v_i}} \sum_{u_k} \frac{a_{v_i,u_k} a_{u_k,v_j}}{d_{u_k}} = \left(D_v^{-1} A_{vu}^T D_u^{-1} A_{uv}\right)_{i,j}, \tag{7}$$

$$W_{v_i v_j}^{MD} = \frac{1}{d_{v_j}} \sum_{u_k} \frac{a_{v_i,u_k} a_{u_k,v_j}}{d_{u_k}} = \left(A_{vu}^T D_u^{-1} A_{uv} D_v^{-1}\right)_{i,j}, \tag{8}$$

where $W_{v_i v_j}^{HC}$ represents the redistributed weight of the resources originating from the source item node v_i allocated to the target item node v_j after conducting two-layer HC propagation; d_{u_k} indicates the degree of node u_k and $D_u = \sum_v A_{uv}$ is the diagonal degree matrix. a_{v_i,u_j} is the entry

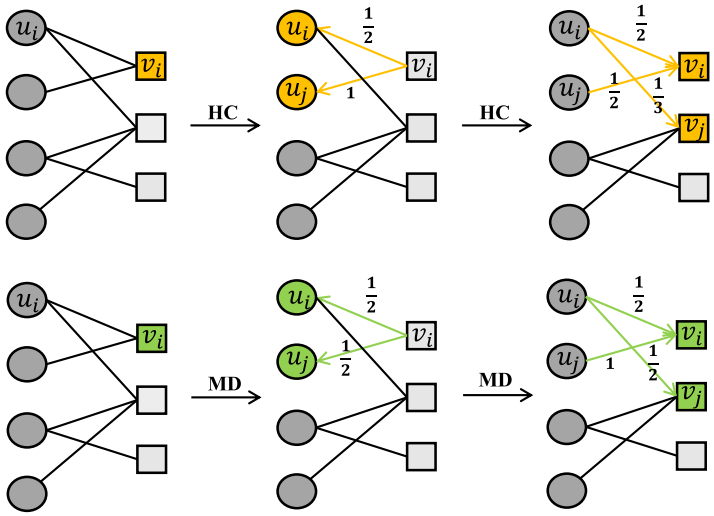

Fig. 2. The Heat Conduction (HC) and Mass Diffusion (MD) processes in the user-item bipartite network. Users are shown as circles and items as squares, with the resources originated from the item node are shown by orange or green, respectively. The diffusion weights of each process are annotated in the propagation paths.

(v_i, u_j) of the adjacency matrix A^T_{vu}. After careful analysis, we find that the HC and MD are associated with the row-normalized and column-normalized adjacency matrix of the bipartite network, respectively. Taking Figure 2 as an example,

$$W^{HC} = \left(D_v^{-1}A^T_{vu}\right)\left(D_u^{-1}A_{uv}\right) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 1 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} \frac{3}{4} & \frac{1}{4} & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}, \tag{9}$$

$$W^{MD} = \left(A^T_{vu}D_u^{-1}\right)\left(A_{uv}D_v^{-1}\right) = \begin{pmatrix} \frac{1}{2} & 1 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 1 \\ 0 & 0 & \frac{1}{2} & 0 \end{pmatrix}\begin{pmatrix} \frac{1}{2} & \frac{1}{3} & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{3} & 1 \\ 0 & \frac{1}{3} & 0 \end{pmatrix} = \begin{pmatrix} \frac{3}{4} & \frac{1}{6} & 0 \\ \frac{1}{4} & \frac{2}{3} & \frac{1}{2} \\ 0 & \frac{1}{6} & \frac{1}{2} \end{pmatrix}, \tag{10}$$

we can observe that the MD can assign lower weights to nodes with higher degrees compared to HC, since $w^{MD}_{0,1} = \frac{1}{6} < w^{HC}_{1,0} = \frac{1}{4}$ and $w^{MD}_{2,1} = \frac{1}{6} < w^{HC}_{2,1} = \frac{1}{2}$. Essentially, MD and HC work by re-weighting aggregated information from neighboring nodes based on their degrees, but in two opposite ways. In fact, re-sampling and re-weighting are widely adopted methods to resolve the problem of imbalanced data; both of them are realized by re-balancing the distributions of head and tail classes in the loss function [56, 57].

Inspired by these two graph filters, we propose a novel **Bilateral Filtering Graph Convolutional Network (BFGCN)** in the bipartite network and then extend it to the homogeneous network. Assume that $B \in \mathbb{R}^{N\times M}$ is the user-item bipartite adjacency matrix. $D_u \in \mathbb{R}^{N\times N}$ and $D_v \in \mathbb{R}^{M\times M}$ are the row-wise and column-wise diagonal degree matrix, where $D_u = \sum_v B_{uv}$ and $D_v = \sum_u B_{uv}$. $D_u^{-1}B$ and BD_v^{-1} represent the row-normalized and column-normalized adjacency matrix for item nodes, respectively. Then, we employ the gated fusion layer to selectively integrate the two graph filters:

$$H^{k-1}_{N(u)} = \alpha^{k-1}D_u^{-1}BH_v^{k-1}W^{k-1}_{rn} + (1-\alpha^{k-1})BD_v^{-1}H_v^{k-1}W^{k-1}_{cn} \tag{11}$$

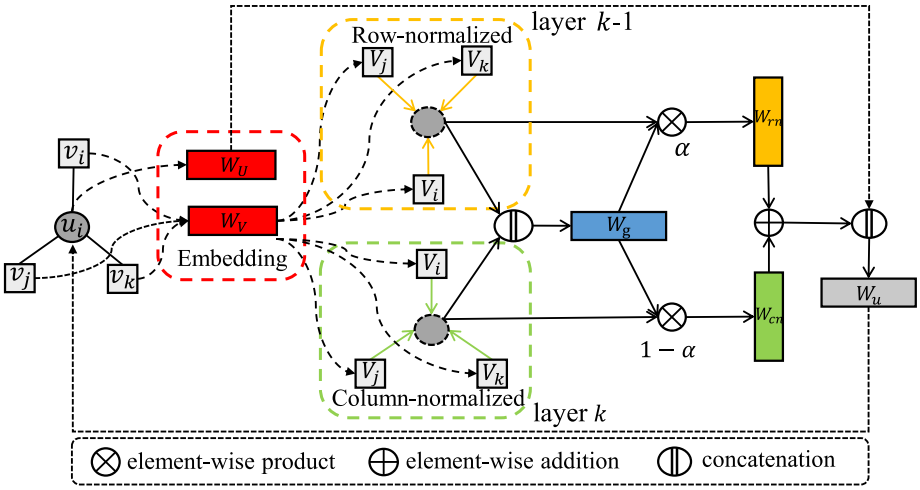

Fig. 3. Overview of the Bilateral Filtering Graph Convolutional Network (BFGCN) in the user-item bipartite network.

with

$$\alpha^{k-1} = sigmoid\Big(\Big[D_u^{-1}BH_v^{k-1} \| BD_v^{-1}H_v^{k-1}\Big]W_g^v\Big), \tag{12}$$

where $H_v^{k-1} = (V_1^{k-1}, V_2^{k-1}, \ldots, V_M^{k-1})^T$ with size $M \times d$ denotes the embedding matrix of all item nodes at (k-1)-th convolutional layer, $H_{N(u)}^{k-1}$ is the neighborhood embedding matrix, which aggregates information from the neighborhood. W_{rn}, W_{cn}, W_g^v are the trainable weight matrices with size $d \times d$, $d \times d$ and $2d \times 1$, respectively. ($\|$) represents the operation concatenation. *sigmoid* represents the sigmoid function. It is not difficult to find that this message function can adaptively assign weights for the row-normalized and column-normalized graph filters through a gating mechanism.

