TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 14/29 pp1617–1635
DOI: 10.26599/TST.2024.9010048
Volume 30, Number 4, August 2025

Graph Structure Learning for Robust Recommendation

Lei Sang, Hang Yuan, Yuee Huang, and Yiwen Zhang*

Abstract: Recommendation systems play a crucial role in uncovering concealed interactions among users and items within online social networks. Recently, Graph Neural Network (GNN)-based recommendation systems exploit higher-order interactions within the user-item interaction graph, demonstrating cutting-edge performance in recommendation tasks. However, GNN-based recommendation models are susceptible to different types of noise attacks, such as deliberate perturbations or false clicks. These attacks propagate through the graph and adversely affect the robustness of recommendation results. Conventional two-stage method that purifies the graph before training the GNN model is suboptimal. To strengthen the model's resilience to noise, we propose Graph Structure Learning for Robust Recommendation (GSLRRec), a joint learning framework that integrates graph structure learning and GNN model training for recommendation. Specifically, GSLRRec considers the graph adjacency matrix as adjustable parameters, and simultaneously optimizes both the graph structure and the representations of user/item nodes for recommendation. During the joint training process, the graph structure learning employs low-rank and sparse constraints to effectively denoise the graph. Our experiments illustrate that the simultaneous learning of both structure and GNN parameters can provide more robust recommendation results under various noise levels.

Key words: robust recommendation; Graph Neural Network (GNN); Graph Structure Learning (GSL)

1 Introduction

The proliferation of e-commerce and social media platforms has led to an enormous amount of information on the Internet. While these information has facilitated people's daily activities, it also has posed the challenge of "information overload" problem. Consequently, people have to spend significant time and effort to find valuable information[1]. Traditional search engine technology cannot provide adequate personalized services to different users. Therefore, recommendation systems become essential. By analyzing users' historical behavior, recommendation systems can infer their hidden preferences and filter out irrelevant content, recommending content that they may find appealing. Over time, recommendation systems have undergone various technical advancements[2]. The most common traditional recommendation systems include content-based and collaborative filtering-based recommendation models[3]. However, both of these methods have some limitations: (1) Collaborative filtering based methods face the challenge of sparse data, where user-item interactions are sparse, leading to inaccurate recommendations. Moreover, collaborative filtering based methods also encounter the cold start issue arising from a lack of historical interactions;

• Lei Sang, Hang Yuan, and Yiwen Zhang are with School of Computer Science and Technology, Anhui University, Hefei 230601, China. E-mail: sanglei@ahu.edu.cn; yuanhang@stu.ahu.edu.cn; zhangyiwen@ahu.edu.cn.
• Yuee Huang is with School of Public Health, Wannan Medical College, Wuhu 241002, China. E-mail: huangyewindow@163.com.
* To whom correspondence should be addressed.
Manuscript received: 2023-12-19; revised: 2024-02-26; accepted: 2024-03-01

(2) Content-based methods rely solely on specific data analysis to assess item relevance, making it to be a change to directly capture users' overall interests[4].

In recent times, the progress in Graph Neural Network (GNN) technology has laid a robust foundation and created opportunities to tackle the aforementioned challenges in recommender systems[5–7]. GNN has been highly successful in various domains, such as relationship extraction, protein structure prediction, and knowledge mapping. The core concept of GNN revolves around iteratively aggregating feature information from neighboring nodes, and combine them with the representation of the current central node[8, 9]. Unlike traditional recommendation systems that only consider first-order direct connectivity between user and item, GNN-based recommendation systems can capture higher-order interactions within the user-item interaction graph, enabling the acquisition of more optimal representations for users and items in the context of recommendation[10–12]. GNN applies iterative propagation operations to each node in the graph, wherein it learns target node embeddings through the aggregation of local neighbors' contextual features from the previous layer. Through the stacking of multiple layers of propagation operations, GNN effectively explores high-order interactions among users and items, producing high-quality embeddings for items or users in the context of recommendation. For instance, Neural Graph Collaborative Filtering (NGCF) leverages high-order interactivity in graph data to obtain more optimal embedded representations of users and items. The model then performs an inner integration operation on them to obtain the result, which outperforms previous methods. LightGCN model argue that preserving all the details in the graph convolution network of the NGCF model is unnecessary. As a consequence, He et al.[7] removed the non-linear activation function and feature transformation matrix, resulting in a simpler and more effective model.

Despite promising performance, however, GNN-based recommendation systems are susceptible to various imperceptible noise attacks on the graph structure, including deliberate perturbations or false clicks[13–15]. Initially, numerous malicious users seek to manipulate recommendation system by introducing deliberate and anomalous perturbations with the aim of boosting item sales, increasing traffic, or even disrupting the system to render it ineffective. For instance, these users might create fake profiles by establishing connections with targeted users or connecting to fake users to mislead neural graph based recommendation models. Similarly, connecting fake items to target users, or making malicious changes to existing items, can also fool the neural graph based recommendation models. Secondly, under the implicit feedback context, the recommendation system is unable to identify false clicking behaviors of users, which can also introduce bias unwanted edge into the recommendation graph[16–20]. For instance, within the realm of e-commerce, a substantial proportion of clicks may not result in actual purchases, and a significant number of purchases may culminate in negative reviews. These noises from deliberate perturbations or false clicks could inflict more severe damage to the training of GNNs than the standard neural networks, primarily attributable to the distinctive propagation process inherent in GNNs. Due to GNNs computing node embeddings through recursive aggregation of information from neighborhoods, this iterative mechanism induces cascading effects. In this process, minor noise in a graph propagates to neighboring nodes at a significant distance, consequently influencing the embeddings of many others[21, 22].

For instance, in Fig. 1, deliberate perturbations are

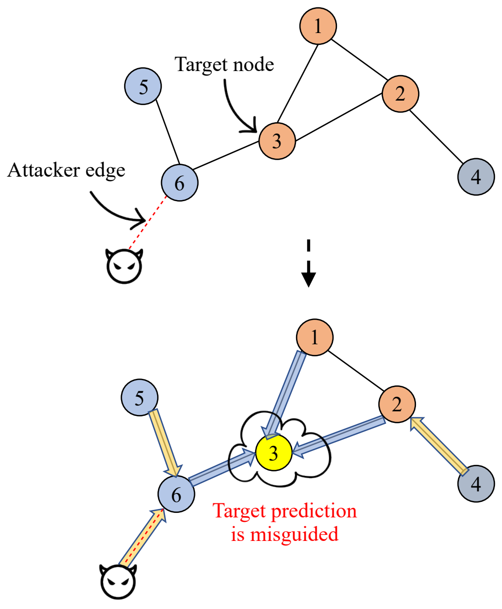

Fig. 1 Illustration that applying perturbation to graph structure can easily misguide the target prediction. We deliberately add perturbation edge to Node 6. Consequently, the neural graph based model is misled into generating inaccurate predictions for the neighboring Node 3, primarily as a consequence of the cascading amplification effect.

intentionally introduced to the edge connected to Node 6. Consequently, the neural graph based model is deceived into generating inaccurate predictions for the neighboring Node 3, primarily due to the cascading amplification effect. Hence, GNN-based recommendation systems exhibit high sensitivity to the quality of the provided data and typically demand an optimal graph structure for effectively learning informative representations. In a realistic recommendation scenario, minor noises may frequently occur, but should not significantly affect predictions. Therefore, stabilizing the training process of GNN-based recommendation systems is crucial.

To tackle the robustness problem of recommendation systems, researchers have investigated numerous solutions. Robust recommendation seeks to ensure that recommendation models can withstand the challenges posed by various imperfect environments. A prevalent solution for achieving robust recommendation is the two-stage method. In this approach, the attacked graph is initially purified by removing adversarial edges and restoring any deleted edges. Subsequently, the downstream recommendation model is trained on the purified graph[23–25]. For example, Unorganized Malicious Attack (UMA) detection algorithm aims to detect unorganized malicious edge attacks from online rating systems, which can be seen as a variant of matrix completion to recover the noise user-item bipartite graph[26, 27]. However, two-stage method has some limitations. It separates the graph structure purification from the model training, which may cause information loss and suboptimal performance. Besides, without the guidance of supervised GNN recommendation signals, the graph structure purification process may unintentionally eliminate normal edges, hindering the system from achieving optimal performance[28–30]. Furthermore, current research on robust recommendation has yet to address the cascading amplification effect resulting from the iterative propagation of anomalous attacks within the graph structure. This phenomenon can lead to the widespread dissemination of anomalous attacks to a large number of nodes, significantly affecting the overall recommendation performance.

In this paper, we address the robustness limitations of the existing GNN-based recommender systems by proposing Graph Structure Learning for Robust Recommendation (GSLRRec), a novel framework that jointly learns a clean graph structure from a noisy graph and the corresponding user/item node representations for recommendation. Graph Structure Learning (GSL) aims to improve the robustness of GNN-based recommendation by producing a denoised graph structure for learning user and item embeddings. A key challenge of GSL is how to remove adversarial edges and restore deleted edges based on some criteria. Inspired by the commonalities exhibited by graphs in the real world, we identify low-rankness and sparsity as two key properties that can be utilized as constraints in the process of GSL[31]. Based on this, we design the GSLRRec model, which incorporates GSL into GNNs. In particular, GSLRRec considers the graph adjacency matrix as learnable parameters, and concurrently optimizes both the graph structure and the representations of user/item nodes for recommendation. Comprehensive experiments conducted on two benchmarks illustrate that GSLRRec consistently surpasses the baseline model across various levels of noise attacks.

Our contributions can be summarized as follows:

• GSLRRec framework: The paper introduces GSLRRec, a novel framework that integrates GSL into GNN-based recommendation systems, enabling simultaneous learning of refined graph structures and user/item node representations.

• Graph property constraints: By leveraging real-world graph properties, like low-rankness and sparsity, GSLRRec enhances robustness by effectively refining graph structures through the GSL process, leading to improved accuracy in recommendation.

• Strong experimental validation: Through rigorous experiments on benchmark datasets, GSLRRec's performance superiority over baseline models is consistently demonstrated, affirming its efficacy in achieving robust and accurate recommendations across various noise levels.

2 Preliminary

In this section, we initially present the foundation of the GNN-based recommendation. Following that, we outline the overarching concept of incorporating graph structure learning into the training process of GNN-based recommendation.