After aggregating the neighboring representations from the source nodes, we concatenate them with the representations of the source nodes and then feed them into a neural network:

$$H_u^k = \sigma\left(\left[H_u^{k-1} \| H_{N(u)}^{k-1}\right] W_u^{k-1} + b_u^{k-1}\right), \tag{13}$$

where $H_u^{k-1} = (U_1^{k-1}, U_2^{k-1}, \ldots, U_N^{k-1})^T$ with size $N \times d$ denotes the embedding matrix of all user nodes at (k-1)-th convolutional layer. W_u^{k-1} and b_u^{k-1} denote weight matrix with size $2d \times d$ and bias vector with size d at (k-1)-th layer. The whole process of the BFGCN for user nodes is illustrated in Figure 3. Symmetrically, we can formalize the propagation rules for item nodes similarly, which is omitted here.

Overall, we can rewrite the BFGCN in matrix form to get a holistic view of the proposed propagation rules:

$$\mathbf{H}_{\mathcal{N}}^{k-1} = \alpha^{k-1}\mathbf{D}^{-1}\mathbf{A}\mathbf{H}^{k-1}\mathbf{W}_{rn}^{k-1} + (1-\alpha^{k-1})\mathbf{A}\mathbf{D}^{-1}\mathbf{H}^{k-1}\mathbf{W}_{cn}^{k-1} \tag{14}$$

with

$$\alpha^{k-1} = sigmoid([\mathbf{D}^{-1}\mathbf{A}\mathbf{H}^{k-1} \| \mathbf{A}\mathbf{D}^{-1}\mathbf{H}^{k-1}]\mathbf{W}_{\mathrm{g}}), \tag{15}$$

where $\mathbf{D}^{-1} = \begin{pmatrix} D_u^{-1} & \mathbf{0} \\ \mathbf{0} & D_v^{-1} \end{pmatrix}$, $\mathbf{A} = \begin{pmatrix} \mathbf{0} & B \\ B^T & \mathbf{0} \end{pmatrix}$ and $\mathbf{0}$ is all zero matrix; $\mathbf{H}^{k-1} \in \mathbb{R}^{(N+M)\times d}$ is the stacking embedding matrix of all nodes at (k-1)-th layer. $\mathbf{W}_{rn}$, $\mathbf{W}_{cn}$, $\mathbf{W}_{\mathrm{g}}$ are the weight matrices with size

$d \times d$, $d \times d$ and $2d \times 1$. Take the two-layer graph convolution with the row-normalized filter as an example:

$$(\mathbf{D}^{-1}\mathbf{A})(\mathbf{D}^{-1}\mathbf{A})\mathbf{H} = \begin{pmatrix} D_u^{-1}BD_v^{-1}B^T & \mathbf{0} \\ \mathbf{0} & D_v^{-1}B^TD_u^{-1}B \end{pmatrix} \begin{pmatrix} H_u \\ H_v \end{pmatrix}. \tag{16}$$

The formulation is actually the same as Equation (7). By the design of the gating mechanism, we adaptively combine the HC and MD processes in the graph convolutional network. After conducting the message function, we update the representation matrix of all nodes as follows:

$$\mathbf{H}^k = \sigma\left(\left[\mathbf{H}^{k-1} \| \mathbf{H}_{\mathcal{N}}^{k-1}\right] \mathbf{W}^{k-1} + \mathbf{b}^{k-1}\right). \tag{17}$$

We can find that the convolution form of the bipartite network is consistent with that of the homogeneous network. By implementing the matrix-form propagation rules in the bipartite network, we can simultaneously and efficiently update the representations for all users and items.

4.3 Multiplex Attention Network

Then, we begin to show how to incorporate multi-typed social interactions into user representations. Since we model multi-typed relationships between users as a user-user multiplex network $\mathcal{MG}(\mathcal{U}, \mathcal{E})$ where each edge type $\mathcal{MG}_r(\mathcal{U}, \mathcal{E}_r)$ forms a homogeneous network, we can conduct graph convolution in each network layer, respectively, then combine them through a well-designed pooling operation to capture different influences of each type. Assume that $A_r \in \mathbb{R}^{N \times N}$ is the adjacency matrix of $\mathcal{MG}_r$ and $D_r \in \mathbb{R}^{N \times N}$ is the corresponding diagonal degree matrix, where D_r is a diagonal matrix and $D_r = \Sigma_j(A_r)_{\cdot j}$. Similarly, we can extend BFGCN to the homogeneous network as follows:

$$H_{\mathcal{N}_r(u)}^{l-1} = \beta^{l-1} D_r^{-1} A_r H_{u,r}^{l-1} W_{rn,r}^{l-1} + (1 - \beta^{l-1}) A_r D_r^{-1} H_{u,r}^{l-1} W_{cn,r}^{l-1} \tag{18}$$

with

$$\beta^{l-1} = sigmoid\left(\left[D_r^{-1} A_r H_{u,r}^{l-1} \| A_r D_r^{-1} H_{u,r}^{l-1}\right] W_{g,r}^{u}\right) \tag{19}$$

$$H_{u,r}^{l} = \sigma\left(\left[H_{u,r}^{l-1} \| H_{\mathcal{N}_r(u)}^{l-1}\right] W_{u,r}^{l-1} + b_{u,r}^{l-1}\right), \tag{20}$$

where $H_{u,r}^{l} = (U_{1,r}^{l}, U_{2,r}^{l}, \ldots, U_{N,r}^{l})^T$ with size $N \times d$ is the embedding matrix of all user nodes at lth convolution layer on edge type r. $W_{rn,r}$, $W_{cn,r}$, and $W_{g,r}^{u}$ are trainable parameters with size $d \times d$, $d \times d$, and $2d \times 1$, respectively. Compared to the vanilla GCN [5], the main difference between us is that we adaptively combine the row- and column-normalized adjacency matrix, whereas theirs is the symmetric normalized one without learnable parameters.

We then stack user representations of all layers together as $\mathbf{H}_u = \{H_{u,1}^{l}, \ldots, H_{u,r}^{l}, \ldots, H_{u,R}^{l}\}$ and reshape it as $H_{i,j,r} \in \mathbb{R}^{N \times d \times |R|}$, where i, j, and r denote node, representation dimension, and edge type, respectively. Currently, attention mechanism allows the model to focus on the most informative influential factors while filtering the less-informative ones [58]. In the real world, users can be affected by neighbors with different relationship types. Inspired by channel-wise attention in the field of computer vision [59, 60], we propose a novel **Multiplex Attention Network (MAN)** to capture different inter-layer influences of the user-user multiplex network. As illustrated in Figure 4, we first apply the weighted pooling along the representation dimension j for node i on edge type r, then squeeze the three-order representation tensor into a matrix along the dimension j:

$$z_{i,r} = squeeze_j\left(\sum_j w_j H_{i,j,r}\right), \tag{21}$$

where w_j is the trainable scalar. Then, we utilize a two-layer fully connected network followed by a softmax function along the edge type r to generate the attention weight $a_{i,r}$ over each edge type r for node i:

$$\boldsymbol{a}_{i,r} = softmax_r(sigmoid(\sigma(\boldsymbol{z}_{i,r} W_1) W_2)), \tag{22}$$

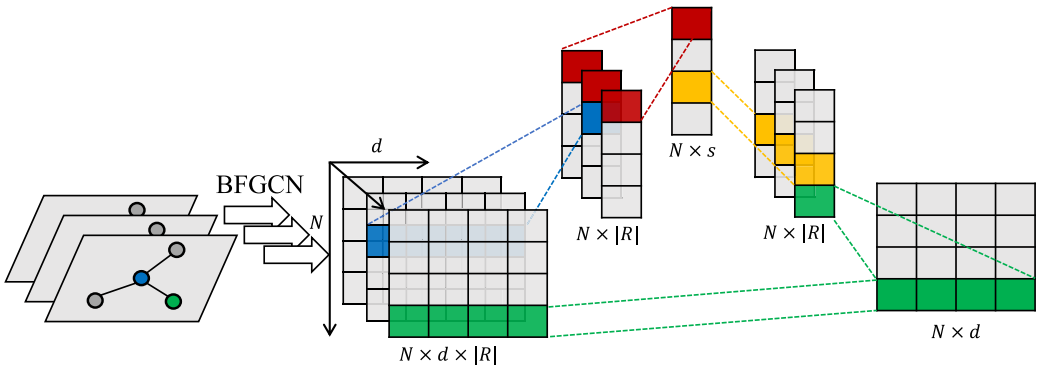

Fig. 4. Overview of the Multiplex Attention Network (MAN). The multi-channel attention mechanism learns the different inter-layer influences of the user-user multiplex network.

where σ is ReLU function. W_1 and W_2 are trainable weight matrices with the shape of $|R| \times s$ and $s \times |R|$, where s is empirically set as $\frac{d}{2}$. Thus, we obtain the final user representation by aggregating weighted representations from different network layers, denoted as:

$$\tilde{U}_i^l = \sigma\left(\sum_{r \in R} a_{i,r} U_{i,r}^l\right). \tag{23}$$

4.4 Model Architecture

In the above sections, we have explained the proposed bilateral filtering graph convolutional network and multiplex attention network in detail, respectively. As illustrated in Figure 5, we combine them as the whole framework. BFHAN contains two major components: BFGCN can help learn better presentation of users/items that have imbalanced interactions in specific relationship types, while MAN can adaptively determine the importance of each relationship type and resolve the imbalance within each type at a hierarchical level. After propagating with K and L times in the user-item and user-user network, respectively, we obtain the user representation, i.e., U_i^K and $\tilde{U}_i^L$, together with item representation, i.e., V_j^K. To leverage the information from every convolutional layer, we concatenate the user/item representations learned by different graph convolutional layers to obtain the final representations, denoted as:

$$\begin{aligned} \hat{U}_i &= U_i^0 \| \cdots \| U_i^K \| \tilde{U}_i^L \\ \hat{V}_j &= V_j^0 \| \cdots \| V_j^K. \end{aligned} \tag{24}$$

Since the above architecture is independent of the downstream task, the model parameters can be trained through a task-specific loss function. For the regression task such as rating prediction, we employ a **multilayer perceptron (MLP)** as the output layer and minimize the mean square error between the predicted values and the ground-truth ones. $L2$ regularization is also applied to avoid model overfitting. The whole loss function is denoted as follows:

$$\begin{aligned} \hat{r}_{ij} &= \mathbf{MLP}(\hat{U}_i \| \hat{V}_j) \\ \mathcal{L} &= \sum_{i,j \in \mathcal{T}} (\hat{r}_{ij} - r_{ij})^2 + \lambda \mathcal{L}_{reg}, \end{aligned} \tag{25}$$

where $\mathcal{T}$ is the training set, $\hat{r}_{ij}$ is the predicted rating value of user i for item j , r_{ij} is the ground-truth rating, and λ is the weight for $L2$ regularization. For the classification task such as node classification, we employ a bilinear decoder followed by a softmax function as output layer and

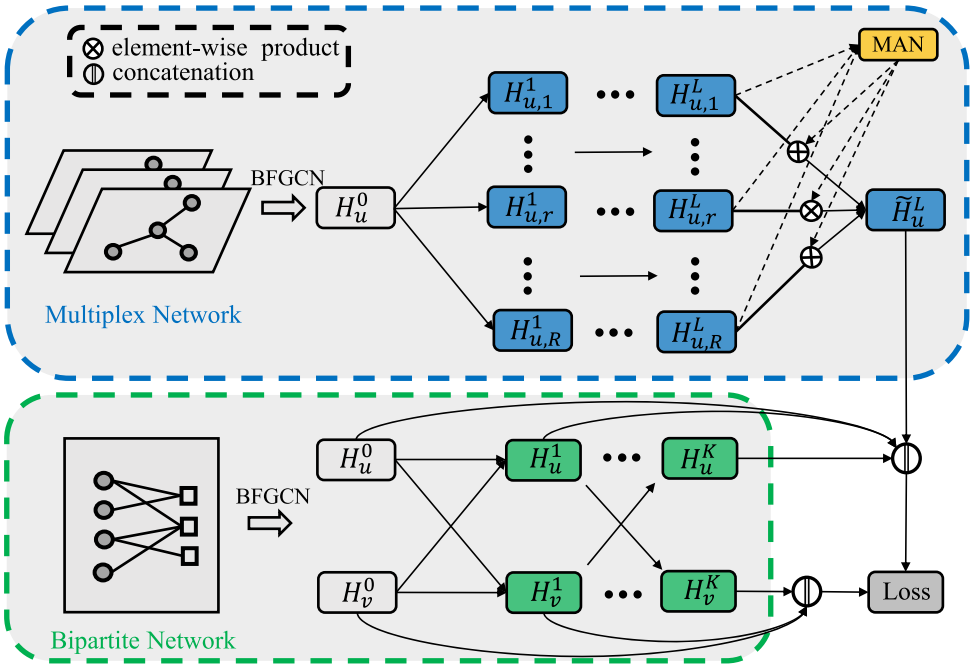

Fig. 5. The overall architecture of BFHAN. It contains two major components, i.e., BFGCN in the user-item bipartite network and the user-user multiplex network, together with MAN, which integrates information from the user-user multiplex network.

minimize the cross-entropy loss, denoted as follows:

$$\begin{aligned} \hat{y}_{ij} &= softmax_j\left(\hat{U}_i W_b \hat{V}_j^T\right) \\ \mathcal{L} &= -\sum_{i\in\mathcal{T}}\sum_j y_{ij}\log\hat{y}_{ij} + \lambda\mathcal{L}_{reg}, \end{aligned} \tag{26}$$

where W_b is a trainable parameter matrix and $\hat{y}_{ij}$ is the predicted possibility for user u_i belonging to v_j, and y_{ij} is the ground-truth label of user u_i.

4.5 Model Analysis

We consider an undirected and connected graph $\mathcal{G} = (V, E)$ with adjacency matrix $\mathbf{A}$ and corresponding diagonal degree matrix $\mathbf{D} = diag(d_1, d_2, \ldots, d_n)$. We add self-loop to each node in the $\mathcal{G}$, then the adjacency matrix and diagonal degree matrix change to $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}} = diag(d_1 + 1, d_2 + 1, \ldots, d_n + 1)$, respectively. Then, we have the following theorem:

Theorem 1. *If a simple graph $\mathcal{G}(V, E)$ with adjacency matrix $\mathbf{A}$ and degree matrix $\mathbf{D}$ is connected, when each node of $\mathcal{G}$ is added self-loop and the corresponding adjacency and degree matrix are denoted as $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{D}}$, then*

$$\begin{aligned} \lim_{k\to+\infty}(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}})^k_{i,j} &= \frac{d_j+1}{2m+n} \\ \lim_{k\to+\infty}(\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1})^k_{i,j} &= \frac{d_i+1}{2m+n} \\ \lim_{k\to+\infty}(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}})^k_{i,j} &= \frac{\sqrt{(d_i+1)(d_j+1)}}{2m+n} \end{aligned} \tag{27}$$

where $n = |V|$, $m = |E|$ and $d_i = \sum_j \mathbf{A}_{ij}$.