2.1 GNN-based recommendation

Let $\mathcal{G} = (V, E)$ be the input graph with node set V and edge set E, where V contains two types of nodes representing users and items, and E contains an edge of one type representing whether users and items have interaction before. Let the user-item interaction matrix be $\boldsymbol{S} \in \mathcal{S}^{M\times N}$, where $\mathcal{S}$ is a matrix set, representing all matrices of size $M \times N$, and M and N indicate the number of users and items, respectively. if user u has interacted with item i, S_{ui}=1, else S_{ui}=0. The adjacency matrix of the user-item interaction graph is represented as

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{0} & \boldsymbol{S} \\ \boldsymbol{S}^{\mathrm{T}} & \boldsymbol{0} \end{pmatrix} \tag{1}$$

with the input graph data, the embedding layer acquires low-dimensional hidden representations of nodes. These representations can effectively capture the intrinsic properties of users and items[32].

We denote a user u (an item i) with an embedding vector $e_u \in \mathcal{S}^d$ ($e_i \in \mathbf{R}^d$), where d is the size of embedding. This can be conceptualized as transformation matrices that individually map the user or item into the hidden space, $\boldsymbol{E} = [e_u; e_i] \in \mathbf{R}^{(M+N)\times d}$.

A GNN encoder GNN $(\boldsymbol{A}, \boldsymbol{E})$, parameterized by θ, receives the graph structure as input, resulting in a final node embedded representation $\boldsymbol{Z}$ = GNN $(\boldsymbol{A}, \boldsymbol{E})$ and feeding into downstream recommendation tasks. In the GNN recommendation system section of the model, we opt to forgo the use of feature transformation and nonlinear activation in traditional Graph Convolution Networks (GCNs). The $e_u^{(0)}$ and $e_i^{(0)}$ represent the initial high-dimensional embedded representation of users and items, respectively, which leverage the user-item interaction graph to propagate embeddings as

$$e_u^{(l+1)} = \sum_{i\in\mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} e_i^{(l)},$$
$$e_i^{(l+1)} = \sum_{u\in\mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} e_u^{(l)} \tag{2}$$

where $e_u^{(l)}$ and $e_i^{(l)}$ represent the embedded representation of user u and item i after l-layer propagation, respectively, $e_u^{(l+1)}$ and $e_i^{(l+1)}$ represent the embedded representation of user u and item i after (l+1)-layer propagation, $\mathcal{N}_u$ is the collection of items interacting with user u, and $\mathcal{N}_i$ is the collection of users interacting with item i. Here we use common processing methods in GNNs to perform iterative calculations through matrices, and combine $e_u^{(0)}$, $e_u^{(1)}, \ldots$ together to represent $\boldsymbol{E}$, which is the user matrix. We also do the same for the item matrix. By propagating L layers, we can calculate a simple weighted mean to get the final representation,

$$\boldsymbol{Z} = \frac{1}{L+1}\left(\boldsymbol{E}^{(0)} + \boldsymbol{E}^{(1)} + \boldsymbol{E}^{(2)} + \cdots + \boldsymbol{E}^{(L)}\right) \tag{3}$$

from which we can get $z_u = e_u^{(L)}$ and $z_i = e_i^{(L)}$ for users and items, respectively, then we use inner product to generate the prediction score as

$$\hat{y}_{ui} = z_u^{\mathrm{T}} z_i \tag{4}$$

where $\hat{y}_{ui}$ represents the predicted score of user u and item i. The matrix form of this recommendation model is as follows. Let the l-th layer embedding matrix be $\boldsymbol{E}^{(l)}$. We can obtain the matrix equivalent form of GNN model,

$$\boldsymbol{E}^{(l+1)} = \left(\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\right)\boldsymbol{E}^{(l)} \tag{5}$$

where $\boldsymbol{D}$ is an $(M+N)\times(M+N)$ diagonal matrix. Then we get the final embedding matrix used for model prediction as Eq. (3). Finally, we recommend the corresponding item i to u according to the score ranking of all items corresponding to each user with the help of $\hat{y}_{ui} = z_u^{\mathrm{T}} z_i$.

We take the Bayesian Personalized Ranking (BPR) as the loss function,

$$\mathcal{L}_{\text{rec}}\ (\boldsymbol{A}, \boldsymbol{Y}; \theta) =$$
$$-\sum_{u=1}^{M}\sum_{i\in\mathcal{N}_u}\sum_{j\notin\mathcal{N}_u} \ln\sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right) + \lambda\left\|\boldsymbol{E}^{(0)}\right\|^2 \tag{6}$$

where $\mathcal{L}_{\text{rec}}(\cdot)$ represents BPR loss function, λ indicates the L_2 regularization strength, σ is the activation function, and $\boldsymbol{Y}$ stands for ground truth, which is the real set of user-item interactions. The core idea of BPR loss is that the appeared items are more important than items that do not appear. It worth noting that the only trainable model parameters are the user and item embeddings at the 0-th layer: $\theta = \boldsymbol{E}^{(0)}$. In summary, the functionality of our framework is as follows: Input the user-item adjacency matrix that has been subjected to adversarial attack, obtain a clean adjacency matrix through graph structure learning and the GNN model parameters that can improve the recommendation performance at the same time.

2.2 GSL for GNN-based recommendation

Given an input graph $\boldsymbol{A}$, the goal of GSL is to learn a purified clean adjacency matrix P, as well as corresponding node embeddings $\boldsymbol{Z}^{\star}$ = GNN (θ, $\boldsymbol{P}$). The essence of a GSL is a learning graph representation that typically consists of three parts, including structure modeling, message propagation, and learning objectives[31].

Structure modeling. The essence of GSL lies in an encoding function that constructs an optimal graph structure $\boldsymbol{P}$ from original graph $\boldsymbol{A}$ regularised by intrinsic graph structure properties. The formula is as follows:

$$\mathcal{L}_{\text{reg}}(\boldsymbol{P},\boldsymbol{A}) \tag{7}$$

where $\mathcal{L}_{\text{reg}}(\cdot)$ denotes regularization loss function, and the new graph adjacency matrix $\boldsymbol{P}$ is treated as learnable parameters.

Message propagation. After learning to optimize the graph structure $\boldsymbol{P}$, the node features the propagate to the refined field. After several iterative training sessions of the GNN model, each node can aggregate the information of all neighbor nodes, obtaining a better embedded representation $\boldsymbol{Z}^{\star}=\text{GNN}(\theta, \boldsymbol{P})$, considering the complexity of the GSL part of the model, we use constraints, such as low rank and sparsity, as stopping criteria for iterative training.

Learning objectives. Our goal is not only to obtain a clean graph structure, but more importantly, to achieve greater accuracy in downstream recommendation tasks. Therefore, our learning objective consists of two components:

$$\mathcal{L}=\mathcal{L}_{\text{reg}}(\boldsymbol{P},\boldsymbol{A})+\varphi\mathcal{L}_{\text{rec}}(\boldsymbol{P},\boldsymbol{Y},\theta) \tag{8}$$

where φ is a hyperparameter whose function is to balance the weights of the two parts.

3 Proposed Model

GNN-based recommender systems are vulnerable to noise attacks that can degrade their performance significantly. In these systems, each node's representation depends on its neighbors, and this affects the global graph structure during the propagation process of GNN training. Attackers can manipulate the interactions between users and items by (1) adding edges between non-interacting pairs, or (2) deleting edges between interacting pairs. To mitigate the impact of such attacks, we propose to restore the attacked graph to its original or a close-to-original structure by exploiting the sparsity and low-rank properties of real-world graphs. Inspired by Pro-GNN[31], we impose these two constraints on each epoch of GSL and use the recovered graph to train better parameters for downstream GNN models. Figure 2 illustrates our proposed GSLRRec model. The top-left part of the figure shows the GSL section, which includes low-rank constraint and sparse constraint. The low-rank constraint can eliminate noise from the graph and retain useful information. The sparse constraint can also remove noise from the graph. The right part of the figure depicts the GNN model training section, which adopts a three-layer

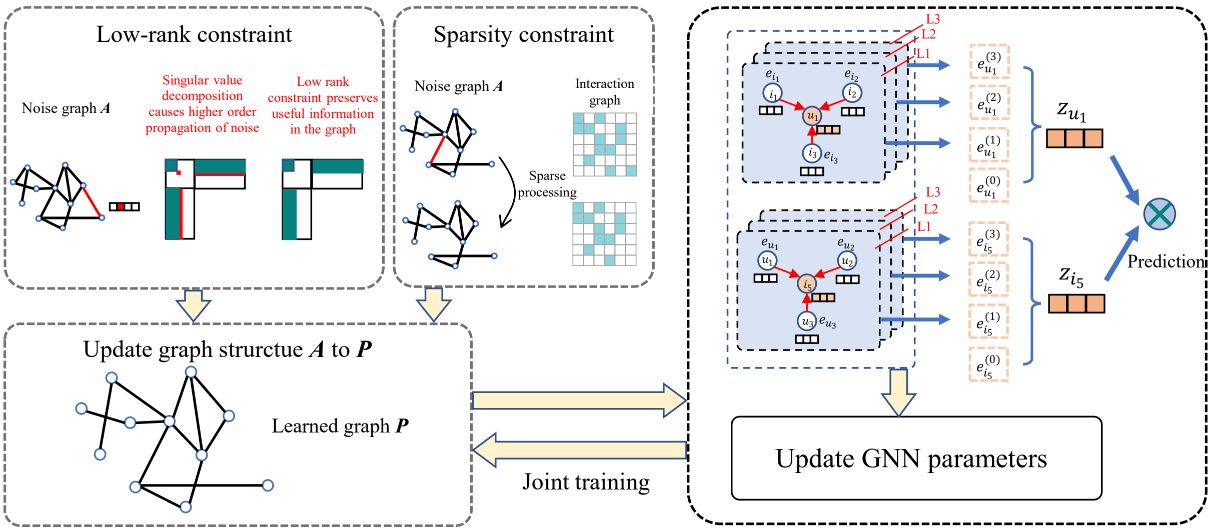

Fig. 2 The proposed GSLRRec model. GSLRRec is a joint learning framework that integrates graph structure learning and GNN model training for recommendation. Indeed, GSLRRec specifically treats the graph adjacency matrix as learnable parameters, engaging in the joint optimization of both the graph structure and node representations for recommendation. During the joint training process, the graph structure learning employs low-rank and sparse constraints to effectively denoise the graph.

neighborhood aggregation mechanism. Taking user u_1 as an example, in the L1 layer, nodes i_1, i_2, i_3, etc., interacting with u_1 are aggregated to represent u_1, obtaining $e_{u_1}^{(1)}$. Similarly, in the L2 and L3 layers, $e_{u_1}^{(2)}$ and $e_{u_1}^{(3)}$ are obtained, respectively. These representations are then weighted summed with the initial $e_{u_1}^{(0)}$ to obtain the final representation of u_1, denoted as $\boldsymbol{Z}_{u_1}$. Similarly, for item i_5, the final representation $\boldsymbol{Z}_{i_5}$ is obtained. Then, the dot product of $\boldsymbol{Z}_{u_1}$ and $\boldsymbol{Z}_{i_5}$ is computed to obtain the prediction score, and this part is trained using the BPR loss. We adopt a joint training approach, jointly conducting GSL and GNN model training. The nuclear norm is used as the indicator for the low-rank constraint, and the l_1 norm is used as the indicator for the sparse constraint. $\boldsymbol{P}$ is updated jointly, and then fed into the GNN model for training. This process is repeated in each epoch.