Proof. The power of the row-normalized adjacency matrix $\mathbf{P} = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$ can be modeled as random walks on graph $\mathcal{G}$. Note that, since extra self-connections are added, $\mathcal{G}$ is non-bipartite. Since $\mathcal{G}$ is strongly connected and non-bipartite, this process is an irreducible and aperiodic Markov chain with stationary probability vector $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_n)$, which are proportional to the degrees of the nodes, i.e., $\pi_i = \frac{d_i+1}{2m+n}$ [61].

- Since $\mathbf{P}$ is the transition probability matrix of $\mathcal{G}$, $\lim_{k\to+\infty}(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}})^k_{i,j} = \lim_{k\to+\infty}\mathbf{P}^k_{ij} = \pi_j = \frac{d_j+1}{2m+n}$.
- Due to $\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1} = (\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}})^T$, we can derive that $\lim_{k\to+\infty}(\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1})^k_{i,j} = \lim_{k\to+\infty}(\mathbf{P}^T)^k_{ij} = \lim_{k\to+\infty}(\mathbf{P}^k)^T_{ij} = \frac{d_i+1}{2m+n}$.
- Assume that $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}} = \mathbf{Q}$, we can derive that $\lim_{k\to+\infty}(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}})^k_{i,j} = \lim_{k\to+\infty}\mathbf{Q}^k_{i,j} = \lim_{k\to+\infty}(\tilde{\mathbf{D}}^{\frac{1}{2}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Q}^k)_{i,j} = \lim_{k\to+\infty}(\tilde{\mathbf{D}}^{\frac{1}{2}}\mathbf{P}^k\tilde{\mathbf{D}}^{-\frac{1}{2}})_{i,j} = \frac{\sqrt{(d_i+1)(d_j+1)}}{2m+n}$. □

In fact, the above Theorem 1 can be generalized to the following proposition and the proof is the same as above:

Proposition 2. *Giving a simple graph $\mathcal{G}(V, E)$ with adjacency matrix $\mathbf{A}$ and degree matrix $\mathbf{D}$ is connected, when each node of $\mathcal{G}$ is added self-loop, then*

$$\lim_{k\to+\infty}(\tilde{\mathbf{D}}^{-r}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-(1-r)})^k_{i,j} = \frac{(d_i+1)^{1-r}(d_j+1)^r}{2m+n}, \tag{28}$$

where $r \in [0, 1]$, $n = |V|$ and $m = |E|$.

Theorem 1 and Proposition 2 show that the above graph convolutional filters described above will converge to node degree after stacking infinite graph convolutional layers, which mean that using any of them as a propagation filter will generate indistinguishable representations for each node. Therefore, Theorem 1 and Proposition 2 to some extent show that over-smoothing problem exists in the vanilla GCN no matter which filter to choose. In fact, the similar properties have already been revealed by References [8, 62]. Differently, we prove them in the view of the stationary distribution of a Markov chain and further deduce the formulations of different graph convolutional filters.

Further, we find that different convolutional filters have different effects on the neighborhood aggregation. Assume that $\mathbf{H}^0 = (h^0_1, h^0_2, \ldots, h^0_n)^T$ is the initial node embedding matrix, where $h^0_i \in \mathbb{R}^{1\times d}, i = 1, 2, \ldots, n$. If we ignore the nonlinear activation function and transformation matrix, then propagating k times GCN can be formalized as $\mathbf{H}^k = f(\mathbf{A})\mathbf{H}^{k-1}$, where $f(\mathbf{A})$ is the graph convolutional filter. Then, we have the following conclusions:

- when $f(\mathbf{A}) = \tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$, $\lim_{k\to+\infty}\mathbf{H}^k_i = \lim_{k\to+\infty}(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}})^k\mathbf{H}^0_i = \frac{d_1+1}{2m+n}(h^0_1)^T + \frac{d_2+1}{2m+n}(h^0_2)^T + \cdots + \frac{d_n+1}{2m+n}(h^0_n)^T$, we can find that the weights of each component h^0_i in the final embedding h^k_i is proportional to the degree d_i, which indicates that h^k_i is influenced more by hub nodes with high degrees.
- when $f(\mathbf{A}) = \tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1}$, $\lim_{k\to+\infty}\mathbf{H}^k_i = \lim_{k\to+\infty}(\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1})^k\mathbf{H}^0_i = \frac{d_i+1}{2m+n}[(h^0_1)^T + (h^0_2)^T + \cdots + (h^0_n)^T]$, we can find that each h^0_i have the same weight in h^k_i, which means that h^k_i can contain more information of the nodes with low degrees, i.e., long-tail nodes.
- when $f(\mathbf{A}) = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, $\lim_{k\to+\infty}\mathbf{H}^k_i = \lim_{k\to+\infty}(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}})^k\mathbf{H}^0_i = \frac{\sqrt{d_i+1}}{2m+n}[(\sqrt{d_1+1})(h^0_1)^T + (\sqrt{d_2+1})(h^0_2)^T + \cdots + (\sqrt{d_n+1})(h^0_n)^T]$, the weights of hub nodes are reduced more

than that of low-degree nodes due to the square root function, which means that the symmetric-normalized filter alleviates the weights of representations of hub nodes while comparatively enhancing the ones of long-tail nodes.

Overall, $\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$ tends to strengthen the representations of hub nodes during neighborhood aggregation. On the contrary, $\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1}$ tends to strengthen the representations of long-tail nodes, which is consistent with the example shown in Equation (10) that MD can assign higher weights for the low-degree nodes. As a tradeoff, $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ alleviates the impacts of hub nodes while relatively enhancing the long-tail nodes through the square root function. However, all the above filters have limitations to learn node representations, since they cannot adjust the weights for the hub and long-tail nodes, respectively, when aggregating neighboring representations, while our proposed BFGCN has an adaptive weighting mechanism to achieve this goal.

ALGORITHM 1: BFHAN

Input: the user-item bipartite network $\mathcal{BG}$ and user-user multiplex network $\mathcal{MG}$
Output: Prediction function $\mathcal{F}(u, v|\Theta)$

Initial the embedding of users and items as U_i^0 and V_i^0 through Equation(6);
for $k = 1, 2, \ldots, K$ **do**
    Update the user embedding H_u^k and item embedding H_v^k according to Equations (14)–(17);
for $r = 1, 2, \ldots, R$ **do**
    for $l = 1, 2, \ldots, L$ **do**
        Update $H_{u,r}^l$ according to Equations (18)–(20);
Update the user and item embedding $\tilde{U}_i^L$ according to Equations (23)–(24);
for *number of training epochs* **do**
    Calculate gradients $\partial\mathcal{L}/\partial\Theta$ on the mini-batch training set by back-propagation through Equation (25) or Equation (26); Update the model parameters Θ through SGD;
return $\mathcal{F}(u, v|\Theta)$

4.6 Model Complexity

Algorithm 1 describes the implementation details of our proposed model. As described above, BFHAN is mainly composed of two graph convolutional layers in the user-item bipartite network and the user-user multiplex network, respectively. Fortunately, both of them are sparse networks so the layer-wise graph convolutional filter can be implemented efficiently through sparse matrix multiplication. The total cost of the sparse matrix multiplication is $O(|\mathcal{E}|d)$, the cost of the feature transformation is $O((|\mathcal{U}| + |\mathcal{V}|)d^2)$, and the cost of the gating mechanism is $O((|\mathcal{U}| + |\mathcal{V}|)d)$. Therefore, the whole time complexity of the K-layers BFGCN in the bipartite network is $O(K(|\mathcal{E}|d + (|\mathcal{U}| + |\mathcal{V}|)(d^2 + d)))$. Since BFGCN can be naturally extended to the homogeneous networks, the time complexity of the BFGCN for the user-user multiplex network is analogous to that of the bipartite network. Finally, the time complexity of the MAN is about $O(|R|(|\mathcal{U}| + |\mathcal{V}|)d)$.