Next, we first introduce the low-rank assumption of our model, followed by the sparsity assumption. Then we present our objective function and optimization algorithm.

3.1 Low rank of graph structure

Real-world graph structures often share common attributes, such as low rank[31]. This attribute is observed in many clean graphs, such as social networks, where most individuals are connected only to a few neighbors. Low rank matrix approximation has become a useful research tool in computer vision, image analysis, and other fields. In recommendation systems, user-item interaction bipartite graphs are often of low rank. Nevertheless, adversarial attacks have the capability to alter the low rank of the interaction graph. Empirical findings indicate that both targeted and non-targeted attacks can substantially modify the spectrum of graphs and elevate the rank of the adjacency matrix. Moreover, removing the noised edge decreases the rank of the graph matrix faster than removing the normal edge[33]. These findings provide strong evidence to support the use of low-rank constraints for repairing disturbed graphs, since they can effectively remove opposing edges. Repairing disturbed graphs with low-rank constraints minimizes the rank of the graph matrix,

$$\begin{aligned}&\min_{P}\ \mathrm{rank}\,(P),\\&\text{s.t., } \|A-P\|_{\mathrm{F}} \leqslant \phi\end{aligned} \tag{9}$$

where ϕ is the noise level, $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm, which is defined by $\|\boldsymbol{P}\|_{\mathrm{F}}^2 = \Sigma_{ij}\boldsymbol{P}_{ij}^2$.

Based on practical experience, minimizing rank is a non-convex problem that is NP-hard. A more manageable convex relaxation involves minimizing the sum of the matrix’s singular values, commonly referred to as the nuclear norm. The nuclear norm of a matrix is the sum of its singular values, constituting a convex function that can be efficiently optimized.

The rank of a matrix is a discrete integer representing the number of non-zero singular values in the matrix. In the context of the matrix rank minimization problem, the typically considered problem takes the following form:

$$\begin{aligned}&\min\ \mathrm{rank}\,(\boldsymbol{X}),\\&\text{s.t., } \mathcal{A}\,(\boldsymbol{X}) = \mathcal{B}\end{aligned} \tag{10}$$

where $\mathcal{A}$ and $\mathcal{B}$ are given linear transformations, and $\boldsymbol{X}$ is the matrix to be solved. This problem has been proven to be an NP-hard problem, meaning there are no known efficient algorithms that can solve it in polynomial time.

To overcome this problem, we introduce the nuclear norm, a continuous convex function that approximates the rank of a matrix. The nuclear norm is the sum of the singular values of a matrix and can be obtained through the matrix’s singular value decomposition. Specifically, for a matrix $\boldsymbol{X}$, its singular value decomposition is given by

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}} \tag{11}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices, Σ is a diagonal matrix with its diagonal elements representing the singular values of $\boldsymbol{X}$.

The nuclear norm is defined as the l_1 norm of the singular values,

$$\|\boldsymbol{X}\|_* = \sum_{q=1}^{\min(m,\,n)} \sigma_q \tag{12}$$

where $\|\cdot\|_*$ represents the nuclear norm, m and n are the number of rows and columns of matrix $\boldsymbol{X}$, respectively, and σ_q is the q-th singular value of $\boldsymbol{X}$.

In many cases, the nuclear norm of a matrix can serve as an approximation of its rank. Specifically, for any matrix $\boldsymbol{X}$, there exists an inequality relationship,

$$\mathrm{rank}\,(\boldsymbol{X}) \leqslant \|\boldsymbol{X}\|_* \leqslant \sqrt{\mathrm{rank}\,(\boldsymbol{X})\cdot\min\,(m,\,n)} \tag{13}$$

This relationship indicates that the nuclear norm serves as a compact and convex relaxation of the matrix rank. When the matrix $\boldsymbol{X}$ is of low rank, the value of the nuclear norm is close to the rank of the matrix. Therefore, by minimizing the nuclear norm, we can obtain a lower-rank matrix, effectively providing an approximate solution to the original problem of minimizing rank.

Previous studies have demonstrated that, in practice, nuclear norm constraints can yield solutions with low rank[34]. By substituting the rank function with the nuclear norm, we attain a rigorous convex relaxation of the rank minimization problem. The problem is formulated as follows:

$$\min_{P} \|\boldsymbol{P}\|_*, \quad \text{s.t., } \|\boldsymbol{A}-\boldsymbol{P}\|_{\mathrm{F}} \leqslant \phi \tag{14}$$

where $\|\boldsymbol{P}\|_* = \sum_i \rho_i$, and ρ is the i-th singular value of $\boldsymbol{P}$. In addition, malicious attackers aim to attack the graph structure while maximizing the concealment of their attack. Therefore, the added disturbance is not necessarily obvious. Based on the above two points, we define the objective function as

$$\arg\min_{\boldsymbol{P}\in\mathcal{P}} \mathcal{L}' = \frac{1}{2}\|\boldsymbol{A}-\boldsymbol{P}\|_{\mathrm{F}}^2 + \alpha\|\boldsymbol{P}\|_* \tag{15}$$

where $\mathcal{L}'$ represents the objective function, $\mathcal{P}$ indicates the set of matrices whose rank is lower than the matrix $\boldsymbol{P}$, $\|\boldsymbol{A}-\boldsymbol{P}\|_{\mathrm{F}}^2$ makes sure that the learned matrix is similar to the initial matrix $\boldsymbol{A}$, α is a hyperparameter that controls the contribution of the nuclear norm in the objective function.

3.2 Sparsity of graph structure

Most real-world recommender system datasets and their user-item interaction matrices, such as Movielens, Amazon, and Yelp, are highly sparse, with more than 90% of missing values. This means that only a few users interact with a few items. However, noise attacks can alter the sparsity of these matrices by adding or deleting edges between users and items. We aim to restore the original or a close-to-original graph structure by imposing sparsity on the attacked graph matrix[31]. The l_0 norm counts the number of non-zero elements in a vector or a matrix. It can theoretically induce sparsity on our binary user-item interaction matrix. However, optimizing the l_0 norm is NP-hard and impractical. Therefore, we use the l_1 norm, which sums up the absolute values of the elements in a vector (also known as lasso regularization). The l_1 norm is a convex relaxation of the l_0 norm and easier to optimize. Thus, we employ the l_1 norm as a sparsity constraint and formulate it as

$$\|\boldsymbol{P}\|_1 = \max_j \sum_{i=1}^{m} \left|p_{i,j}\right| \tag{16}$$

where $p_{i,j}$ represents the value of an element in $\boldsymbol{P}$ that is located at coordinate (i, j), and $\|\boldsymbol{P}\|_1$ denotes the maximum of the sum of the absolute values of the column vectors of the user-item interaction matrix. So we update the objective function as

$$\arg\min_{\boldsymbol{P}\in\mathcal{P}} \mathcal{L}' = \|\boldsymbol{A}-\boldsymbol{P}\|_{\mathrm{F}}^2 + \alpha\|\boldsymbol{P}\|_* + \beta\|\boldsymbol{P}\|_1 \tag{17}$$

where $\|\boldsymbol{P}\|_1$ denotes the l_1 norm of $\boldsymbol{P}$, and the β is a hyperparameter used to control the contribution of the l_1 norm in the objective function. It is worth noting that $\boldsymbol{A}-\boldsymbol{P}$ can be perceived as the disparity between the clean and attacked graphs following a series of perturbations. These perturbations involve the addition or removal of edges from the clean graph as a consequence of the attack.

3.3 Complete ojective function of GSLRRec

After obtaining the target function described in the previous sections, it is natural to optimize the attacked interaction matrix using this target function, and then use the optimized matrix to train the GNN recommendation model. However, dividing the algorithm into two separate steps may not necessarily yield optimal results. Therefore, we propose a joint training algorithm framework that simultaneously learns the clean user-item interaction graph and the GNN parameters. Based on previous experience, our joint algorithm outperforms the two-step approach. The final objective function we use is as follows:

$$\arg\min_{\boldsymbol{P}\in\mathcal{P}} \mathcal{L}' = \|\boldsymbol{A}-\boldsymbol{P}\|_{\mathrm{F}}^2 + \alpha\|\boldsymbol{P}\|_* + \beta\|\boldsymbol{P}\|_1 + \gamma\mathcal{L}_{\mathrm{rec}}(\boldsymbol{P}, \boldsymbol{Y}, \theta) \tag{18}$$

where γ is a hyperparameter that controls the contribution of the BPR loss in the objective function. It is worth mentioning that adversarial attacks aim to interfere with recommendation performance, which results in an increase in the BPR loss. By adding the BPR loss to the objective function, we can promote the process of graph structure learning.

3.4 Optimization

In this section, we provide the details of our algorithm for optimizing Eq. (18). Our algorithm aims to jointly train the the parameter of GNN θ (i.e., the user-item embedding representation matrix) and the clean interaction graph $\boldsymbol{P}$. However, due to the complexity of the objective function and the inconsistency of various indicators, optimizing θ and $\boldsymbol{P}$ simultaneously is challenging. To address this issue, we apply the idea of Alternating Direction Method of Multipliers (ADMM)[31, 35]. The key idea behind ADMM is to decompose the objective function into two or more pieces and optimize each piece separately while keeping the others fixed. This approach has been shown to be effective in balancing the optimization of θ and $\boldsymbol{P}$, leading to improved performance.