5 EXPERIMENTS

5.1 Experimental Settings

5.1.1 Datasets. In this study, we verify the performance of the proposed method with three different tasks.

Table 2. Datasets for Social Recommendation Task with Single-typed and Multi-typed Relationships

	Ciao	Epinions	MMORPG
# users	7,375	22,166	25,244
# items	106,797	296,277	551
# user-item interactions	284,052	922,267	25,244
# user-user relationships	111,781	355,727	866,967
# relationship types	1	1	3

For the case of single-typed relationships, we use two benchmark datasets, i.e., Ciao and Epinions,[1] which are taken from popular social networking websites Ciao[2] and Epinions.[3] Both of them consist of user ratings for items and trust relationships between users. Especially, the trust network between users is homogeneous and the statistics of the datasets are presented in Table 2. Since we aim to use social information to recommend items, we remove the isolated user nodes. For this case, we adopt the mean square loss function and dub it as a rating prediction task.

For the case of multi-typed relationships, we use a commercial MMORPG dataset[4] where there are three types of relationships between players (i.e., friendship, transaction, and teammate) and each player belongs to only one organization (i.e., guild). We aim to recommend the guilds for the players who have not joined any guilds. For this case, we adopt the cross-entropy loss function and dub it as guild recommendation task.

It is worth noting that there are few available public datasets for multi-relational social recommendation. To replicate our study, on one hand, we would like to open our codes for public research[5]; on the other hand, since heterogeneous networks that contain multi-typed relationships (defined by meta-paths) between nodes are well studied, therefore, we use the widely used datasets, i.e., ACM, DBLP,[6] and IMDB[7] to leverage multi-typed relationships between nodes for node classification. Each dataset consists of different types of edges between nodes, which are generated by different predefined meta-paths. For this case, we adopt the cross-entropy loss function and dub it as a node classification task. The statistics of the datasets are presented in Table 3.

5.1.2 Evaluation Metrics. To evaluate the performance for each task, several popular metrics are adopted, respectively. For the rating prediction task, we use **Mean Absolute Error (MAE)** and **Root Mean Square Error (RMSE)**. Note that smaller values of MAE and RMSE mean better performance. For the guild recommendation task, we employ the **top-k Hit Rate (HR@k)** and the **top-k Mean Reciprocal Rank (MRR@k)**. Especially, HR@k is computed as the proportion of the ground-truth guilds inside the top-k predicted list for users. And MRR@k is the average of the reciprocal ranks of ground-truth guilds in the top-k predicted list for users. For the node classification task, we use **Micro F1-Score (F1)** as an evaluation metric.

5.1.3 Baselines. In our experiments, the proposed model is compared with various state-of-the-art baselines for single or multi-relational social recommendation. The details of these baselines are listed as follows:

[1] http://www.cse.msu.edu/~tangjili/trust.html.
[2] http://www.ciao.co.uk.
[3] http://www.epinions.com.
[4] https://n.163.com/.
[5] https://github.com/fuxiAIlab/BFHAN.
[6] https://github.com/Jhy1993/HAN.
[7] https://github.com/seongjunyun/Graph_Transformer_Networks.

Table 3. Datasets for Node Classification Task with Multi-typed Relationships

	ACM	DBLP	IMDB
# nodes	3,025	4,057	12,772
# edge types	2	3	4
# Training	600	800	300
# Validation	300	400	300
# Test	2,125	2,857	2,339

- **GCMC** [35] is a matrix completion model based on GCN, which aggregates information from one-hop neighbors in the user-item bipartite network.
- **NGCF** [36] employs GCN to capture the user-item collaborative filtering effect. We replace the original loss function with the mean square error.
- **DiffNet** [13] is a deep influence propagation model that utilizes GCN to mimic the social influence propagation process in the user-user network.
- **GraphRec** [14] applies attention mechanisms in both the user-user and user-item networks to capture heterogeneous strengths of both interactions for social recommendation.
- **DANSER** [15] is a dual graph convolutional model that leverages the graph attention networks to learn the representations for the nodes and items from the static and dynamic social effects, respectively.
- **RGCN** [9] is designed for representation learning of multi-relational networks. We use it as a baseline for modeling multi-typed relationships between users.
- **HAN** [37] is a heterogeneous graph attention network that utilizes a hierarchical attention mechanism to integrate the representations of neighboring nodes with different semantics.

We also studied the effect of several variants of our proposed method as follows:

- **BFHAN** (sym) is a variant of BFHAN that replaces BFGCN by the symmetric normalized adjacency matrix.
- **BFHAN** ($\alpha = \beta = 0.5$) is a variant of BFHAN that treats row-normalized and column-normalized adjacency matrix equally.
- **BFHAN** (mean) is a variant of BFHAN that replaces MAN with mean pooling.

It is worth noting that DiffNet, GraphRec, and DANSER are designed for the situations where there is only one type of relationship between users. However, all the above methods only consider the situation where there exists a single kind of relationship between users, while RGCN and HAN can handle the situations where there are multiple relationships between users. Thus, we choose RGCN and HAN as additional competitive baselines to take multi-typed relationships into account.

5.1.4 Parameter Settings. We implement the proposed method on Tensorflow [63], which is a popular open-source platform for machine learning. For the rating prediction and guild recommendation tasks, we use 80% of data as the training set to learn model parameters, 10% as validation data to fine tune hyper-parameters, and the rest 10% as the test set for final performance comparison. For the node classification task, we use the same train, validation, and test datasets as Reference [37]. For all the tasks, we use the xavier initializer [64] to initialize the model parameters and apply the Adam optimizer [65] to optimize them, where the learning rate is 0.001. To avoid overfitting, we tune the L_2 regularization coefficient in $\{10^{-5}, 10^{-4}, \ldots, 10^{-1}\}$ and dropout ratio [66] in $\{0, 0.1, 0.3, 0.5, 0.7, 0.9\}$, respectively. Meanwhile, an early stopping strategy is adopted, where we stop training if the performance on validation set degenerates for 30 successive

Table 4. Evaluation Results of Different Methods on Social Recommendation with Single-typed Relationships

	Ciao		Epinions	
	MAE	RMSE	MAE	RMSE
GCMC	0.9383	1.1776	0.9436	1.1994
NGCF	0.7504	0.9911	0.8401	1.0718
DiffNet	0.7418	0.9789	0.8201	1.0610
GraphRec	0.7648	0.9775	0.8417	1.0832
DANSER	0.7573	0.9819	0.8098	1.0760
BFGCN (sym)	0.7591	0.9843	0.8542	1.0775
BFGCN ($\alpha = 0.5$)	0.7762	0.9875	0.8550	1.0853
BFGCN	0.7431	0.9890	0.8376	1.0695
BFHAN ($r = 1$, sym)	0.7269	0.9776	0.8087	1.0446
BFHAN ($r = 1$, $\alpha = \beta = 0.5$)	0.7356	**0.9749**	0.8122	1.0473
BFHAN ($r = 1$)	**0.7251**	0.9762	**0.8046**	**1.0403**

epochs. For the rating predicting task, we use mini batch training and the batch size is searched in {128, 256, 512, 1024, 2048}. Since residual network is introduced, we set the dimension of all hidden layers in the neural network to be the same as the user/item representation for reducing the number of parameters and the representation size is searched in {8, 16, 32, 64, 128, 256}. For all the hyper-parameters of the baseline methods, we use the values suggested by the corresponding papers with careful fine-tuning.