We first keep the parameter θ unchanged and only update $\boldsymbol{P}$. Since the l_1 norm and the nuclear norm are non-differentiable, we use Forward-Backward Splitting (FBS) method[36] to alternate a gradient descent step and a proximal step,

$$\boldsymbol{P}^{(k)} = \overbrace{\text{proximal}_{\eta R}\Big(\underbrace{\boldsymbol{P}^{(k-1)} - \mu \cdot \nabla_{\boldsymbol{P}} \mathcal{L}(\boldsymbol{P},\theta)}_{\text{Gradient descent step}}\Big)}^{\text{Proximal step}} \tag{19}$$

where $\mathcal{L}(\boldsymbol{P},\theta) = ||\boldsymbol{A}-\boldsymbol{P}||_{\text{F}}^2 + \mathcal{L}_{\text{rec}}(\boldsymbol{P},\theta)$, η is the learning rate, and proximal$(\cdot)$ is used in solveing non-differentiable optimization problems.

$||\boldsymbol{P}||_1$ in this paper can be represented as

$$\text{proximal}_{(\alpha,\, l_1)}(\boldsymbol{P}) = \text{sgn}(\boldsymbol{P}) \odot (|\boldsymbol{P}| - \alpha)_+ \tag{20}$$

where $()_+$ denotes the element-wise positive part of a matrix, sgn$(\boldsymbol{P})$ indicates the sign matrix of $\boldsymbol{P}$, and we use "$\odot$" to denote Hadamard product of matrices.

Similarly, optimizing the nuclear norm $||\boldsymbol{S}||_*$ as

$$\text{proximal}_{(\beta,\, *)}(\boldsymbol{P}) = \boldsymbol{U}\,\text{dia}\left((\delta_i - \beta)_+\right)_i \boldsymbol{V}^{\text{T}} \tag{21}$$

where $\boldsymbol{U}\,\text{dia}(\delta_1, \delta_2, \ldots, \delta_n)\boldsymbol{V}^{\text{T}}$ is the singular value decomposition of $\boldsymbol{P}$. Finally, we use tr$(\boldsymbol{P})$ to indicate the trace of matrix $\boldsymbol{P}$, i.e., tr$(\boldsymbol{P}) = \sum_i \boldsymbol{P}_{ii}$.

After updating the interaction graph $\boldsymbol{P}$, we update the user and item embedding representation matrix $\theta = \boldsymbol{E}^{(0)}$ by sending the updated interaction graph to the GNN recommendation model and training the parameter θ. This process can be implemented using the ordinary stochastic gradient descent algorithm. Initially, the user-item interaction matrix A and hyper-parameters are input, and the user-item embedding representation $\theta = \boldsymbol{E}^{(0)}$ is initialized randomly. The algorithm then enters the while loop section, where the matrix $\boldsymbol{P}$ is updated in three incremental steps. Furthermore, we set a threshold δ in the range of (0, 1) for the matrix learning part of the model. If $\boldsymbol{P}[i][j] \geqslant \delta$, we set $\boldsymbol{P}[i][j] = 1$, otherwise $\boldsymbol{P}[i][j] < \delta$, we set $\boldsymbol{P}[i][j]$=0. Once $\boldsymbol{P}$ is updated, the GNN recommendation model is trained for κ iterations. The model is trained iteratively until convergence is achieved, or until the maximum number of epochs is reached. Algorithm 1 illustrates the complete process of the algorithm.

4 Experiment

In this section, we outline the experimental settings and conduct a comparative analysis between our GSLRRec model and several baseline models using various datasets and attack methods. Additionally, we perform ablation studies to scrutinize the essential components of our model that contribute to enhancing robustness. Finally, we examine the effect of hyperparameters on the experimental results. Additionally, we employ the DeepRobust framework in our experiments[37].

4.1 Experimental settings

4.1.1 Baselines

To evaluate the effectiveness of our GSLRRec model, we conduct a comparative analysis with both classical and advanced GNN-based recommendation models using the same dataset. This selection of models for comparison demonstrates the versatility and consistency of our model over a wide range of advanced recommendation models. The following models are used as baselines for our comparative analysis.

- **ExpoMF**: Matrix Factorization (MF) finds broad applications within the domain of recommendation

Algorithm 1 Framework of GSLRRec

```
Input: User-item interaction matrix $A$; hyper-parameters
       $\alpha, \beta, \gamma, \kappa, l_r$, etc.
Output: Learned user-item interaction matrix $P$; GNN
        parameters $\theta$
1 Initialize $P$ to $A$;
2 Randomly initialize $\theta$;
3 while not converged do
4     $P \leftarrow P - \mu \cdot \nabla_P(||A-P||_F^2 + \gamma\mathcal{L}_{rec}(P, Y, \theta))$;
5     $P \leftarrow \text{proximal}_{(\beta,*)}(P)$;
6     $P \leftarrow \text{proximal}_{(\alpha,l_1)}(P)$;
7     for $i = 1$ to $\kappa$ do
8         $g \leftarrow \frac{\partial \mathcal{L}_{rec}(P, Y, \theta)}{\partial \theta}$;
9         $\theta \leftarrow \theta - l_r \times g$;
10 Return $P$ and $\theta$
```

systems. ExpoMF introduces two exposure probability parameter models based on popularity and content, which have been shown to outperform previous matrix factorization models[38].

• **CFGAN**: CFGAN introduces a recommendation framework grounded in collaborative filtering, leveraging the power of Generative Adversarial Network (GAN). To address the challenges faced by traditional GAN models in recommendation systems, this framework proposes a vector-wise adversarial training method, which has been shown to significantly improve recommendation performance[39].

• **NeuMF**: Despite being outperformed by the most advanced recommendation models available, NeuMF is still used as a baseline due to its significance as the first model to introduce a neural network into recommendation systems[40].

• **DHCF**: DHCF presents a framework for collaborative filtering using a two-channel hypergraph convolution network, which addresses the issue of inadequate modeling of high-order correlations between users and items in conventional collaborative filtering methods[41].

• **GCMC**: GCMC introduces a graph self-encoder framework designed for matrix completion tasks within recommendation systems. This approach leverages interaction information between users and items to generate hidden features capable of capturing intricate user-item relationships. Empirical results demonstrate that the performance of the proposed model significantly surpasses existing methods in the realm of recommendation systems[42].

• **NGCF**: NGCF is the pioneering model to introduce GNN into recommendation systems. Compared to the previous NeuMF model, NGCF has significantly improved the recommendation the performance and has become a widely used benchmark in the field of recommendation systems[32].

• **LR-GCCF**: LR-GCCF presents an improvement over NGCF that addresses the drawbacks of its non-linear activation function while retaining its characteristic transformation. The proposed approach has been shown to outperform NGCF in terms of recommendation performance[43].

• **LightGCN**: LightGCN presents an improvement over NGCF that discards the feature transformation and non-linear activation of GCNs to achieve better recommendation performance and increased model efficiency[7].

• **IMP-GCN**: IMP-GCN contends that the learning of user embeddings can benefit from integrating high-order neighboring users who may not share common interests with the target user. To realize this, users and their corresponding interaction items are partitioned into distinct subgraphs, and high-order graph convolutions are applied within each subgraph[44].

4.1.2 Evaluation metrics

To evaluate the effectiveness of our approach for top-*n* recommendation task, we employ two commonly used evaluation metrics: recall and Normalized Discounted Cumulative Gain (NDCG). Both metrics have values within the range of 0 to 1, where higher values signify superior performance. Recall is computed by dividing the number of true positives by the sum of true positives and false negatives. A true positive is an item that is correctly identified as relevant, while a false negative is an item that is incorrectly identified as not relevant,

$$\text{recall@}N = \frac{\sum_{v=1}^{V} |R_v \cap T_v|}{\sum_{v=1}^{V} |T_v|} \tag{22}$$

where V is the number of users, R_v is the set of top-N recommendations for user v, T_v is the set of relevant items for user v. In other words, recall quantifies the proportion of relevant items that are successfully recommended to the user. A high recall score indicates that our system is able to retrieve a significant number of relevant items, while a low recall score suggests that there are many relevant items that our system are missed.

NDCG is a metric that evaluates how well a recommender system ranks relevant items. NDCG considers both the relevance level and the position of each item within the recommendation list.

$$\text{NDCG@}N = \frac{\text{DCG@}N}{\text{IDCG@}N} \tag{23}$$

where DCG@N is the discounted cumulative gain at rank N,

$$\text{DCG@}N = \sum_{c=1}^{N} \frac{2^{\text{rel}_c} - 1}{\log_2(c+1)} \tag{24}$$

and IDCG@N is the ideal discounted cumulative gain at rank N,

$$\mathrm{IDCG}@N=\sum_{c=1}^{|\mathcal{R}|}\frac{2^{\mathrm{rel}_c}-1}{\log_2(c+1)} \tag{25}$$

where rel_c is the relevance score of item at rank c, and $\mathcal{R}$ is the set of relevant items.

4.1.3 Parameter settings

The datasets used in our experiments are ml-100k[†] and yelp[‡]. We use outer_steps = 1 and inner_steps = 2 for our GSLRRec model, which means that every time the matrix is updated, the GNN model will be trained twice. We set early_stop to 10 for ml-100k and no early_stop for yelp. The learning rate for the GNN model is set to $l_r = 0.001$, and the learning rate for the training matrix is set to lr_adj = 0.01. We set the weight of the l_1 norm to $\alpha = 5\times10^{-4}$, the weight of the kernel norm to $\beta = 1.5$, and the weight of the BPRloss to $\gamma = 1$ for the objective function. During random attack mode, we set the threshold value δ of ml-100k to 0.005 and the threshold value δ of yelp to 0.1. During Betweenness Centrality (BC) attack mode, we set the δ of both datasets to 0.08. We explore the effect of threshold value δ on the results in more detail in the parameters analysis section. The other hyperparameters in the comparison method retain their original settings.

4.2 Attack methods

In this paper, we adopt two graph attack techniques for the recommendation graph structure:

- **Random attack**: Random attack is a widely used attack method due to its simplicity and versatility. In this study, we randomly perturb the edges of the recommendation graph by adding or deleting interactions between users and items at different proportions. For instance, if there is an interaction between user u_1 and item i_1, we may delete it, or if there is no interaction, we may add it. Despite its apparent simplicity, this attack method has proven to be remarkably effective in compromising the effectiveness of the recommendation system.
- **BC attack**: BC serves as a prominent metric for assessing the significance of individual nodes within a graph. This metric quantifies a node's capacity to function as an intermediary between other nodes, occupying crucial positions on their shortcuts. Elimination of such nodes from the graph may impede communication among remaining nodes. A node's importance to the graph structure is ascertained by the number of positions it occupies. We employ this metric to evaluate the significance of nodes in the graph structure.