5.2 Performance Comparison

5.2.1 Rating Prediction with Single-typed Relationships. To verify the effectiveness of our proposed BFGCN in the bipartite network and homogeneous networks, we conduct experiments to compare with other benchmark methods in the rating prediction task and report the average results after running 10 times. In this task, as there exists a single type of relationship between users, our proposed method is simplified as BFHAN ($r = 1$), where $r = 1$ means the relationship type equals 1. For a fair comparison, we set $L = 1$ in Equation (24), which means BFHAN ($r = 1$) only aggregates information from one-hop neighbors in the user-user social network. As shown in Table 4, BFHAN ($r = 1$) outperforms all the baselines in both Ciao and Epinions datasets, i.e., 4.25% MAE improvement for Ciao and 3.32% RMSE improvement for Epinions than DANSER.

Further, we perform ablation studies to verify the effect of several variants and the results are shown in Table 4. We can see that both BFHAN (sym) and BFHAN ($\alpha = \beta = 0.5$) degrade the prediction performance. Then, we remove the user-user social network to conduct experiments with several variants of the BFGCN, i.e., BFGCN (sym) and BFGCN ($\alpha = 0.5$). As can be observed from Table 4, the complete BFGCN outperforms the two variants mentioned above. Moreover, BFGCN still achieves better performance than GCMC and NGCF, both of which use the symmetric normalized adjacency matrix to aggregate neighboring representations. In general, the ablation experiments indicate that our proposed BFGCN, which can adaptively adjust the row and column normalized adjacency matrices, is more flexible and better than the fixed one adopted by vanilla GCN .

We also study the influence w.r.t. propagation layer number K in the user-item bipartite network on the model performance. Table 5 shows the results of different layer number K. We can find that increasing the layer depth does not improve the accuracy of the model in the Ciao and Epinions

Table 5. Effect of Different Propagation Layer Number K in the Bipartite Network

	Ciao		Epinions	
	MAE	RMSE	MAE	RMSE
BFHAN (r = 1, K = 1)	**0.7251**	**0.9762**	**0.8046**	**1.0403**
BFHAN (r = 1, K = 2)	0.7447	0.9786	0.8116	1.0705
BFHAN (r = 1, K = 3)	0.7530	0.9856	0.8931	1.1459

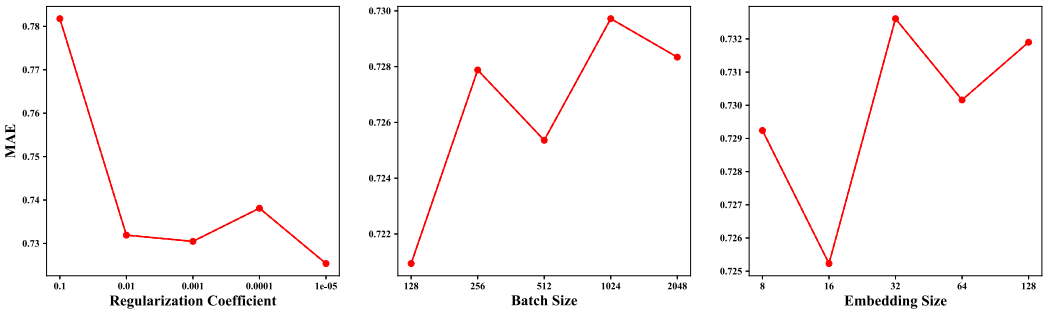

Fig. 6. Parameter sensitivity w.r.t. regularization coefficient, batch size, and embedding size on the Ciao dataset.

datasets. This indicates that aggregating high-order collaborative information may suffer from the overfitting issue.

Finally, we study the sensitivity of parameters on the Ciao dataset and report the results of MAE in Figure 6. We can observe that as the regularization coefficient decreases, the performance w.r.t. MAE gets better. That is in line with expectations that suitable regularization coefficient can avoid overfitting. For batch size, we can see that small batch size can get better performance, since mini-batch gradient descent training can make the model less prone to overfitting. As for embedding size, we can find that with the growth of the embedding size, the performance raises first and then starts to drop and the best embedding size is about 16.

5.2.2 Impacts of Different Graph Convolutional Filters. To further illustrate the impact of BFGCN, we show the performance of various graph convolutional filters, i.e., $\mathbf{D}^{-1}\mathbf{A}, \mathbf{A}^{-1}\mathbf{D}$, $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, GAT and our proposed BFGCN, under different sparsity of the observed items in the user-item bipartite network. Specifically, we divide items into different groups according to the number of observed ratings (the degree of item nodes) in the training data. Then, we show the performance and quantity of each group under different graph convolutional filters in Figure 7. We can observe that the degrees of most nodes are very low due to the power-law distribution, that is, most of the nodes in the network are long-tail nodes. Further, we find an interesting phenomenon that the row-normalized filter $\mathbf{D}^{-1}\mathbf{A}$ performs the worst among the above filters in both Ciao and Epinions when the degree of the item node is less than 2, but consistently well as the degree increases compared to $\mathbf{A}\mathbf{D}^{-1}$ and $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. On the contrary, the column-normalized filter $\mathbf{A}\mathbf{D}^{-1}$ shows the opposite trend, that is, performs well when the degree is less than 2 but consistently poorly as the degree increases. For the symmetric normalized filter $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, it performs balanced no matter when the item degree is small or large. This phenomenon is consistent with our analysis in Section 4.5 that $\mathbf{D}^{-1}\mathbf{A}$ tends to strengthen the high-degree nodes and $\mathbf{A}\mathbf{D}^{-1}$ favors the low-degree nodes, while $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the tradeoff between the two filters. Nonetheless,

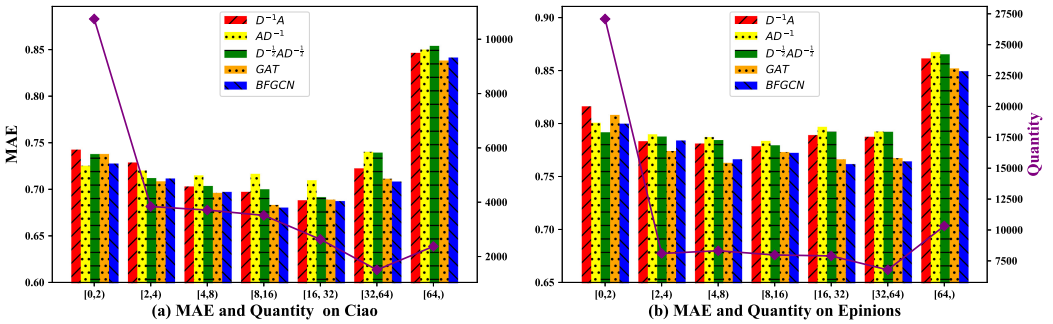

Fig. 7. MAE and Quantity under different groups of observed items (horizontal axis) on the Ciao and Epinions datasets, where the horizontal axis represents the number of observed items in the respective training set.

Table 6. Evaluation Results of Different Methods on Social Recommendation with Multi-typed Relationships

	MMORPG			
	HR@5	HR@10	MRR@5	MRR@10
DiffNet	0.2641	0.3835	0.1519	0.1667
BFHAN ($r = 1$)	0.5764	0.6463	0.4872	0.5084
RGCN	0.5985	0.6255	0.4969	0.5138
HAN	0.6217	0.6549	0.5063	0.5273
BFHAN (sym)	0.5977	0.6467	0.4897	0.5018
BFHAN ($\alpha = \beta = 0.5$)	0.6125	0.6376	0.5162	0.5261
BFHAN (mean)	0.6181	0.6687	0.5380	0.5465
BFHAN	**0.6297**	**0.6765**	**0.5408**	**0.5482**

our proposed BFGCN, which combines the above two filters, can perform better than each of the above graph convolutional filters in almost all item groups. Meanwhile, BFGCN has competitive performances compared to GAT in almost all sparsity levels.

5.2.3 Guild Recommendation with Multi-typed Relationships. To study whether considering multi-typed relationships can improve the performance of recommendation, we experiment with the BFHAN for guild recommendation in the MMORPG dataset. We first ignore the types of relationships between users and experiment with DiffNet and BFHAN ($r = 1$). Then, we use RGCN, HAN, and BFHAN to leverage all types of relationships. In particular, we add a player-guild-player meta-path, i.e., two players belong to the same guild, for fair comparisons with RGCN and HAN. It is worth mentioning that we set $K = 1$ both in BFHAN ($r = 1$) and BFHAN. In addition, we conduct experiments with other variants, such as BFHAN (sym), BFHAN ($\alpha = \beta = 0.5$), and BFHAN (mean). As shown in Table 6, we can observe that BFHAN ($r = 1$) outperforms DiffNet, but is slightly worse than HAN. Further, we find that BFHAN exceeds all the other baseline methods, i.e., RGCN and HAN, when all types of relationships are considered. The results of ablation experiments also manifest that the BFGCN can outperform the symmetric normalized filter, and multiplex attention network is more capable of capturing different influences of neighboring users with heterogeneous relationships compared to mean pooling. Overall, the recommendation performance can be improved by leveraging multiple types of relationships between users when multi-typed relationships can complement each other.

Table 7. Effect of Different Propagation Layers (L) on the MMORPG Dataset

	HR@5	HR@10	MRR@5	MRR@10
BFHAN ($L = 0$)	0.1470	0.2252	0.0785	0.0869
BFHAN ($L = 1$)	0.5024	0.5568	0.4078	0.4268
BFHAN ($L = 2$)	**0.6297**	**0.6765**	**0.5408**	**0.5482**
BFHAN ($L = 3$)	0.6044	0.6559	0.5032	0.5250

Table 8. Performance Comparison (F1-score) of Different Methods on the Node Classification Task

	ACM	DBLP	IMDB
GAT	0.8741	0.9100	0.6091
RGCN	0.8965	0.9264	0.5855
HAN	0.8876	**0.9364**	0.5992
$BFHAN_{-bn}$ (sym)	0.8929	0.9198	0.6076
$BFHAN_{-bn}$ ($\beta = 0.5$)	0.8984	0.9259	0.5743
$BFHAN_{-bn}$(mean)	0.8915	0.9187	0.6152
$BFHAN_{-bn}$	**0.9085**	0.9313	**0.6192**

We also study the influence w.r.t. propagation layer number L in the user-user multiplex network on the model performance. As summarized in Table 7, we can find that leveraging graph convolutions in the user-user networks can greatly improve the accuracy of recommendation, as $L = 0$ means that the social networks are ignored. Furthermore, the model performance increases first and then decreases as the number of propagation layers increases, which indicates that users are more influenced by nearby users, while remote users have fewer impacts, so stacking too many layers may lead to model overfitting.

5.2.4 Node Classification with Multi-typed Relationships. To further validate the effect of our proposed method on node representation learning in the heterogeneous network, we conduct node classification task for further studies. We use the same datasets and experimental settings as Reference [37]. Since the benchmark datasets have already predefined meta-paths, we remove the bipartite network model part, denoted as $BFHAN_{-bn}$, for fair comparisons with GAT, RGCN, and HAN. It is worth noting that we perform GAT in the superimposed single-layer network and we set $L = 1$ in $BFHAN_{-bn}$. From Table 8, we can find that $BFHAN_{-bn}$ performs better than all the baselines in the two datasets, i.e., 2.35% F1-score improvement for ACM and 3.34% F1-score improvement for IMDB than HAN. To verify the impacts of each model component, we conduct the same ablation experiments as described above. The results in Table 8 validate that the performance of the proposed BFGCN is better than the symmetric or average normalized filters. Meanwhile, using the multiplex attention network to fuse the node representations of each relationship type is better than mean pooling.

We also study the effect of propagation depth in the three datasets. As shown in Table 9, leveraging graph convolutional network is beneficial to node representations, as $L = 0$ means that the relationships between the nodes are not considered. However, the performance declines after stacking many graph convolutional layers. This may be because stacking many layers will bring the noise nodes, while the existing graph convolutional filters cannot effectively remove them and we leave it to future work.

Table 9. Effect of Different Propagation Layers (L) in the Multiplex Network

	ACM	DBLP	IMDB
$BFHAN_{-bn}$ ($L = 0$)	0.8138	0.7780	0.5080
$BFHAN_{-bn}$ ($L = 1$)	**0.9085**	**0.9313**	**0.6192**
$BFHAN_{-bn}$ ($L = 2$)	0.9006	0.9129	0.5960
$BFHAN_{-bn}$ ($L = 3$)	0.8026	0.8446	0.5699

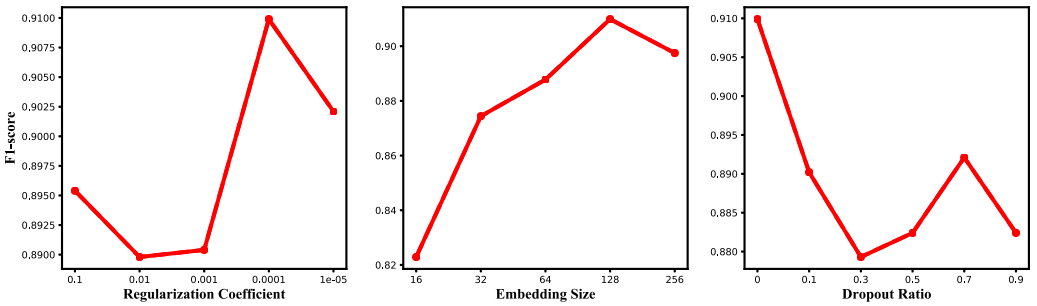

Fig. 8. Parameter sensitivity w.r.t. regularization coefficient, embedding size, and dropout ratio on the ACM dataset.

Finally, we study the sensitivity of parameters w.r.t. regularization coefficient, embedding size, and dropout ratio on the ACM dataset and report the results of F1-score in Figure 8. We can observe that as the regularization coefficient decreases to 0.0001, the performance w.r.t. F1-score gets the best. As for embedding size, we can find that with the growth of the embedding size, the performance raises first and then descends and the best embedding size is around 128. For the dropout ratio, we find that when the dropout ratio equals 0, the performance is the best. In all, embedding size has a relatively large influence on the performance.

6 CONCLUSION AND FUTURE WORK

In this work, we propose the BFHAN, which can improve node representations of the power-law networks and handle the situation when there are multiple types of relationships between user nodes. The whole BFHAN consists of two components: one is the Bilateral Filtering Graph Convolutional Network, to adaptively adjust the aggregation weights of the neighboring representations according to node degrees; another is the Multiplex Attention Network, to capture different influences from multi-typed relationships between user nodes. We further theoretically analyze the relationship between the convergence values of existing graph convolutional filters and node degrees. Based on the analysis, we explain the advantages of our proposed BFGCN in terms of long-tail node representation learning compared with existing filters. Empirically, we have designed three different tasks to verify the effectiveness of the above two modules, respectively. The experimental results demonstrate that the entire framework BFHAN significantly outperforms several state-of-the-art GCN-based methods for social recommendation tasks. At the same time, ablation experiments also verify the effectiveness of each component.

In the future, we would like to explore the influences of different graph convolutional filters more deeply and a better strategy to combine them to further improve the representation learning for the long-tail nodes. Meanwhile, as a supplement to our work, we also would like to consider multiplex interactions between users and items, e.g., clicks, purchase, and add-to-cart, to fully capture the interests of users.