BC is a metric in network analysis used to assess the importance of a node in a graph by measuring its involvement in connecting different pairs of nodes. It quantifies the extent to which a node acts as an intermediary in information propagation across the graph. A higher BC for a node implies a more critical role, as it facilitates the flow of information between various nodes. Specifically, BC is defined by calculating the number of occurrences of a node on all shortest paths in the graph. The BC for a node can be expressed as follows:

$$C_B(v)=\sum_{s\neq v\neq t}\frac{\sigma_{st}(v)}{\sigma_{st}} \tag{26}$$

where s and t are two nodes in the graph, σ_{st} represents the number of shortest paths from node s to node t, and $\sigma_{st}(v)$ is the number of those paths that pass through node v. The computation of BC involves traversing all pairs of nodes in the graph and calculating the counts of shortest paths and the counts of paths passing through node v for each pair of nodes s and t. Once the computation is complete, the BC score for each node can be obtained.

BC is a crucial metric that aids in designing adversarial attack methods. Our goal is to strategically target influential nodes in order to undermine the graph's robustness. BC can guide us in selecting nodes for attack since nodes with high BC typically play a critical role in the connectivity and information propagation within the graph. Attacking these nodes may diminish the efficiency of information dissemination.

In this study, we compute the BC for all nodes presented in the user-item interaction graph, and subsequently identify the top 10% of nodes exhibiting the highest scores. We proceed to randomly introduce interacting edges to these nodes in a specific proportion, implementing an attack method predicated on BC[45].

4.3 Overall performance

We conduct experiments to evaluate the recommendation performance of the GSLRRec model and other baseline models under random attack and BC attack on two datasets. We vary the perturbation rate of the edges from 0.1 to 0.5 with a step size of 0.1, and

[†] https://grouplens.org/datasets/movielens/
[‡] https://www.yelp.com/dataset

use recall and NDCG as evaluation metrics. We repeat each experiment 5 times and report the average results in Tables 1 and 2, Figs. 3 and 4. Bold values indicate the best performance.

• The experimental results reveal that the GSLRRec model outperforms other comparison methods across varying perturbation rates. Specifically, on the ml-100k dataset (see Table 1), when the perturbation rate is set to 0.3, the GSLRRec model demonstrates a 12.93% improvement in recall and a 13.08% enhancement in NDCG compared to the LightGCN model. Furthermore, to emphasize the effectiveness of the proposed model, a comparative analysis is conducted with IMP-GCN, which reveals that GSLRRec retains certain advantages, underscoring its superiority over existing models.

• Among the various baseline models, GNN-based recommendation models have shown superior performance over traditional methods due to their high-order connectivity characteristics. This finding is consistent with previous research studies. In the experiments, it is observed that as the attack ratio increases, the recommendation performances of traditional matrix decomposition methods, such as ExpoMF and other models, decrease significantly. In contrast, GNN-based recommendation models, like NGCF and LightGCN， demonstrate better robustness against attacks, but their recommendation performances still decrease. However, compared to these models, the GSLRRec model maintaines relatively good recommendation performances at different attack ratios, indicating that the proposed method exhibits better robustness and generalization.

• LightGCN, a collaborative filtering algorithm derived from the GCN, preserves only the crucial neighborhood aggregation component. User and item embeddings undergo linear propagation across the user-item interaction graph. LightGCN then employs

Table 1 Results of random attack on ml-100k dataset with different perturbation rates (bold indicates maximum value).

Method	0.1		0.2		0.3		0.4		0.5	
	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
ExpoMF	0.1175	0.1017	0.0760	0.0635	0.0594	0.0472	0.0473	0.0371	0.0397	0.0308
CFGAN	0.1698	0.1551	0.1612	0.1571	0.1382	0.1252	0.1159	0.0989	0.1065	0.1001
NeuMF	0.2935	0.2783	0.2763	0.2599	0.2686	0.2581	0.2510	0.2434	0.2475	0.2350
DHCF	0.3168	0.3084	0.2936	0.2821	0.2746	0.2669	0.2473	0.2389	0.2259	0.2105
GCMC	0.3320	0.3473	0.3081	0.2865	0.2809	0.2643	0.2603	0.2410	0.2529	0.2246
NGCF	0.3207	0.2970	0.2945	0.2737	0.2803	0.2533	0.2655	0.2395	0.2690	0.2384
LR-GCCF	0.3158	0.2940	0.2842	0.2593	0.2762	0.2495	0.2717	0.2457	0.2664	0.2358
LightGCN	0.3386	0.3238	0.3233	0.3046	0.3079	0.2935	0.3055	0.2874	0.3039	0.2845
IMP-GCN	0.3402	0.3218	0.3264	0.3079	0.3169	0.2992	0.3102	0.2945	0.3101	0.2963
GSLRRec	**0.3568**	**0.3370**	**0.3517**	**0.3321**	**0.3477**	**0.3319**	**0.3413**	**0.3240**	**0.3428**	**0.3257**

Table 2 Results of random attack on yelp dataset with different perturbation rates (bold indicates maximum value).

Method	0.1		0.2		0.3		0.4		0.5	
	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
ExpoMF	0.1766	0.0978	0.1669	0.0951	0.1538	0.0834	0.1238	0.0724	0.1179	0.0686
CFGAN	0.2179	0.1468	0.2167	0.1405	0.2059	0.1313	0.1884	0.1208	0.1961	0.1286
NeuMF	0.2109	0.1326	0.2061	0.1276	0.1889	0.1226	0.1717	0.1099	0.1688	0.1054
DHCF	0.2215	0.1364	0.2058	0.1273	0.2053	0.1305	0.2015	0.1247	0.1776	0.1148
GCMC	0.2491	0.1485	0.2203	0.1325	0. 2012	0.1259	0.1902	0.1177	0.1777	0.1080
NGCF	0.2351	0.1358	0.2016	0.1186	0.1825	0.1072	0.1723	0.0991	0.1524	0.0894
LR-GCCF	0.2248	0.1327	0.1975	0.1172	0.1796	0.1078	0.1657	0.0965	0.1636	0.0959
LightGCN	0.2582	0.1553	0.2366	0.1466	0.2256	0.1395	0.2209	0.1347	0.2154	0.1307
IMP-GCN	**0.2616**	**0.1622**	0.2472	0.1495	0.2398	0.1505	0.2186	0.1311	0.2126	0.1303
GSLRRec	0.2579	0.1575	**0.2522**	**0.1589**	**0.2415**	**0.1528**	**0.2319**	**0.1488**	**0.2184**	**0.1371**

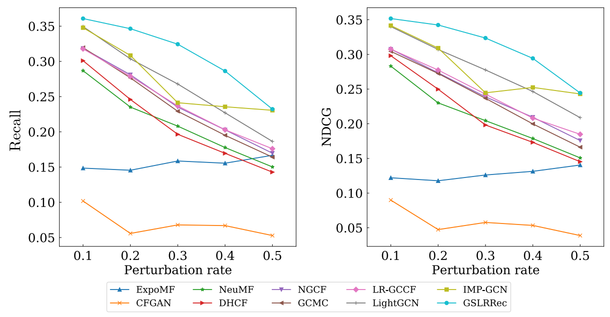

Fig. 3　Result of BC attack on ml-100k dataset.

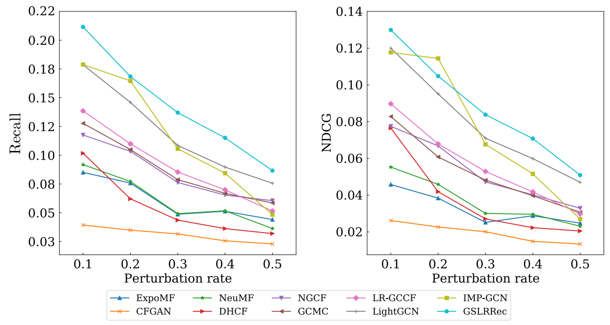

Fig. 4　Result of BC attack on yelp dataset.

the weighted sum of embeddings obtained from all layers to generate the final embedding. The model's simplicity, linearity, and conciseness make it less complex to implement and train, showcasing superior performance in comparison to prior graph-based recommendation models. In the experiments conducted on the ml-100k dataset, when the attack ratios is set at 0.1, the recall metric of LightGCN shows improvements of 15.37%, 5.59%, 7.72% over NeuMF, NGCF, LR-GCCF, respectively (see Table 2).

• The core idea of the proposed model is to jointly perform GSL and downstream recommendation model parameter training. In terms of GSL, the commonalities of graph low rankness and sparsity are used as constraints to purify the user-item interaction graph after being attacked, thereby improving recommendation performance. Compared to the baseline models, GSLRRec model adapts better to different attack situations. In the experiments conducted on the ml-100k dataset, GSLRRec consistently demonstrates superior performance across all attack ratios, outperforming all baseline models in both recall and NDCG evaluation metrics.

• In addition, we also notice the change trend of recommendation performance under different attack ratios. At low attack ratios, the performance differences between various models are small; however, as the attack ratio increases, the performance of the baseline model drops significantly. This also verifies the hypothesis that GSL effectively improves the robustness of recommendation models after the user-item interaction graph is attacked.

In summary, the experimental results demonstrate that the joint learning graph structure and training recommendation model parameters has a positive effect on improving the robustness of recommendation models. These results also provide valuable references and inspirations for future research.

4.4　Ablation study

To incorporate the low-rank and sparsity properties of the graph into the GSLRRec model, we conduct ablation experiments to investigate their impact on the model's performance.

4.4.1　Regularizers

In our ablation experiments, we conduct two variations of GSLRRec by setting the coefficient of the nuclear norm that controls the low-rank property (α) to 0, denoted as GSLRRec-a1, and by setting the coefficient of the l_1 norm that controls sparsity (β) to 0, denoted as GSLRRec-a2. All other parameters are kept unchanged. The evaluation metric selected for presenting the results of these experiments is recall on the ml-100k dataset after BC attack, as the NDCG results are found to be similar. As illustrated in Fig. 5, GSLRRec-a1 and GSLRRec-a2 do not exhibit significant performance improvements at low perturbation rates. In fact, the performance of GSLRRec-a1 is inferior to that of LightGCN when the perturbation rate is set to 0.1. However, as the perturbation rate increases, the performance advantages of GSLRRec-a1 and GSLRRec-a2 gradually become

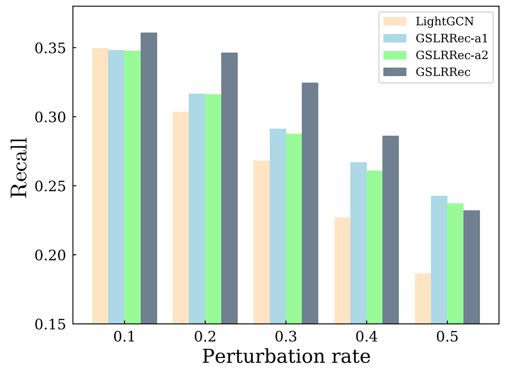

Fig. 5　Experimental results of GSLRRec's variants with different perturbation rate.

apparent. Nevertheless, both variants do not perform as well as the original GSLRRec, indicating that the low-rank and sparsity properties of the graph could only partially restore the graph structure that had been maliciously attacked. This also suggests that graph attacks have the ability to disrupt these inherent properties. Notably, the variant GSLRRec-a2 shows slightly better performance than GSLRRec at a perturbation rate of 0.5, which is an exception to the overall conclusion but does not affect the main finding.

4.4.2 Two-stage vs. one-stage

In the introduction section of this paper, we posit that the traditional two-stage method involves initial GSL followed by training the recommendation model parameters using the obtained graph. In contrast, the joint learning method simultaneously performs GSL and recommendation model parameter training. To empirically demonstrate the superiority of the joint learning method employed in GSLRRec over the two-stage method, we conduct experiments on the ml-100k dataset, and the results are presented in Tables 3 and 4. Our findings reveal that the two-stage method outperforms LightGCN in most cases, thereby affirming that GSL enhances model robustness. Notably, our joint learning method yields even better results than the two-stage method across different attack ratios, thus validating our underlying assumption.

4.5 Hyperparameter analysis for threshold value δ

The threshold value δ is utilized to adjust the elements in the interaction matrix to either 1 or 0, following the matrix update process. Specifically, an element in the interaction matrix is modified to 1 if its value exceeds δ, indicating the presence of an interaction between the user and item. Conversely, if the element is below δ, it is changed to 0, denoting the absence of an interaction between the user and item. This modification facilitates the extraction of positive and negative samples during the training phase of the GNN. It is evident that the value of the parameter δ significantly impacts the final performance of the model. To determine the optimal value of δ that yields the best model performance, we conduct experiments by varying the parameter δ while keeping all other parameters constant.

4.5.1 Without attack

In this study, we evaluate the performance of the GSLRRec model on two datasets, namely ml-100k and yelp, in the absence of any attacks on the user-item interaction graph. We perform experiments by adjusting the parameter δ in the range of [0.01, 0.3], while keeping all other parameters unchanged. Since the GNN part of the GSLRRec model is based primarily on LightGCN, we use LightGCN as the baseline in this part and compare the experimental results with those obtained without attacks. Here, we consider the recall indicator as a reference, and note that the NDCG results are similar. The experimental findings, presented in Fig. 6. Our results indicate that the changes in the δ value have a significant impact on GSLRRec's efficacy for the ml-100k dataset, with GSLRRec outperforming LightGCN when the δ value is within [0.1, 0.2]. Similarly, for the yelp dataset, GSLRRec slightly outperforms LightGCN when the δ value is approximately 1.5. These outcomes are noteworthy, and we attribute the reasons for the results

Table 3 Results of LightGCN, two-stage method, and joint learning method under random attack (bold indicates maximum value).

Method	Perturbation rate									
	0.1		0.2		0.3		0.4		0.5	
	recall	NDCG	recall	NDCG	recall	NDCG	recall	NDCG	recall	NDCG
LightGCN	0.3386	0.3238	0.3233	0.3046	0.3079	0.2935	0.3055	0.2874	0.3039	0.2845
Two-stage	0.3274	0.3100	0.3242	0.3076	0.3157	0.2984	0.3120	0.2981	0.3095	0.2948
Joint learning	**0.3568**	**0.3370**	**0.3517**	**0.3321**	**0.3477**	**0.3319**	**0.3413**	**0.3240**	**0.3428**	**0.3257**

Table 4 Results of LightGCN, two-stage method, and joint learning method under BC attack (bold indicates maximum value).

Method	Perturbation rate									
	0.1		0.2		0.3		0.4		0.5	
	recall	NDCG	recall	NDCG	recall	NDCG	recall	NDCG	recall	NDCG
LightGCN	0.3494	0.3401	0.3036	0.3071	0.2679	0.2776	0.2270	0.2465	0.1864	0.2088
Two-stage	0.3457	0.3386	0.3145	0.3202	0.2918	0.3054	0.2605	0.2781	**0.2374**	**0.2601**
Joint learning	**0.3608**	**0.3517**	**0.3464**	**0.3424**	**0.3244**	**0.3236**	**0.2863**	**0.2943**	0.2322	0.2444

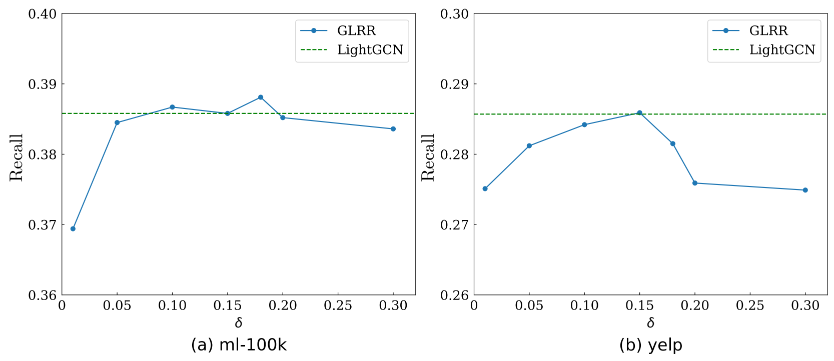

Fig. 6 Experimental results of GSLRRec when the user-item interaction graph is not attacked.

to the existence of noise in the original user-item interaction graph. In reality, the interaction between u and i may indicate user u's negative evaluation of i rather than an interest in i. We refer to this phenomenon as natural false click noise in the original user-item interaction graph. Our experimental findings indicate that the GSLRRec model has high robustness and effectively reduces noise in the original user-item interaction graph.

4.5.2 With attack

In this section, we evaluate the performance of the GSLRRec model on the ml-100k dataset with an attack ratio of 0.1 as an example, and found that the results obtained from other datasets and attack ratios are similar. The experimental outcomes are presented in Tables 5 and 6, from which we derive the following conclusions: Under the random attack mode, GSLRRec achieves the best performance when $\delta = 0.005$, whereas under the BC attack mode, optimal performance is observed at $\delta = 0.080$. The value of parameter δ used in the main experimental section of this study is determined based on these results. To gain insights into the effect of δ on the model's performance, we analyze the value distribution of elements in the updated user-item interaction graph of the ml-100k dataset. Our analysis reveals that over 90% of the non-zero data fell within the range of (0, 0.01), and the value of δ at which the model achieved the best performance fells within this range. This finding suggests that we can determine the value of δ that is closest to the optimal performance of the model based on the numerical distribution law of the interaction graph. This direction represents one of our future research endeavors.

Table 5 GSLRRec performance under random attack (bold indicates maximum value).

δ	Recall	NDCG
0.001	0.3512	0.3295
0.005	**0.3557**	**0.3331**
0.010	0.3549	0.3328
0.100	0.3401	0.3204
0.500	0.3402	0.3218

Table 6 GSLRRec performance under BC attack (bold indicates maximum value).

δ	Recall	NDCG
0.001	0.3137	0.2952
0.010	0.3159	0.2967
0.080	**0.3608**	**0.3517**
0.100	0.3533	0.3464
0.500	0.3513	0.3439

5 Related Work

In this section, we provide a concise overview of the relevant literature in the areas of recommendation systems, adversarial attacks, and defense mechanisms for recommendation systems.

5.1 Recommendation systems

The rapid expansion of the Internet has generated an enormous volume of data, attributed to the swift advancements in cloud computing and big data technology. While this large scale data contain valuable information that promotes the rapid development of society, it also brings serious information overload problems. In this context, the recommendation system has emerged as a widely used tool to solve the problem of information overload and has been applied by academia and industry[46]. The recommendation system aim to mine personalized interest from a vast amount of data based on the user's historical behavior and personal characteristics. The recommendation system has been employed in various fields, such as e-commerce, news recommendation, and information retrieval. Traditional recommendation methods include collaborative filtering, content-based recommendation, and mixed recommendation[47–49]. Collaborative filtering, such as the matrix factorization algorithm, is the most classical method that recommends new items based on the user's historical interaction with the item. The recommendation system with the matrix factorization algorithm as its core has been widely used. However, the recommendation system has faced several issues, such as data sparseness and cold start, that have been criticized. Content-based recommendation systems and mixed recommendation methods also have limited effectiveness and scalability, restricting the performance of recommendation[11].

In recent years, GNNs have witnessed significant advancements in diverse domains, such as image processing, natural language understanding, and speech recognition, leading to increased attention in the Artificial Intelligence (AI) research community[50]. The core principle of GNNs involves iterative feature aggregation from neighboring nodes and subsequent integration of the aggregated information with the central node representation. GCNs employ first-order local convolution for iterative information propagation from neighboring nodes. Notably, Zhang et al.[10] is the first to apply GCN in recommendation systems, leading to subsequent proposals of GNN-based recommendation systems that have demonstrated improved performance. Wang et al.[32] introduced the Neural Graph Collaborative Filtering (NGCF) method, which incorporates higher-order connectivity of the graph to enhance recommendation systems, resulting in improved performance. Subsequently, the Linear Residual Graph Convolutional Collaborative Filtering (LR-GCCF) method[43] further enhances NGCF's non-linear activation function by incorporating the concept of residual and improving the information aggregation mechanism. LightGCN method[7] improves upon previous methods by considering that non-linear activation functions and feature transformation matrices are not essential for collaborative filtering, and excluding them from the model significantly enhanced the performance. Fan et al.[51] suggested that trustworthy recommender systems should be evaluated from six aspects: safety and robustness, nondiscrimination and fairness, explainability, privacy, environmental well-being, accountability, and auditability. The adoption of GNN-based recommendation systems has shown substantial enhancements in recommendation performance. However, as the usage of GNN-based recommendation systems has expanded, the presence of robustness issues has become increasingly apparent[52]. Even minor modifications to graph data can result in substantial deviations in recommendation accuracy, as GNN models have an amplifying effect on such changes. Consequently, this vulnerability presents an opportunity for malicious attackers to attain maximum impact while incurring minimal costs.