REFERENCES

[1] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. In *Recommender Systems Handbook*. Springer, 1–35.

[2] Linyuan Lü, Matúš Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou. 2012. Recommender systems. *Phys. Rep.* 519, 1 (2012), 1–49.

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

[4] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2021. A survey on neural recommendation: From collaborative filtering to content and context enriched recommendation. *arXiv preprint arXiv:2104.13030* (2021).

[5] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations*.

[6] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.

[7] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2021. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 32, 1 (2021), 4–24.

[8] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[9] 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 *Proceedings of the European Semantic Web Conference*. Springer, 593–607.

[10] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2001–2009.

[11] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=rJXMpikCZ.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 6000–6010.

[13] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 235–244.

[14] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *Proceedings of the Web Conference*. ACM, 417–426.

[15] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Peng He, Paul Weng, Han Gao, and Guihai Chen. 2019. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In *Proceedings of the Web Conference*. ACM, 2091–2102.

[16] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439 (1999), 509–512.

[17] Yoon-Joo Park and Alexander Tuzhilin. 2008. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2nd ACM Conference on Recommender Systems*. 11–18.

[18] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. 2012. Challenging the long tail recommendation. *Proc. VLDB Endow.* 5, 9 (2012), 896–907.

[19] Mingxin Gan and Rui Jiang. 2013. Improving accuracy and diversity of personalized recommendation through power law adjustments of user similarities. *Decis. Supp. Syst.* 55, 3 (2013), 811–821.

[20] Dominik Kowald, Markus Schedl, and Elisabeth Lex. 2020. The unfairness of popularity bias in music recommendation: A reproducibility study. In *Proceedings of the European Conference on Information Retrieval*. Springer, 35–42.

[21] Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. 2008. SoRec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. ACM, 931–940.

[22] Jingjing Li, Ke Lu, Zi Huang, and Heng Tao Shen. 2019. On both cold-start and long-tail recommendation with social data. *IEEE Trans. Knowl. Data Eng.* 33, 1 (2019), 194–208.

[23] Adit Krishnan, Ashish Sharma, Aravind Sankar, and Hari Sundaram. 2018. An adversarial approach to improve long-tail performance in neural collaborative filtering. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1491–1494.

[24] Siyi Liu and Yujia Zheng. 2020. Long-tail session-based recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 509–514.

[25] Aravind Sankar, Junting Wang, Adit Krishnan, and Hari Sundaram. 2020. Beyond localized graph neural networks: An attributed motif regularization framework. *arXiv preprint arXiv:2009.05197* (2020).
[26] Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Yiqi Wang, Jiliang Tang, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. 2020. Investigating and mitigating degree-related biases in graph convoltuional networks. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. 1435–1444.
[27] Adit Krishnan, Ashish Sharma, and Hari Sundaram. 2018. Insights from the long-tail: Learning latent representations of online user behavior in the presence of skew and sparsity. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 297–306.
[28] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. 2010. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proc. Nat. Acad. Sci.* 107, 10 (2010), 4511–4515.
[29] Meng Qu, Jian Tang, Jingbo Shang, Xiang Ren, Ming Zhang, and Jiawei Han. 2017. An attention-based collaboration framework for multi-view network representation learning. In *Proceedings of the ACM on Conference on Information and Knowledge Management*. ACM, 1767–1776.
[30] Hongming Zhang, Liwei Qiu, Lingling Yi, and Yangqiu Song. 2018. Scalable multiplex network embedding. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, Vol. 18. 3082–3088.
[31] Jun Zhao, Zhou Zhou, Ziyu Guan, Wei Zhao, Wei Ning, Guang Qiu, and Xiaofei He. 2019. IntentGC: A scalable graph convolution framework fusing heterogeneous information for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2347–2357.
[32] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[33] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. In *Proceedings of the International Conference on Learning Representations*.
[34] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 32, 1 (2020), 4–24. DOI: 10.1109/TNNLS.2020.2978386
[35] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).
[36] 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.
[37] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *Proceedings of the Web Conference*. ACM, 2022–2032.
[38] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. 2020. Heterogeneous network representation learning: A unified framework with survey and benchmark. *IEEE Trans. Knowl. Data Eng.* (2020), 1–1. DOI: 10.1109/TKDE.2020.3045924
[39] Le Wu, Yong Ge, Qi Liu, Enhong Chen, Richang Hong, Junping Du, and Meng Wang. 2017. Modeling the evolution of users' preferences and social links in social networking services. *IEEE Trans. Knowl. Data Eng.* 29, 6 (2017), 1240–1253.
[40] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*. 1257–1264.
[41] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM, 135–142.
[42] Dayou Liu Jiming Liu Bo Yang, Yu Lei. 2013. Social collaborative filtering by trust. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*. 2747–2753.
[43] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2015. TrustSVD: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29.
[44] Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 426–434.
[45] Chun Yi Liu, Chuan Zhou, Jia Wu, Yue Hu, and Li Guo. 2018. Social recommendation with an essential preference space. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. 346–353.
[46] Xiwang Yang, Yang Guo, Yong Liu, and Harald Steck. 2014. A survey of collaborative filtering based social recommender systems. *Comput. Commun.* 41 (2014), 1–10.
[47] Le Wu, Junwei Li, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. 2020. DiffNet++: A neural influence and interest diffusion network for social recommendation. *IEEE Trans. Knowl. Data Eng.* (2020), 1–1. DOI: 10.1109/TKDE.2020.3048414
[48] M. Vijaikumar, Shirish Shevade, and M. Narasimha Murty. 2019. SoRecGAT: Leveraging graph attention mechanism for top-N social recommendation. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 430–446.

[49] Wenqi Fan, Tyler Derr, Yao Ma, Jianping Wang, Jiliang Tang, and Qing Li. 2019. Deep adversarial social recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 1351–1357.

[50] Adit Krishnan, Hari Cheruvu, Cheng Tao, and Hari Sundaram. 2019. A modular adversarial approach to social recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1753–1762.

[51] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 1263–1272.

[52] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[53] David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Appl. Comput. Harmon. Anal.* 30, 2 (2011), 129–150.

[54] Jian-Guo Liu, Tao Zhou, and Qiang Guo. 2011. Information filtering via biased heat conduction. *Phys. Rev. E* 84, 3 (2011), 037101.

[55] Tianrun Gao, Yuexia Zhang, Xuzhen Zhu, and Lihua Li. 2017. Personalized recommendation based on unbalanced symmetrical mass diffusion. In *Proceedings of the IEEE 3rd International Conference on Multimedia Big Data*. IEEE, 384–388.

[56] Haibo He and Edwardo A. Garcia. 2009. Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.* 21, 9 (2009), 1263–1284.

[57] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*. 2980–2988.

[58] John Boaz Lee, Ryan A. Rossi, Sungchul Kim, Nesreen K. Ahmed, and Eunyee Koh. 2019. Attention models in graphs: A survey. *ACM Trans. Knowl. Discov. Data* 13, 6 (2019), 1–25.

[59] Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, Wei Liu, and Tat-Seng Chua. 2017. SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5659–5667.

[60] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7132–7141.

[61] Sheldon M. Ross, John J. Kelly, Roger J. Sullivan, William James Perry, Donald Mercer, Ruth M. Davis, Thomas Dell Washburn, Earl V. Sager, Joseph B. Boyce, and Vincent L. Bristow. 1996. *Stochastic Processes*. Vol. 2. Wiley New York.

[62] Meng Liu, Hongyang Gao, and Shuiwang Ji. 2020. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 338–348.

[63] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th Symposium on Operating Systems Design and Implementation*. 265–283.

[64] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. 249–256.

[65] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

[66] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1 (2014), 1929–1958.

Received November 2020; revised April 2021; accepted June 2021