5.2 Adversarial attacks and defenses of recommender systems

The field of GNNs has witnessed remarkable progress in node classification, natural language processing, and other domains[53]. However, the vulnerability of GNN models to noise attacks has become a growing concern[54]. Zügner et al.[14] introduced a meta-method to attack GNN models, resulting in a significant reduction in the accuracy of downstream node classification. There are few effective methods to prevent malicious edge-adding or edge-reducing disturbance attacks on the user-item interactive adjacency matrix generated by the recommendation system. To address this challenge, Yuan et al.[55] proposed a general adversarial training framework that enhances both model robustness and overall performance while achieving a trade-off between them. Yuan and Yao[56] developed a new framework to reduce the impact of original graph data noise. Chen et al.[57] introduced the RAT framework, which enhances the robustness and recommendation performance of a deep collaborative filtering model by training it through adversarial noise propagation. Additionally, Tang et al.[58] highlighted the vulnerabilities of multimedia recommendation systems and proposed a robust recommendation system called AMR, which maintains stable recommendation accuracy rates under adversarial attacks. Other researchers have also proposed various methods to improve the robustness of GNN recommendation systems. For example, Deldjoo et al.[59] introduced the AML-RecSys model, which is based on the characteristics of existing recommendation models that are susceptible to perturbation. The main contribution of this model is the introduction of adversarial regularization, which improves the system's resilience against adversarial attacks. However, despite this progress, more research is needed to enhance the robustness of GNN recommendation systems. In contrast to the recommended system defense methods mentioned earlier, our objective is to recover clean graphs by utilizing critical graph properties while learning GCN parameters. This approach allows our models to maintain high stability under different attacks, ensuring excellent recommendation performance while maximizing the restoration of user-item interaction graphs that have been compromised by adversarial attacks.

5.3 GSL

GNNs serve as a practical model for analyzing graph-structured data; however, their performances are intricately tied to the quality of the underlying graph structure. The presence of noise in the graph structure can significantly hinder the performance of GNNs and negatively impact their downstream tasks. To tackle this challenge, the field of GSL has emerged as a pivotal task, striving to simultaneously learn an optimized graph structure and its corresponding representation. Research in the field of GSL has been progressing, leading to the development of various GNNs. For instance, GLCN is a unified network structure that combines graph learning and convolution for semi-supervised learning. Its objective is to acquire the optimal graph representation for graph[60]. Another example is HGSL, a GNN for heterogeneous graph structure learning, which generates a heterographic structure suitable for downstream tasks, and learns GNN parameters jointly by mining feature similarity, interaction between features and structures, and higher-order semantic structure in the heterographic graph[61]. Furthermore, GLNN provides a solution to the industry problem of high latency by combining the low latency and MAP-free dependence of MLP with the prediction accuracy of GNN[62]. Pro-GNN presents a framework of joint training graph structure and GNN models that achieve excellent performance in node classification tasks[31].

6 Conclusion and Future Work

The recommendation performance of GNN-based recommendation systems can be significantly compromised by adversarial attacks. In response to this issue, we introduce a resilient GNN-based recommendation model, GSLRRec, which comprises two components: GSL and GNN training. The GSL aims to restore the user-item interaction graph after an attack by exploiting the low rank and sparsity of the recommendation graph. Concurrently, during the GNN training phase, model parameters are trained according to the high-order connectivity of users and items to obtain recommendation results. A key feature of GSLRRec is its ability to learn new user-item interaction graphs and train GNN parameters simultaneously, as opposed to the traditional two-stage training method. Across different datasets and attack ratios, the GSLRRec model consistently improves the recommendation performance compared to the baselines. Moreover, even without being attacked, GSLRRec outperforms the baselines when an appropriate threshold value δ is used, demonstrating its ability to remove the natural noise from the original graph and its robustness.

The GSLRRec model has a drawback in that its GSL component employs dense matrices, leading to insufficient GPU memory during model training. In the future, we plan to address this issue to enable the model's application to datasets with large numbers of users and items. Additionally, we will explore the optimal value of the parameter δ in the model, which is an important direction for future work. Finally, our proposed model can be easily transferred, and its GNN training part can be replaced with any advanced GNN-based recommendation model, which is expected to enhance the robustness of the original recommendation model.

The GSLRRec model we proposed can be viewed as a framework comprising two main components: GSL and downstream graph recommendation tasks. The specific operational process involves optimizing the graph structure through GSL to restore the graph to its pre-attack state as much as possible. Subsequently, the optimized graph is fed into the downstream recommendation model to train the GNN parameters. This process iterates through each epoch, ultimately yielding the optimal results. This constitutes the essence of our framework.

Our framework is designed to enhance the robustness of recommendation systems. When the downstream task shifts to node classification tasks or tasks in the computer vision domain, modifications to the downstream task component of the GSLRRec framework are sufficient. The first step remains GSL to optimize the graph structure, followed by utilizing the optimized graph in the downstream task. The overall approach remains unchanged, making the GSLRRec easily transferable.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 62272001 and 62206002), the Anhui Provincial Natural Science Foundation (No. 2208085QF195), the Hefei Key Common Technology Project (No. GJ2022GX15), the University Collaborative Innovation Project of Anhui Province (No. GXXT-2021-087), and the Anhui Province Key Research and Development Program (No. 202104a05020058).

References

[1] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, Graph neural networks in recommender systems: A survey, *ACM Comput. Surv.*, vol. 55, no. 5, p. 97, 2023.

[2] L. Wu, X. He, X. Wang, K. Zhang, and M. Wang, A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation, *IEEE Trans. Knowledge Data Eng.*, vol. 35, no. 5, pp. 4425–4445, 2023.

[3] F. Ricci, L. Rokach, and B. Shapira, Recommender systems: Techniques, applications, and challenges, in *Recommender Systems Handbook* (3rd ed.) , F. Ricci, L. Rokach, and B. Shapira, eds., New York, NY, USA: Springer, 2022, pp. 1–35.

[4] L. He, X. Wang, D. Wang, H. Zou, H. Yin, and G. Xu, Simplifying graph-based collaborative filtering for recommendation, in *Proc. 16th ACM Int. Conf. Web Search and Data Mining*, Singapore, 2023, pp. 60–68.

[5] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, Graph neural networks for social recommendation, in *Proc. the World Wide Web Conf.*, San Francisco, CA, USA, 2019, pp. 417–426.

[6] W. Fan, Y. Ma, Q. Li, J. Wang, G. Cai, J. Tang, and D. Yin, A graph neural network framework for social recommendations, *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 5, pp. 2033–2047, 2022.

[7] X. He, K. Deng, X. Wang, Y. Li, Y. D. Zhang, and M. Wang, LightGCN: Simplifying and powering graph convolution network for recommendation, in *Proc. 43rd Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Virtual Event, 2020, pp. 639–648.

[8] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, A comprehensive survey on graph neural networks, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, 2021.

[9] Y. Zhang, Y. Zhang, D. Yan, S. Deng, and Y. Yang, Revisiting graph-based recommender systems from the perspective of variational auto-encoder, *ACM Trans. Inf. Syst.*, vol. 41, no. 3, p. 81, 2022.

[10] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, Graph convolutional networks: A comprehensive review, *Comput. Soc. Netw.*, vol. 6, no. 1, p. 11, 2019.

[11] A. A. Kardan and M. Ebrahimi, A novel approach to hybrid recommendation systems based on association rules mining for content recommendation in asynchronous discussion groups, *Inf. Sci.*, vol. 219, pp. 93–110, 2013.

[12] C. Gao, X. Wang, X. He, and Y. Li, Graph neural networks for recommender system, in *Proc. 15th ACM Int. Conf. Web Search and Data Mining*, Tempe, AZ, USA, 2022, pp. 1623–1625.

[13] W. Jin, Y. Li, H. Xu, Y. Wang, S. Ji, C. Aggarwal, and J. Tang, Adversarial attacks and defenses on graphs, *ACM SIGKDD Explor. Newsl.*, vol. 22, no. 2, pp. 19–34, 2021.

[14] D. Zügner, A. Akbarnejad, and S. Günnemann, Adversarial attacks on neural networks for graph data, in *Proc. 24th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, London, UK, 2018, pp. 2847–2856.

[15] F. Zhang and S. Wang, Detecting group shilling attacks in online recommender systems based on bisecting K-means clustering, *IEEE Trans. Conput. Soc. Syst.*, vol. 7, no. 5, pp. 1189–1199, 2020.

[16] C. Lin, S. Chen, H. Li, Y. Xiao, L. Li, and Q. Yang, Attacking recommender systems with augmented user profiles, in *Proc. 29th ACM Int. Conf. Information & Knowledge Management*, Virtual Event, 2020, pp. 855–864.

[17] F. Wang, H. Zhu, G. Srivastava, S. Li, M. R. Khosravi, and L. Qi, Robust collaborative filtering recommendation with user-item-trust records, *IEEE Trans. Comput. Soc. Syst.*, vol. 9, no. 4, pp. 986–996, 2022.

[18] S. Zhao, W. Wang, Z. Du, J. Chen, and Z. Duan, A black-box adversarial attack method via nesterov accelerated gradient and rewiring towards attacking graph neural networks, *IEEE Trans. Big Data*, vol. 9, no. 6, pp. 1586–1597, 2023.

[19] W. Fan, T. Derr, X. Zhao, Y. Ma, H. Liu, J. Wang, J. Tang, and Q. Li, Attacking black-box recommendations via copying cross-domain user profiles, in *Proc. 37th IEEE Int. Conf. Data Engineering*, Chania, Greece, 2021, pp. 1583–1594.

[20] J. Chen, W. Fan, G. Zhu, X. Zhao, C. Yuan, Q. Li, and Y. Huang, Knowledge-enhanced black-box attacks for recommendations, in *Proc. 28th ACM SIGKDD Conf. Knowledge Discovery and Data Mining*, Washington, DC, USA, 2022, pp. 108–117.

[21] W. Fan, H. Xu, W. Jin, X. Liu, X. Tang, S. Wang, Q. Li, J. Tang, J. Wang, and C. Aggarwal, Jointly attacking graph neural network and its explanations, in *Proc. 39th IEEE Int. Conf. Data Engineering*, Anaheim, CA, USA, 2023, pp. 654–667.

[22] H. Xu, Y. Ma, H. C. Liu, D. Deb, H. Liu, J. L. Tang, and A. K. Jain, Adversarial attacks and defenses in images, graphs and text: A review, *Int. J. Autom. Comput.*, vol. 17, no. 2, pp. 151–178, 2020.

[23] M. Fang, G. Yang, N. Z. Gong, and J. Liu, Poisoning attacks to graph-based recommender systems, in *Proc. 34th Annu. Computer Security Applications Conf.*, San Juan,

PR, USA, 2018, pp. 381–392.

[24] M. Aktukmak, Y. Yilmaz, and I. Uysal, Quick and accurate attack detection in recommender systems through user attributes, in *Proc. 13th ACM Conf. Recommender Systems*, Copenhagen, Denmark, 2019, pp. 348–352.

[25] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, A comprehensive survey on graph anomaly detection with deep learning, *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 12, pp. 12012–12038, 2023.

[26] M. Pang, W. Gao, M. Tao, and Z. H. Zhou, Unorganized malicious attacks detection, in *Proc. 32nd Int. Conf. Neural Information Processing Systems*, Montréal, Canada, 2018, pp. 6976–6985.

[27] Y. Zhang, F. Regol, S. Pal, S. Khan, L. Ma, and M. Coates, Detection and defense of topological adversarial attacks on graphs, in *Proc. 24th Int. Conf. Artificial Intelligence and Statistics*, Virtual Event, 2021, pp. 2989–2997.

[28] Y. Zhu, W. Xu, J. Zhang, Q. Liu, S. Wu, and L. Wang, Deep graph structure learning for robust representations: A survey, arXiv preprint arXiv: 2103.03036, 2021.

[29] Y. Lai and K. Zhou, How fraudster detection contributes to robust recommendation, https://api.semanticscholar.org/CorpusID:253734623, 2023.

[30] V. W. Anelli, Y. Deldjoo, T. DiNoia, and F. A. Merra, Adversarial recommender systems: Attack, defense, and advances, in *Recommender Systems Handbook* (3rd ed.), F. Ricci, L. Rokach, and B. Shapira, eds., (3rd ed.) New York, NY, USA: Springer, 2022, pp. 335–379.

[31] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, Graph structure learning for robust graph neural networks, in *Proc. 26th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining*, Virtual Event, 2020, pp. 66–74.

[32] X. Wang, X. He, M. Wang, F. Feng, and T. S. Chua, Neural graph collaborative filtering, in *Proc. 42nd Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Paris, France, 2019, pp. 165–174.

[33] H. Xu, L. Xiang, J. Yu, A. Cao, and X. Wang, Speedup robust graph structure learning with low-rank information, in *Proc. 30th ACM Int. Conf. Information & Knowledge Management*, Virtual Event, 2021, pp. 2241–2250.

[34] N. Entezari, S. A. Al-Sayouri, A. Darvishzadeh, and E. E. Papalexakis, All you need is low (rank): Defending against adversarial attacks on graphs, in *Proc. 13th Int. Conf. Web Search and Data Mining*, Houston, TX, USA, 2020, pp. 169–177.

[35] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[36] T. Goldstein, C. Studer, and R. Baraniuk, A field guide to forward-backward splitting with a FASTA implementation, arXiv preprint arXiv: 1411.3406, 2014.

[37] Y. Li, W. Jin, H. Xu, and J. Tang, DeepRobust: A platform for adversarial attacks and defenses, in *Proc. 35th AAAI Conf. Artificial Intelligence*, Virtual Event, 2021, pp. 16078–16080.

[38] D. Liang, L. Charlin, J. McInerney, and D. M. Blei, Modeling user exposure in recommendation, in *Proc. 25th Int. Conf. World Wide Web*, Montréal, Canada, 2016, pp. 951–961.

[39] D. K. Chae, J. S. Kang, S. W. Kim, and J. T. Lee, CFGAN: A generic collaborative filtering framework based on generative adversarial networks, in *Proc. 27th ACM Int. Conf. Information and Knowledge Management*, Torino, Italy, 2018, pp. 137–146.

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

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

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

[43] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach, in *Proc. 34th AAAI Conf. Artificial Intelligence*, New York, NY, USA, 2020, pp. 27–34.

[44] F. Liu, Z. Cheng, L. Zhu, Z. Gao, and L. Nie, Interest-aware message-passing GCN for recommendation, in *Proc. 30th Web Conf.*, Ljubljana, Slovenia, 2021, pp. 1296–1305.

[45] C. Pu, K. Wang, and Y. Xia, Robustness of link prediction under network attacks, *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 67, no. 8, pp. 1472–1476, 2020.

[46] S. Zhang, L. Yao, A. Sun, and Y. Tay, Deep learning based recommender system: A survey and new perspectives, *ACM Comput. Surv.*, vol. 52, no. 1, p. 5, 2020.

[47] Y. Koren, R. Bell, and C. Volinsky, Matrix factorization techniques for recommender systems, *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[48] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, Collaborative filtering recommender systems, in *The Adaptive Web: Methods and Strategies of Web Personalization*, P. Brusilovsky, A. Kobsa, and W. Nejdl, eds. Berlin, Germany: Springer, 2007, pp. 291–324.

[49] P. Lops, M. de Gemmis, and G. Semeraro, Content-based recommender systems: State of the art and trends, in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, eds. New York, NY, USA: Springer, 2011, pp. 73–105.

[50] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, Graph neural networks: A review of methods and applications, *AI Open*, vol. 1, pp. 57–81, 2020.

[51] W. Fan, X. Zhao, X. Chen, J. Su, J. Gao, L. Wang, Q. Liu, Y. Wang, H. Xu, L. Chen, et al., A comprehensive survey on trustworthy recommender systems, arXiv preprint arXiv: 2209.10117, 2022.

[52] B. Ding, R. Zhang, L. Xu, G. Liu, S. Yang, Y. Liu, and Q. Zhang, U^2D^2Net: Unsupervised unified image dehazing and denoising network for single hazy image enhancement, *IEEE Trans. Multimedia*, vol. 26, pp. 202–217, 2024.

[53] R. Zhang, L. Xu, Z. Yu, Y. Shi, C. Mu, and M. Xu, Deep-IRTarget: An automatic target detector in infrared imagery using dual-domain feature extraction and allocation, *IEEE Trans. Multimedia*, vol. 24, pp. 1735–1749, 2022.

[54] R. Zhang, L. Wu, Y. Yang, W. Wu, Y. Chen, and M. Xu, Multi-camera multi-player tracking with deep player identification in sports video, *Pattern Recognit.*, vol. 102, p. 107260, 2020.

[55] F. Yuan, L. Yao, and B. Benatallah, Adversarial collaborative neural network for robust recommendation, in *Proc. 42nd Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Paris, France, 2019, pp. 1065–1068.

[56] F. Yuan, L. Yao, and B. Benatallah, Adversarial collaborative auto-encoder for top-N recommendation, in *Proc. 2019 Int. Joint Conf. Neural Networks*, Budapest, Hungary, 2019, pp. 1–8.

[57] H. Chen, F. Qian, C. Liu, Y. Zhang, H. Su, and S. Zhao, Training robust deep collaborative filtering models via adversarial noise propagation, *ACM Trans. Inf. Syst.*, vol. 42, no. 1, p. 9, 2024.

[58] J. Tang, X. Du, X. He, F. Yuan, Q. Tian, and T. S. Chua, Adversarial training towards robust multimedia recommender system, *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 5, pp. 855–867, 2020.

[59] Y. Deldjoo, T. Di Noia, and F. A. Merra, Adversarial machine learning in recommender systems (AML-RecSys), in *Proc. 13th Int. Conf. Web Search and Data Mining*, Houston, TX, USA, 2020, pp. 869–872.

[60] B. Jiang, Z. Zhang, D. Lin, and J. Tang, Graph learning-convolutional networks, arXiv preprint arXiv: 1811.09971, 2018.

[61] J. Zhao, X. Wang, C. Shi, B. Hu, G. Song, and Y. Ye, Heterogeneous graph structure learning for graph neural networks, in *Proc. 35th AAAI Conf. Artificial Intelligence*, Virtual Event, 2021, pp. 4697–4705.

[62] S. Zhang, Y. Liu, Y. Sun, and N. Shah, Graph-less neural networks: Teaching old MLPs new tricks via distillation, in *Proc. 10th Int. Conf. Learning Representations*, https://doi.org/10.48550/arXiv.2110.08727, 2023.

Lei Sang received the PhD degree from University of Technology Sydney, Australia in 2021. He is currently a lecturer at School of Computer Science and Technology, Anhui University, China. His current research interests include natural language processing, data mining, and recommendation systems. In recent years, he has published multiple papers in international journals and conferences, led a project supported by the National Natural Science Foundation of China.

Yuee Huang received the PhD degree in public health from Anhui University of Science & Technology, China in 2016. She is currently a full professor and a master instructor at School of Public Health, Wannan Medical College, China. She is the director of academic affairs office and the academic and technical leader of Wannan Medical College, China. Her research interests include maternal, child and adolescent health, epidemiology and health statistics, etc. Please see more information on the website https://ph.wnmc.edu.cn/.

Hang Yuan received the BEng degree in information management and information systems from Anhui University, China in 2020. Now he is a master student in computer technology at Anhui University. He received the Outstanding Graduate Award in 2022. His research interests include graph neural networks, recommendation systems, and graph structure learning.

Yiwen Zhang received the PhD degree in management science and engineering from Hefei University of Technology, China in 2013. He is currently a full professor at School of Computer Science and Technology, Anhui University, China. He has published more than 70 papers in highly regarded conferences and journals. including *IEEE TKDE, IEEE TMC, IEEE TSC, ACM TOIS, IEEE TNNLS, ACM TKDD*, ICSOC, ICWS, etc. His research interests include service computing, cloud computing, and big data analytics. Please see more information on the website http://bigdata.ahu.edu.cn/.